Apostolos Antonacopoulos ·
Subhasis Chaudhuri · Rama Chellappa ·
Cheng-Lin Liu · Saumik Bhattacharya ·
Umapada Pal (Eds.)

LNCS 15306

Pattern Recognition

27th International Conference, ICPR 2024
Kolkata, India, December 1–5, 2024
Proceedings, Part VI

6 Part VI

ICPR 2024 INDIA

IAPR

Springer

MOREMEDIA

Lecture Notes in Computer Science **15306**

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

Apostolos Antonacopoulos ·
Subhasis Chaudhuri · Rama Chellappa ·
Cheng-Lin Liu · Saumik Bhattacharya ·
Umapada Pal
Editors

Pattern Recognition

27th International Conference, ICPR 2024
Kolkata, India, December 1–5, 2024
Proceedings, Part VI

Springer

Editors
Apostolos Antonacopoulos
University of Salford
Salford, UK

Subhasis Chaudhuri
Indian Institute of Technology Bombay
Mumbai, India

Rama Chellappa
Johns Hopkins University
Baltimore, MD, USA

Cheng-Lin Liu
Chinese Academy of Sciences
Beijing, China

Saumik Bhattacharya
IIT Kharagpur
Kharagpur, India

Umapada Pal
Indian Statistical Institute Kolkata
Kolkata, India

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-78171-1 ISBN 978-3-031-78172-8 (eBook)
https://doi.org/10.1007/978-3-031-78172-8

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

President's Address

On behalf of the Executive Committee of the International Association for Pattern Recognition (IAPR), I am pleased to welcome you to the 27th International Conference on Pattern Recognition (ICPR 2024), the main scientific event of the IAPR.

After a completely digital ICPR in the middle of the COVID pandemic and the first hybrid version in 2022, we can now enjoy a fully back-to-normal ICPR this year. I look forward to hearing inspirational talks and keynotes, catching up with colleagues during the breaks and making new contacts in an informal way. At the same time, the conference landscape has changed. Hybrid meetings have made their entrance and will continue. It is exciting to experience how this will influence the conference. Planning for a major event like ICPR must take place over a period of several years. This means many decisions had to be made under a cloud of uncertainty, adding to the already large effort needed to produce a successful conference. It is with enormous gratitude, then, that we must thank the team of organizers for their hard work, flexibility, and creativity in organizing this ICPR. ICPR always provides a wonderful opportunity for the community to gather together. I can think of no better location than Kolkata to renew the bonds of our international research community.

Each ICPR is a bit different owing to the vision of its organizing committee. For 2024, the conference has six different tracks reflecting major themes in pattern recognition: Artificial Intelligence, Pattern Recognition and Machine Learning; Computer and Robot Vision; Image, Speech, Signal and Video Processing; Biometrics and Human Computer Interaction; Document Analysis and Recognition; and Biomedical Imaging and Bioinformatics. This reflects the richness of our field. ICPR 2024 also features two dozen workshops, seven tutorials, and 15 competitions; there is something for everyone. Many thanks to those who are leading these activities, which together add significant value to attending ICPR, whether in person or virtually. Because it is important for ICPR to be as accessible as possible to colleagues from all around the world, we are pleased that the IAPR, working with the ICPR organizers, is continuing our practice of awarding travel stipends to a number of early-career authors who demonstrate financial need. Last but not least, we are thankful to the Springer LNCS team for their effort to publish these proceedings.

Among the presentations from distinguished keynote speakers, we are looking forward to the three IAPR Prize Lectures at ICPR 2024. This year we honor the achievements of Tin Kam Ho (IBM Research) with the IAPR's most prestigious King-Sun Fu Prize "for pioneering contributions to multi-classifier systems, random decision forests, and data complexity analysis". The King-Sun Fu Prize is given in recognition of an outstanding technical contribution to the field of pattern recognition. It honors the memory of Professor King-Sun Fu who was instrumental in the founding of IAPR, served as its first president, and is widely recognized for his extensive contributions to the field of pattern recognition.

The Maria Petrou Prize is given to a living female scientist/engineer who has made substantial contributions to the field of Pattern Recognition and whose past contributions, current research activity and future potential may be regarded as a model to both aspiring and established researchers. It honours the memory of Professor Maria Petrou as a scientist of the first rank, and particularly her role as a pioneer for women researchers. This year, the Maria Petrou Prize is given to Guoying Zhao (University of Oulu), "for contributions to video analysis for facial micro-behavior recognition and remote biosignal reading (RPPG) for heart rate analysis and face anti-spoofing".

The J.K. Aggarwal Prize is given to a young scientist who has brought a substantial contribution to a field that is relevant to the IAPR community and whose research work has had a major impact on the field. Professor Aggarwal is widely recognized for his extensive contributions to the field of pattern recognition and for his participation in IAPR's activities. This year, the J.K. Aggarwal Prize goes to Xiaolong Wang (UC San Diego) "for groundbreaking contributions to advancing visual representation learning, utilizing self-supervised and attention-based models to establish fundamental frameworks for creating versatile, general-purpose pattern recognition systems".

During the conference we will also recognize 21 new IAPR Fellows selected from a field of very strong candidates. In addition, a number of Best Scientific Paper and Best Student Paper awards will be presented, along with the Best Industry Related Paper Award and the Piero Zamperoni Best Student Paper Award. Congratulations to the recipients of these very well-deserved awards!

I would like to close by again thanking everyone involved in making ICPR 2024 a tremendous success; your hard work is deeply appreciated. These thanks extend to all who chaired the various aspects of the conference and the associated workshops, my ExCo colleagues, and the IAPR Standing and Technical Committees. Linda O'Gorman, the IAPR Secretariat, deserves special recognition for her experience, historical perspective, and attention to detail when it comes to supporting many of the IAPR's most important activities. Her tasks became so numerous that she recently got support from Carolyn Buckley (layout, newsletter), Ugur Halici (ICPR matters), and Rosemary Stramka (secretariat). The IAPR website got a completely new design. Ed Sobczak has taken care of our web presence for so many years already. A big thank you to all of you!

This is, of course, the 27th ICPR conference. Knowing that ICPR is organized every two years, and that the first conference in the series (1973!) pre-dated the formal founding of the IAPR by a few years, it is also exciting to consider that we are celebrating over 50 years of ICPR and at the same time approaching the official IAPR 50th anniversary in 2028: you'll get all information you need at ICPR 2024. In the meantime, I offer my thanks and my best wishes to all who are involved in supporting the IAPR throughout the world.

September 2024

Arjan Kuijper
President of the IAPR

Preface

It is our great pleasure to welcome you to the proceedings of the 27th International Conference on Pattern Recognition (ICPR 2024), held in Kolkata, India. The city, formerly known as 'Calcutta', is the home of the fabled Indian Statistical Institute (ISI), which has been at the forefront of statistical pattern recognition for almost a century. Concepts like the Mahalanobis distance, Bhattacharyya bound, Cramer–Rao bound, and Fisher–Rao metric were invented by pioneers associated with ISI. The first ICPR (called IJCPR then) was held in 1973, and the second in 1974. Subsequently, ICPR has been held every other year. The International Association for Pattern Recognition (IAPR) was founded in 1978 and became the sponsor of the ICPR series. Over the past 50 years, ICPR has attracted huge numbers of scientists, engineers and students from all over the world and contributed to advancing research, development and applications in pattern recognition technology.

ICPR 2024 was held at the Biswa Bangla Convention Centre, one of the largest such facilities in South Asia, situated just 7 kilometers from Kolkata Airport (CCU). According to ChatGPT "Kolkata is often called the 'Cultural Capital of India'. The city has a deep connection to literature, music, theater, and art. It was home to Nobel laureate Rabindranath Tagore, and the Bengali film industry has produced globally renowned filmmakers like Satyajit Ray. The city boasts remarkable colonial architecture, with landmarks like Victoria Memorial, Howrah Bridge, and the Indian Museum (the oldest and largest museum in India). Kolkata's streets are dotted with old mansions and buildings that tell stories of its colonial past. Walking through the city can feel like stepping back into a different era. Finally, Kolkata is also known for its street food."

ICPR 2024 followed a two-round paper submission format. We received a total of 2135 papers (1501 papers in round-1 submissions, and 634 papers in round-2 submissions). Each paper, on average, received 2.84 reviews, in single-blind mode. For the first-round papers we had a rebuttal option available to authors.

In total, 945 papers (669 from round-1 and 276 from round-2) were accepted for presentation, resulting in an acceptance rate of 44.26%, which is consistent with previous ICPR events. At ICPR 2024 the papers were categorized into six tracks: Artificial Intelligence, Machine Learning for Pattern Analysis; Computer Vision and Robotic Perception; Image, Video, Speech, and Signal Analysis; Biometrics and Human-Machine Interaction; Document and Media Analysis; and Biomedical Image Analysis and Informatics.

The main conference ran over December 2–5, 2024. The main program included the presentation of 188 oral papers (19.89% of the accepted papers), 757 poster papers and 12 competition papers (out of 15 submitted). A total 10 oral sessions were held concurrently in four meeting rooms with a total of 40 oral sessions. In total 24 workshops and 7 tutorials were held on December 1, 2024.

The plenary sessions included three prize lectures and three invited presentations. The prize lectures were delivered by Tin Kam Ho (IBM Research, USA; King Sun

Fu Prize winner), Xiaolong Wang (University of California, San Diego, USA; J.K. Aggarwal Prize winner), and Guoying Zhao (University of Oulu, Finland; Maria Petrou Prize winner). The invited speakers were Timothy Hospedales (University of Edinburgh, UK), Venu Govindaraju (University at Buffalo, USA), and Shuicheng Yan (Skywork AI, Singapore).

Several best paper awards were presented in ICPR: the Piero Zamperoni Award for the best paper authored by a student, the BIRPA Best Industry Related Paper Award, and the Best Paper Awards and Best Student Paper Awards for each of the six tracks of ICPR 2024.

The organization of such a large conference would not be possible without the help of many volunteers. Our special gratitude goes to the Program Chairs (Apostolos Antonacopoulos, Subhasis Chaudhuri, Rama Chellappa and Cheng-Lin Liu), for their leadership in organizing the program. Thanks to our Publication Chairs (Ananda S. Chowdhury and Wataru Ohyama) for handling the overwhelming workload of publishing the conference proceedings. We also thank our Competition Chairs (Richard Zanibbi, Lianwen Jin and Laurence Likforman-Sulem) for arranging 12 important competitions as part of ICPR 2024. We are thankful to our Workshop Chairs (P. Shivakumara, Stephanie Schuckers, Jean-Marc Ogier and Prabir Bhattacharya) and Tutorial Chairs (B.B. Chaudhuri, Michael R. Jenkin and Guoying Zhao) for arranging the workshops and tutorials on emerging topics. ICPR 2024, for the first time, held a Doctoral Consortium. We would like to thank our Doctoral Consortium Chairs (Véronique Eglin, Dan Lopresti and Mayank Vatsa) for organizing it.

Thanks go to the Track Chairs and the meta reviewers who devoted significant time to the review process and preparation of the program. We also sincerely thank the reviewers who provided valuable feedback to the authors.

Finally, we acknowledge the work of other conference committee members, like the Organizing Chairs and Organizing Committee Members, Finance Chairs, Award Chair, Sponsorship Chairs, and Exhibition and Demonstration Chairs, Visa Chair, Publicity Chairs, and Women in ICPR Chairs, whose efforts made this event successful. We also thank our event manager Alpcord Network for their help.

We hope that all the participants found the technical program informative and enjoyed the sights, culture and cuisine of Kolkata.

October 2024

Umapada Pal
Josef Kittler
Anil Jain

Organization

General Chairs

Umapada Pal	Indian Statistical Institute, Kolkata, India
Josef Kittler	University of Surrey, UK
Anil Jain	Michigan State University, USA

Program Chairs

Apostolos Antonacopoulos	University of Salford, UK
Subhasis Chaudhuri	Indian Institute of Technology, Bombay, India
Rama Chellappa	Johns Hopkins University, USA
Cheng-Lin Liu	Institute of Automation, Chinese Academy of Sciences, China

Publication Chairs

Ananda S. Chowdhury	Jadavpur University, India
Wataru Ohyama	Tokyo Denki University, Japan

Competition Chairs

Richard Zanibbi	Rochester Institute of Technology, USA
Lianwen Jin	South China University of Technology, China
Laurence Likforman-Sulem	Télécom Paris, France

Workshop Chairs

P. Shivakumara	University of Salford, UK
Stephanie Schuckers	Clarkson University, USA
Jean-Marc Ogier	Université de la Rochelle, France
Prabir Bhattacharya	Concordia University, Canada

Tutorial Chairs

B. B. Chaudhuri	Indian Statistical Institute, Kolkata, India
Michael R. Jenkin	York University, Canada
Guoying Zhao	University of Oulu, Finland

Doctoral Consortium Chairs

Véronique Eglin	CNRS, France
Daniel P. Lopresti	Lehigh University, USA
Mayank Vatsa	Indian Institute of Technology, Jodhpur, India

Organizing Chairs

Saumik Bhattacharya	Indian Institute of Technology, Kharagpur, India
Palash Ghosal	Sikkim Manipal University, India

Organizing Committee

Santanu Phadikar	West Bengal University of Technology, India
SK Md Obaidullah	Aliah University, India
Sayantari Ghosh	National Institute of Technology Durgapur, India
Himadri Mukherjee	West Bengal State University, India
Nilamadhaba Tripathy	Clarivate Analytics, USA
Chayan Halder	West Bengal State University, India
Shibaprasad Sen	Techno Main Salt Lake, India

Finance Chairs

Kaushik Roy	West Bengal State University, India
Michael Blumenstein	University of Technology Sydney, Australia

Awards Committee Chair

Arpan Pal	Tata Consultancy Services, India

Sponsorship Chairs

P. J. Narayanan	Indian Institute of Technology, Hyderabad, India
Yasushi Yagi	Osaka University, Japan
Venu Govindaraju	University at Buffalo, USA
Alberto Bel Bimbo	Università di Firenze, Italy

Exhibition and Demonstration Chairs

Arjun Jain	FastCode AI, India
Agnimitra Biswas	National Institute of Technology, Silchar, India

International Liaison, Visa Chair

Balasubramanian Raman	Indian Institute of Technology, Roorkee, India

Publicity Chairs

Dipti Prasad Mukherjee	Indian Statistical Institute, Kolkata, India
Bob Fisher	University of Edinburgh, UK
Xiaojun Wu	Jiangnan University, China

Women in ICPR Chairs

Ingela Nystrom	Uppsala University, Sweden
Alexandra B. Albu	University of Victoria, Canada
Jing Dong	Institute of Automation, Chinese Academy of Sciences, China
Sarbani Palit	Indian Statistical Institute, Kolkata, India

Event Manager

Alpcord Network

Track Chairs – Artificial Intelligence, Machine Learning for Pattern Analysis

Larry O'Gorman	Nokia Bell Labs, USA
Dacheng Tao	University of Sydney, Australia
Petia Radeva	University of Barcelona, Spain
Susmita Mitra	Indian Statistical Institute, Kolkata, India
Jiliang Tang	Michigan State University, USA

Track Chairs – Computer and Robot Vision

C. V. Jawahar	International Institute of Information Technology (IIIT), Hyderabad, India
João Paulo Papa	São Paulo State University, Brazil
Maja Pantic	Imperial College London, UK
Gang Hua	Dolby Laboratories, USA
Junwei Han	Northwestern Polytechnical University, China

Track Chairs – Image, Speech, Signal and Video Processing

P. K. Biswas	Indian Institute of Technology, Kharagpur, India
Shang-Hong Lai	National Tsing Hua University, Taiwan
Hugo Jair Escalante	INAOE, CINVESTAV, Mexico
Sergio Escalera	Universitat de Barcelona, Spain
Prem Natarajan	University of Southern California, USA

Track Chairs – Biometrics and Human Computer Interaction

Richa Singh	Indian Institute of Technology, Jodhpur, India
Massimo Tistarelli	University of Sassari, Italy
Vishal Patel	Johns Hopkins University, USA
Wei-Shi Zheng	Sun Yat-sen University, China
Jian Wang	Snap, USA

Track Chairs – Document Analysis and Recognition

Xiang Bai	Huazhong University of Science and Technology, China
David Doermann	University at Buffalo, USA
Josep Llados	Universitat Autònoma de Barcelona, Spain
Mita Nasipuri	Jadavpur University, India

Track Chairs – Biomedical Imaging and Bioinformatics

Jayanta Mukhopadhyay	Indian Institute of Technology, Kharagpur, India
Xiaoyi Jiang	Universität Münster, Germany
Seong-Whan Lee	Korea University, Korea

Metareviewers (Conference Papers and Competition Papers)

Wael Abd-Almageed	University of Southern California, USA
Maya Aghaei	NHL Stenden University, Netherlands
Alireza Alaei	Southern Cross University, Australia
Rajagopalan N. Ambasamudram	Indian Institute of Technology, Madras, India
Suyash P. Awate	Indian Institute of Technology, Bombay, India
Inci M. Baytas	Bogazici University, Turkey
Aparna Bharati	Lehigh University, USA
Brojeshwar Bhowmick	Tata Consultancy Services, India
Jean-Christophe Burie	University of La Rochelle, France
Gustavo Carneiro	University of Surrey, UK
Chee Seng Chan	Universiti Malaya, Malaysia
Sumohana S. Channappayya	Indian Institute of Technology, Hyderabad, India
Dongdong Chen	Microsoft, USA
Shengyong Chen	Tianjin University of Technology, China
Jun Cheng	Institute for Infocomm Research, A*STAR, Singapore
Albert Clapés	University of Barcelona, Spain
Oscar Dalmau	Center for Research in Mathematics, Mexico

Tyler Derr	Vanderbilt University, USA
Abhinav Dhall	Indian Institute of Technology, Ropar, India
Bo Du	Wuhan University, China
Yuxuan Du	University of Sydney, Australia
Ayman S. El-Baz	University of Louisville, USA
Francisco Escolano	University of Alicante, Spain
Siamac Fazli	Nazarbayev University, Kazakhstan
Jianjiang Feng	Tsinghua University, China
Gernot A. Fink	TU Dortmund University, Germany
Alicia Fornes	CVC, Spain
Junbin Gao	University of Sydney, Australia
Yan Gao	Amazon, USA
Yongsheng Gao	Griffith University, Australia
Caren Han	University of Melbourne, Australia
Ran He	Institute of Automation, Chinese Academy of Sciences, China
Tin Kam Ho	IBM, USA
Di Huang	Beihang University, China
Kaizhu Huang	Duke Kunshan University, China
Donato Impedovo	University of Bari, Italy
Julio Jacques	University of Barcelona and Computer Vision Center, Spain
Lianwen Jin	South China University of Technology, China
Wei Jin	Emory University, USA
Danilo Samuel Jodas	São Paulo State University, Brazil
Manjunath V. Joshi	DA-IICT, India
Jayashree Kalpathy-Cramer	Massachusetts General Hospital, USA
Dimosthenis Karatzas	Computer Vision Centre, Spain
Hamid Karimi	Utah State University, USA
Baiying Lei	Shenzhen University, China
Guoqi Li	Chinese Academy of Sciences, and Peng Cheng Lab, China
Laurence Likforman-Sulem	Institut Polytechnique de Paris/Télécom Paris, France
Aishan Liu	Beihang University, China
Bo Liu	Bytedance, USA
Chen Liu	Clarkson University, USA
Cheng-Lin Liu	Institute of Automation, Chinese Academy of Sciences, China
Hongmin Liu	University of Science and Technology Beijing, China
Hui Liu	Michigan State University, USA

Jing Liu	Institute of Automation, Chinese Academy of Sciences, China
Li Liu	University of Oulu, Finland
Qingshan Liu	Nanjing University of Posts and Telecommunications, China
Adrian P. Lopez-Monroy	Centro de Investigacion en Matematicas AC, Mexico
Daniel P. Lopresti	Lehigh University, USA
Shijian Lu	Nanyang Technological University, Singapore
Yong Luo	Wuhan University, China
Andreas K. Maier	FAU Erlangen-Nuremberg, Germany
Davide Maltoni	University of Bologna, Italy
Hong Man	Stevens Institute of Technology, USA
Lingtong Min	Northwestern Polytechnical University, China
Paolo Napoletano	University of Milano-Bicocca, Italy
Kamal Nasrollahi	Milestone Systems, Aalborg University, Denmark
Marcos Ortega	University of A Coruña, Spain
Shivakumara Palaiahnakote	University of Salford, UK
P. Jonathon Phillips	NIST, USA
Filiberto Pla	University Jaume I, Spain
Ajit Rajwade	Indian Institute of Technology, Bombay, India
Shanmuganathan Raman	Indian Institute of Technology, Gandhinagar, India
Imran Razzak	UNSW, Australia
Beatriz Remeseiro	University of Oviedo, Spain
Gustavo Rohde	University of Virginia, USA
Partha Pratim Roy	Indian Institute of Technology, Roorkee, India
Sanjoy K. Saha	Jadavpur University, India
Joan Andreu Sánchez	Universitat Politècnica de València, Spain
Claudio F. Santos	UFSCar, Brazil
Shin'ichi Satoh	National Institute of Informatics, Japan
Stephanie Schuckers	Clarkson University, USA
Srirangaraj Setlur	University at Buffalo, SUNY, USA
Debdoot Sheet	Indian Institute of Technology, Kharagpur, India
Jun Shen	University of Wollongong, Australia
Li Shen	JD Explore Academy, China
Chen Shengyong	Zhejiang University of Technology and Tianjin University of Technology, China
Andy Song	RMIT University, Australia
Akihiro Sugimoto	National Institute of Informatics, Japan
Qianru Sun	Singapore Management University, Singapore
Arijit Sur	Indian Institute of Technology, Guwahati, India
Estefania Talavera	University of Twente, Netherlands

Wei Tang	University of Illinois at Chicago, USA
Joao M. Tavares	Universidade do Porto, Portugal
Jun Wan	NLPR, CASIA, China
Le Wang	Xi'an Jiaotong University, China
Lei Wang	Australian National University, Australia
Xiaoyang Wang	Tencent AI Lab, USA
Xinggang Wang	Huazhong University of Science and Technology, China
Xiao-Jun Wu	Jiangnan University, China
Yiding Yang	Bytedance, China
Xiwen Yao	Northwestern Polytechnical University, China
Xu-Cheng Yin	University of Science and Technology Beijing, China
Baosheng Yu	University of Sydney, Australia
Shiqi Yu	Southern University of Science and Technology, China
Xin Yuan	Westlake University, China
Yibing Zhan	JD Explore Academy, China
Jing Zhang	University of Sydney, Australia
Lefei Zhang	Wuhan University, China
Min-Ling Zhang	Southeast University, China
Wenbin Zhang	Florida International University, USA
Jiahuan Zhou	Peking University, China
Sanping Zhou	Xi'an Jiaotong University, China
Tianyi Zhou	University of Maryland, USA
Lei Zhu	Shandong Normal University, China
Pengfei Zhu	Tianjin University, China
Wangmeng Zuo	Harbin Institute of Technology, China

Reviewers (Competition Papers)

Liangcai Gao	Da-Han Wang
Mingxin Huang	Yang Xue
Lei Kang	Wentao Yang
Wenhui Liao	Jiaxin Zhang
Yuliang Liu	Yiwu Zhong
Yongxin Shi	

Reviewers (Conference Papers)

Aakanksha Aakanksha
Aayush Singla
Abdul Muqeet
Abhay Yadav
Abhijeet Vijay Nandedkar
Abhimanyu Sahu
Abhinav Rajvanshi
Abhisek Ray
Abhishek Shrivastava
Abhra Chaudhuri
Aditi Roy
Adriano Simonetto
Adrien Maglo
Ahmed Abdulkadir
Ahmed Boudissa
Ahmed Hamdi
Ahmed Rida Sekkat
Ahmed Sharafeldeen
Aiman Farooq
Aishwarya Venkataramanan
Ajay Kumar
Ajay Kumar Reddy Poreddy
Ajita Rattani
Ajoy Mondal
Akbar K.
Akbar Telikani
Akshay Agarwal
Akshit Jindal
Al Zadid Sultan Bin Habib
Albert Clapés
Alceu Britto
Alejandro Peña
Alessandro Ortis
Alessia Auriemma Citarella
Alexandre Stenger
Alexandros Sopasakis
Alexia Toumpa
Ali Khan
Alik Pramanick
Alireza Alaei
Alper Yilmaz
Aman Verma
Amit Bhardwaj
Amit More
Amit Nandedkar
Amitava Chatterjee
Amos L. Abbott
Amrita Mohan
Anand Mishra
Ananda S. Chowdhury
Anastasia Zakharova
Anastasios L. Kesidis
Andras Horvath
Andre Gustavo Hochuli
André P. Kelm
Andre Wyzykowski
Andrea Bottino
Andrea Lagorio
Andrea Torsello
Andreas Fischer
Andreas K. Maier
Andreu Girbau Xalabarder
Andrew Beng Jin Teoh
Andrew Shin
Andy J. Ma
Aneesh S. Chivukula
Ángela Casado-García
Anh Quoc Nguyen
Anindya Sen
Anirban Saha
Anjali Gautam
Ankan Bhattacharyya
Ankit Jha
Anna Scius-Bertrand
Annalisa Franco
Antoine Doucet
Antonino Staiano
Antonio Fernández
Antonio Parziale
Anu Singha
Anustup Choudhury
Anwesan Pal
Anwesha Sengupta
Archisman Adhikary
Arjan Kuijper
Arnab Kumar Das

Arnav Bhavsar
Arnav Varma
Arpita Dutta
Arshad Jamal
Artur Jordao
Arunkumar Chinnaswamy
Aryan Jadon
Aryaz Baradarani
Ashima Anand
Ashis Dhara
Ashish Phophalia
Ashok K. Bhateja
Ashutosh Vaish
Ashwani Kumar
Asifuzzaman Lasker
Atefeh Khoshkhahtinat
Athira Nambiar
Attilio Fiandrotti
Avandra S. Hemachandra
Avik Hati
Avinash Sharma
B. H. Shekar
B. Uma Shankar
Bala Krishna Thunakala
Balaji Tk
Balázs Pálffy
Banafsheh Adami
Bang-Dang Pham
Baochang Zhang
Baodi Liu
Bashirul Azam Biswas
Beiduo Chen
Benedikt Kottler
Beomseok Oh
Berkay Aydin
Berlin S. Shaheema
Bertrand Kerautret
Bettina Finzel
Bhavana Singh
Bibhas C. Dhara
Bilge Gunsel
Bin Chen
Bin Li
Bin Liu
Bin Yao
Bin-Bin Jia
Binbin Yong
Bindita Chaudhuri
Bindu Madhavi Tummala
Binh M. Le
Bi-Ru Dai
Bo Huang
Bo Jiang
Bob Zhang
Bowen Liu
Bowen Zhang
Boyang Zhang
Boyu Diao
Boyun Li
Brian M. Sadler
Bruce A. Maxwell
Bryan Bo Cao
Buddhika L. Semage
Bushra Jalil
Byeong-Seok Shin
Byung-Gyu Kim
Caihua Liu
Cairong Zhao
Camille Kurtz
Carlos A. Caetano
Carlos D. Martã-Nez-Hinarejos
Ce Wang
Cevahir Cigla
Chakravarthy Bhagvati
Chandrakanth Vipparla
Changchun Zhang
Changde Du
Changkun Ye
Changxu Cheng
Chao Fan
Chao Guo
Chao Qu
Chao Wen
Chayan Halder
Che-Jui Chang
Chen Feng
Chenan Wang
Cheng Yu
Chenghao Qian
Cheng-Lin Liu

Chengxu Liu
Chenru Jiang
Chensheng Peng
Chetan Ralekar
Chih-Wei Lin
Chih-Yi Chiu
Chinmay Sahu
Chintan Patel
Chintan Shah
Chiranjoy Chattopadhyay
Chong Wang
Choudhary Shyam Prakash
Christophe Charrier
Christos Smailis
Chuanwei Zhou
Chun-Ming Tsai
Chunpeng Wang
Ciro Russo
Claudio De Stefano
Claudio F. Santos
Claudio Marrocco
Connor Levenson
Constantine Dovrolis
Constantine Kotropoulos
Dai Shi
Dakshina Ranjan Kisku
Dan Anitei
Dandan Zhu
Daniela Pamplona
Danli Wang
Danqing Huang
Daoan Zhang
Daqing Hou
David A. Clausi
David Freire Obregon
David Münch
David Pujol Perich
Davide Marelli
De Zhang
Debalina Barik
Debapriya Roy (Kundu)
Debashis Das
Debashis Das Chakladar
Debi Prosad Dogra
Debraj D. Basu
Decheng Liu
Deen Dayal Mohan
Deep A. Patel
Deepak Kumar
Dengpan Liu
Denis Coquenet
Désiré Sidibé
Devesh Walawalkar
Dewan Md. Farid
Di Ming
Di Qiu
Di Yuan
Dian Jia
Dianmo Sheng
Diego Thomas
Diganta Saha
Dimitri Bulatov
Dimpy Varshni
Dingcheng Yang
Dipanjan Das
Dipanjyoti Paul
Divya Biligere Shivanna
Divya Saxena
Divya Sharma
Dmitrii Matveichev
Dmitry Minskiy
Dmitry V. Sorokin
Dong Zhang
Donghua Wang
Donglin Zhang
Dongming Wu
Dongqiangzi Ye
Dongqing Zou
Dongrui Liu
Dongyang Zhang
Dongzhan Zhou
Douglas Rodrigues
Duarte Folgado
Duc Minh Vo
Duoxuan Pei
Durai Arun Pannir Selvam
Durga Bhavani S.
Eckart Michaelsen
Elena Goyanes
Élodie Puybareau

Emanuele Vivoli
Emna Ghorbel
Enrique Naredo
Enyu Cai
Eric Patterson
Ernest Valveny
Eva Blanco-Mallo
Eva Breznik
Evangelos Sartinas
Fabio Solari
Fabiola De Marco
Fan Wang
Fangda Li
Fangyuan Lei
Fangzhou Lin
Fangzhou Luo
Fares Bougourzi
Farman Ali
Fatiha Mokdad
Fei Shen
Fei Teng
Fei Zhu
Feiyan Hu
Felipe Gomes Oliveira
Feng Li
Fengbei Liu
Fenghua Zhu
Fillipe D. M. De Souza
Flavio Piccoli
Flavio Prieto
Florian Kleber
Francesc Serratosa
Francesco Bianconi
Francesco Castro
Francesco Ponzio
Francisco Javier Hernández López
Frédéric Rayar
Furkan Osman Kar
Fushuo Huo
Fuxiao Liu
Fu-Zhao Ou
Gabriel Turinici
Gabrielle Flood
Gajjala Viswanatha Reddy
Gaku Nakano
Galal Binamakhashen
Ganesh Krishnasamy
Gang Pan
Gangyan Zeng
Gani Rahmon
Gaurav Harit
Gennaro Vessio
Genoveffa Tortora
George Azzopardi
Gerard Ortega
Gerardo E. Altamirano-Gomez
Gernot A. Fink
Gibran Benitez-Garcia
Gil Ben-Artzi
Gilbert Lim
Giorgia Minello
Giorgio Fumera
Giovanna Castellano
Giovanni Puglisi
Giulia Orrù
Giuliana Ramella
Gökçe Uludoğan
Gopi Ramena
Gorthi Rama Krishna Sai Subrahmanyam
Gourav Datta
Gowri Srinivasa
Gozde Sahin
Gregory Randall
Guanjie Huang
Guanjun Li
Guanwen Zhang
Guanyu Xu
Guanyu Yang
Guanzhou Ke
Guhnoo Yun
Guido Borghi
Guilherme Brandão Martins
Guillaume Caron
Guillaume Tochon
Guocai Du
Guohao Li
Guoqiang Zhong
Guorong Li
Guotao Li
Gurman Gill

Haechang Lee
Haichao Zhang
Haidong Xie
Haifeng Zhao
Haimei Zhao
Hainan Cui
Haixia Wang
Haiyan Guo
Hakime Ozturk
Hamid Kazemi
Han Gao
Hang Zou
Hanjia Lyu
Hanjoo Cho
Hanqing Zhao
Hanyuan Liu
Hanzhou Wu
Hao Li
Hao Meng
Hao Sun
Hao Wang
Hao Xing
Hao Zhao
Haoan Feng
Haodi Feng
Haofeng Li
Haoji Hu
Haojie Hao
Haojun Ai
Haopeng Zhang
Haoran Li
Haoran Wang
Haorui Ji
Haoxiang Ma
Haoyu Chen
Haoyue Shi
Harald Koestler
Harbinder Singh
Harris V. Georgiou
Hasan F. Ates
Hasan S. M. Al-Khaffaf
Hatef Otroshi Shahreza
Hebeizi Li
Heng Zhang
Hengli Wang
Hengyue Liu
Hertog Nugroho
Hieyong Jeong
Himadri Mukherjee
Hoai Ngo
Hoda Mohaghegh
Hong Liu
Hong Man
Hongcheng Wang
Hongjian Zhan
Hongxi Wei
Hongyu Hu
Hoseong Kim
Hossein Ebrahimnezhad
Hossein Malekmohamadi
Hrishav Bakul Barua
Hsueh-Yi Sean Lin
Hua Wei
Huafeng Li
Huali Xu
Huaming Chen
Huan Wang
Huang Chen
Huanran Chen
Hua-Wen Chang
Huawen Liu
Huayi Zhan
Hugo Jair Escalante
Hui Chen
Hui Li
Huichen Yang
Huiqiang Jiang
Huiyuan Yang
Huizi Yu
Hung T. Nguyen
Hyeongyu Kim
Hyeonjeong Park
Hyeonjun Lee
Hymalai Bello
Hyung-Gun Chi
Hyunsoo Kim
I-Chen Lin
Ik Hyun Lee
Ilan Shimshoni
Imad Eddine Toubal

Imran Sarker
Inderjot Singh Saggu
Indrani Mukherjee
Indranil Sur
Ines Rieger
Ioannis Pierros
Irina Rabaev
Ivan V. Medri
J. Rafid Siddiqui
Jacek Komorowski
Jacopo Bonato
Jacson Rodrigues Correia-Silva
Jaekoo Lee
Jaime Cardoso
Jakob Gawlikowski
Jakub Nalepa
James L. Wayman
Jan Čech
Jangho Lee
Jani Boutellier
Javier Gurrola-Ramos
Javier Lorenzo-Navarro
Jayasree Saha
Jean Lee
Jean Paul Barddal
Jean-Bernard Hayet
Jean-Philippe G. Tarel
Jean-Yves Ramel
Jenny Benois-Pineau
Jens Bayer
Jerin Geo James
Jesús Miguel García-Gorrostieta
Jia Qu
Jiahong Chen
Jiaji Wang
Jian Hou
Jian Liang
Jian Xu
Jian Zhu
Jianfeng Lu
Jianfeng Ren
Jiangfan Liu
Jianguo Wang
Jiangyan Yi
Jiangyong Duan
Jianhua Yang
Jianhua Zhang
Jianhui Chen
Jianjia Wang
Jianli Xiao
Jianqiang Xiao
Jianwu Wang
Jianxin Zhang
Jianxiong Gao
Jianxiong Zhou
Jianyu Wang
Jianzhong Wang
Jiaru Zhang
Jiashu Liao
Jiaxin Chen
Jiaxin Lu
Jiaxing Ye
Jiaxuan Chen
Jiaxuan Li
Jiayi He
Jiayin Lin
Jie Ou
Jiehua Zhang
Jiejie Zhao
Jignesh S. Bhatt
Jin Gao
Jin Hou
Jin Hu
Jin Shang
Jing Tian
Jing Yu Chen
Jingfeng Yao
Jinglun Feng
Jingtong Yue
Jingwei Guo
Jingwen Xu
Jingyuan Xia
Jingzhe Ma
Jinhong Wang
Jinjia Wang
Jinlai Zhang
Jinlong Fan
Jinming Su
Jinrong He
Jintao Huang

Jinwoo Ahn
Jinwoo Choi
Jinyang Liu
Jinyu Tian
Jionghao Lin
Jiuding Duan
Jiwei Shen
Jiyan Pan
Jiyoun Kim
João Papa
Johan Debayle
John Atanbori
John Wilson
John Zhang
Jónathan Heras
Joohi Chauhan
Jorge Calvo-Zaragoza
Jorge Figueroa
Jorma Laaksonen
José Joaquim De Moura Ramos
Jose Vicent
Joseph Damilola Akinyemi
Josiane Zerubia
Juan Wen
Judit Szücs
Juepeng Zheng
Juha Roning
Jumana H. Alsubhi
Jun Cheng
Jun Ni
Jun Wan
Junghyun Cho
Junjie Liang
Junjie Ye
Junlin Hu
Juntong Ni
Junxin Lu
Junxuan Li
Junyaup Kim
Junyeong Kim
Jürgen Seiler
Jushang Qiu
Juyang Weng
Jyostna Devi Bodapati
Jyoti Singh Kirar
Kai Jiang
Kaiqiang Song
Kalidas Yeturu
Kalle Åström
Kamalakar Vijay Thakare
Kang Gu
Kang Ma
Kanji Tanaka
Karthik Seemakurthy
Kaushik Roy
Kavisha Jayathunge
Kazuki Uehara
Ke Shi
Keigo Kimura
Keiji Yanai
Kelton A. P. Costa
Kenneth Camilleri
Kenny Davila
Ketan Atul Bapat
Ketan Kotwal
Kevin Desai
Keyu Long
Khadiga Mohamed Ali
Khakon Das
Khan Muhammad
Kilho Son
Kim-Ngan Nguyen
Kishan Kc
Kishor P. Upla
Klaas Dijkstra
Komal Bharti
Konstantinos Triaridis
Kostas Ioannidis
Koyel Ghosh
Kripabandhu Ghosh
Krishnendu Ghosh
Kshitij S. Jadhav
Kuan Yan
Kun Ding
Kun Xia
Kun Zeng
Kunal Banerjee
Kunal Biswas
Kunchi Li
Kurban Ubul

Lahiru N. Wijayasingha
Laines Schmalwasser
Lakshman Mahto
Lala Shakti Swarup Ray
Lale Akarun
Lan Yan
Lawrence Amadi
Lee Kang Il
Lei Fan
Lei Shi
Lei Wang
Leonardo Rossi
Lequan Lin
Levente Tamas
Li Bing
Li Li
Li Ma
Li Song
Lia Morra
Liang Xie
Liang Zhao
Lianwen Jin
Libing Zeng
Lidia Sánchez-González
Lidong Zeng
Lijun Li
Likang Wang
Lili Zhao
Lin Chen
Lin Huang
Linfei Wang
Ling Lo
Lingchen Meng
Lingheng Meng
Lingxiao Li
Lingzhong Fan
Liqi Yan
Liqiang Jing
Lisa Gutzeit
Liu Ziyi
Liushuai Shi
Liviu-Daniel Stefan
Liyuan Ma
Liyun Zhu
Lizuo Jin
Longteng Guo
Lorena Álvarez Rodríguez
Lorenzo Putzu
Lu Leng
Lu Pang
Lu Wang
Luan Pham
Luc Brun
Luca Guarnera
Luca Piano
Lucas Alexandre Ramos
Lucas Goncalves
Lucas M. Gago
Luigi Celona
Luis C. S. Afonso
Luis Gerardo De La Fraga
Luis S. Luevano
Luis Teixeira
Lunke Fei
M. Hassaballah
Maddimsetti Srinivas
Mahendran N.
Mahesh Mohan M. R.
Maiko Lie
Mainak Singha
Makoto Hirose
Malay Bhattacharyya
Mamadou Dian Bah
Man Yao
Manali J. Patel
Manav Prabhakar
Manikandan V. M.
Manish Bhatt
Manjunath Shantharamu
Manuel Curado
Manuel Günther
Manuel Marques
Marc A. Kastner
Marc Chaumont
Marc Cheong
Marc Lalonde
Marco Cotogni
Marcos C. Santana
Mario Molinara
Mariofanna Milanova

Markus Bauer
Marlon Becker
Mårten Wadenbäck
Martin G. Ljungqvist
Martin Kampel
Martina Pastorino
Marwan Torki
Masashi Nishiyama
Masayuki Tanaka
Massimo O. Spata
Matteo Ferrara
Matthew D. Dawkins
Matthew Gadd
Matthew S. Watson
Maura Pintor
Max Ehrlich
Maxim Popov
Mayukh Das
Md Baharul Islam
Md Sajid
Meghna Kapoor
Meghna P. Ayyar
Mei Wang
Meiqi Wu
Melissa L. Tijink
Meng Li
Meng Liu
Meng-Luen Wu
Mengnan Liu
Mengxi China Guo
Mengya Han
Michaël Clément
Michal Kawulok
Mickael Coustaty
Miguel Domingo
Milind G. Padalkar
Ming Liu
Ming Ma
Mingchen Feng
Mingde Yao
Minghao Li
Mingjie Sun
Ming-Kuang Daniel Wu
Mingle Xu
Mingyong Li
Mingyuan Jiu
Minh P. Nguyen
Minh Q. Tran
Minheng Ni
Minsu Kim
Minyi Zhao
Mirko Paolo Barbato
Mo Zhou
Modesto Castrillón-Santana
Mohamed Amine Mezghich
Mohamed Dahmane
Mohamed Elsharkawy
Mohamed Yousuf
Mohammad Hashemi
Mohammad Khalooei
Mohammad Khateri
Mohammad Mahdi Dehshibi
Mohammad Sadil Khan
Mohammed Mahmoud
Moises Diaz
Monalisha Mahapatra
Monidipa Das
Mostafa Kamali Tabrizi
Mridul Ghosh
Mrinal Kanti Bhowmik
Muchao Ye
Mugalodi Ramesha Rakesh
Muhammad Rameez Ur Rahman
Muhammad Suhaib Kanroo
Muming Zhao
Munender Varshney
Munsif Ali
Na Lv
Nader Karimi
Nagabhushan Somraj
Nakkwan Choi
Nakul Agarwal
Nan Pu
Nan Zhou
Nancy Mehta
Nand Kumar Yadav
Nandakishor Nandakishor
Nandyala Hemachandra
Nanfeng Jiang
Narayan Hegde

Narayan Ji Mishra
Narayan Vetrekar
Narendra D. Londhe
Nathalie Girard
Nati Ofir
Naval Kishore Mehta
Nazmul Shahadat
Neeti Narayan
Neha Bhargava
Nemanja Djuric
Newlin Shebiah R.
Ngo Ba Hung
Nhat-Tan Bui
Niaz Ahmad
Nick Theisen
Nicolas Passat
Nicolas Ragot
Nicolas Sidere
Nikolaos Mitianoudis
Nikolas Ebert
Nilah Ravi Nair
Nilesh A. Ahuja
Nilkanta Sahu
Nils Murrugarra-Llerena
Nina S. T. Hirata
Ninad Aithal
Ning Xu
Ningzhi Wang
Niraj Kumar
Nirmal S. Punjabi
Nisha Varghese
Norio Tagawa
Obaidullah Md Sk
Oguzhan Ulucan
Olfa Mechi
Oliver Tüselmann
Orazio Pontorno
Oriol Ramos Terrades
Osman Akin
Ouadi Beya
Ozge Mercanoglu Sincan
Pabitra Mitra
Padmanabha Reddy Y. C. A.
Palaash Agrawal
Palaiahnakote Shivakumara
Palash Ghosal
Pallav Dutta
Paolo Rota
Paramanand Chandramouli
Paria Mehrani
Parth Agrawal
Partha Basuchowdhuri
Patrick Horain
Pavan Kumar
Pavan Kumar Anasosalu Vasu
Pedro Castro
Peipei Li
Peipei Yang
Peisong Shen
Peiyu Li
Peng Li
Pengfei He
Pengrui Quan
Pengxin Zeng
Pengyu Yan
Peter Eisert
Petra Gomez-Krämer
Pierrick Bruneau
Ping Cao
Pingping Zhang
Pintu Kumar
Pooja Kumari
Pooja Sahani
Prabhu Prasad Dev
Pradeep Kumar
Pradeep Singh
Pranjal Sahu
Prasun Roy
Prateek Keserwani
Prateek Mittal
Praveen Kumar Chandaliya
Praveen Tirupattur
Pravin Nair
Preeti Gopal
Preety Singh
Prem Shanker Yadav
Prerana Mukherjee
Prerna A. Mishra
Prianka Dey
Priyanka Mudgal

Qc Kha Ng
Qi Li
Qi Ming
Qi Wang
Qi Zuo
Qian Li
Qiang Gan
Qiang He
Qiang Wu
Qiangqiang Zhou
Qianli Zhao
Qiansen Hong
Qiao Wang
Qidong Huang
Qihua Dong
Qin Yuke
Qing Guo
Qingbei Guo
Qingchao Zhang
Qingjie Liu
Qinhong Yang
Qiushi Shi
Qixiang Chen
Quan Gan
Quanlong Guan
Rachit Chhaya
Radu Tudor Ionescu
Rafal Zdunek
Raghavendra Ramachandra
Rahimul I. Mazumdar
Rahul Kumar Ray
Rajib Dutta
Rajib Ghosh
Rakesh Kumar
Rakesh Paul
Rama Chellappa
Rami O. Skaik
Ramon Aranda
Ran Wei
Ranga Raju Vatsavai
Ranganath Krishnan
Rasha Friji
Rashmi S.
Razaib Tariq
Rémi Giraud
René Schuster
Renlong Hang
Renrong Shao
Renu Sharma
Reza Sadeghian
Richard Zanibbi
Rimon Elias
Rishabh Shukla
Rita Delussu
Riya Verma
Robert J. Ravier
Robert Sablatnig
Robin Strand
Rocco Pietrini
Rocio Diaz Martin
Rocio Gonzalez-Diaz
Rohit Venkata Sai Dulam
Romain Giot
Romi Banerjee
Ru Wang
Ruben Machucho
Ruddy Théodose
Ruggero Pintus
Rui Deng
Rui P. Paiva
Rui Zhao
Ruifan Li
Ruigang Fu
Ruikun Li
Ruirui Li
Ruixiang Jiang
Ruowei Jiang
Rushi Lan
Rustam Zhumagambetov
S. Amutha
S. Divakar Bhat
Sagar Goyal
Sahar Siddiqui
Sahbi Bahroun
Sai Karthikeya Vemuri
Saibal Dutta
Saihui Hou
Sajad Ahmad Rather
Saksham Aggarwal
Sakthi U.

Salimeh Sekeh
Samar Bouazizi
Samia Boukir
Samir F. Harb
Samit Biswas
Samrat Mukhopadhyay
Samriddha Sanyal
Sandika Biswas
Sandip Purnapatra
Sanghyun Jo
Sangwoo Cho
Sanjay Kumar
Sankaran Iyer
Sanket Biswas
Santanu Roy
Santosh D. Pandure
Santosh Ku Behera
Santosh Nanabhau Palaskar
Santosh Prakash Chouhan
Sarah S. Alotaibi
Sasanka Katreddi
Sathyanarayanan N. Aakur
Saurabh Yadav
Sayan Rakshit
Scott McCloskey
Sebastian Bunda
Sejuti Rahman
Selim Aksoy
Sen Wang
Seraj A. Mostafa
Shanmuganathan Raman
Shao-Yuan Lo
Shaoyuan Xu
Sharia Arfin Tanim
Shehreen Azad
Sheng Wan
Shengdong Zhang
Shengwei Qin
Shenyuan Gao
Sherry X. Chen
Shibaprasad Sen
Shigeaki Namiki
Shiguang Liu
Shijie Ma
Shikun Li
Shinichiro Omachi
Shirley David
Shishir Shah
Shiv Ram Dubey
Shiva Baghel
Shivanand S. Gornale
Shogo Sato
Shotaro Miwa
Shreya Ghosh
Shreya Goyal
Shuai Su
Shuai Wang
Shuai Zheng
Shuaifeng Zhi
Shuang Qiu
Shuhei Tarashima
Shujing Lyu
Shuliang Wang
Shun Zhang
Shunming Li
Shunxin Wang
Shuping Zhao
Shuquan Ye
Shuwei Huo
Shuyue Lan
Shyi-Chyi Cheng
Si Chen
Siddarth Ravichandran
Sihan Chen
Siladittya Manna
Silambarasan Elkana Ebinazer
Simon Benaïchouche
Simon S. Woo
Simone Caldarella
Simone Milani
Simone Zini
Sina Lotfian
Sitao Luan
Sivaselvan B.
Siwei Li
Siwei Wang
Siwen Luo
Siyu Chen
Sk Aziz Ali
Sk Md Obaidullah

Sneha Shukla
Snehasis Banerjee
Snehasis Mukherjee
Snigdha Sen
Sofia Casarin
Soheila Farokhi
Soma Bandyopadhyay
Son Minh Nguyen
Son Xuan Ha
Sonal Kumar
Sonam Gupta
Sonam Nahar
Song Ouyang
Sotiris Kotsiantis
Souhaila Djaffal
Soumen Biswas
Soumen Sinha
Soumitri Chattopadhyay
Souvik Sengupta
Spiros Kostopoulos
Sreeraj Ramachandran
Sreya Banerjee
Srikanta Pal
Srinivas Arukonda
Stephane A. Guinard
Su O. Ruan
Subhadip Basu
Subhajit Paul
Subhankar Ghosh
Subhankar Mishra
Subhankar Roy
Subhash Chandra Pal
Subhayu Ghosh
Sudip Das
Sudipta Banerjee
Suhas Pillai
Sujit Das
Sukalpa Chanda
Sukhendu Das
Suklav Ghosh
Suman K. Ghosh
Suman Samui
Sumit Mishra
Sungho Suh
Sunny Gupta
Suraj Kumar Pandey
Surendrabikram Thapa
Suresh Sundaram
Sushil Bhattacharjee
Susmita Ghosh
Swakkhar Shatabda
Syed Ms Islam
Syed Tousiful Haque
Taegyeong Lee
Taihui Li
Takashi Shibata
Takeshi Oishi
Talha Ahmad Siddiqui
Tanguy Gernot
Tangwen Qian
Tanima Bhowmik
Tanpia Tasnim
Tao Dai
Tao Hu
Tao Sun
Taoran Yi
Tapan Shah
Taveena Lotey
Teng Huang
Tengqi Ye
Teresa Alarcon
Tetsuji Ogawa
Thanh Phuong Nguyen
Thanh Tuan Nguyen
Thattapon Surasak
Thibault Napolãon
Thierry Bouwmans
Thinh Truong Huynh Nguyen
Thomas De Min
Thomas E. K. Zielke
Thomas Swearingen
Tianatahina Jimmy Francky Randrianasoa
Tianheng Cheng
Tianjiao He
Tianyi Wei
Tianyuan Zhang
Tianyue Zheng
Tiecheng Song
Tilottama Goswami
Tim Büchner

Tim H. Langer
Tim Raven
Tingkai Liu
Tingting Yao
Tobias Meisen
Toby P. Breckon
Tong Chen
Tonghua Su
Tran Tuan Anh
Tri-Cong Pham
Trishna Saikia
Trung Quang Truong
Tuan T. Nguyen
Tuan Vo Van
Tushar Shinde
Ujjwal Karn
Ukrit Watchareeruetai
Uma Mudenagudi
Umarani Jayaraman
V. S. Malemath
Vallidevi Krishnamurthy
Ved Prakash
Venkata Krishna Kishore Kolli
Venkata R. Vavilthota
Venkatesh Thirugnana Sambandham
Verónica Maria Vasconcelos
Véronique Ve Eglin
Víctor E. Alonso-Pérez
Vinay Palakkode
Vinayak S. Nageli
Vincent J. Whannou De Dravo
Vincenzo Conti
Vincenzo Gattulli
Vineet Padmanabhan
Vishakha Pareek
Viswanath Gopalakrishnan
Vivek Singh Baghel
Vivekraj K.
Vladimir V. Arlazarov
Vu-Hoang Tran
W. Sylvia Lilly Jebarani
Wachirawit Ponghiran
Wafa Khlif
Wang An-Zhi
Wanli Xue
Wataru Ohyama
Wee Kheng Leow
Wei Chen
Wei Cheng
Wei Hua
Wei Lu
Wei Pan
Wei Tian
Wei Wang
Wei Wei
Wei Zhou
Weidi Liu
Weidong Yang
Weijun Tan
Weimin Lyu
Weinan Guan
Weining Wang
Weiqiang Wang
Weiwei Guo
Weixia Zhang
Wei-Xuan Bao
Weizhong Jiang
Wen Xie
Wenbin Qian
Wenbin Tian
Wenbin Wang
Wenbo Zheng
Wenhan Luo
Wenhao Wang
Wen-Hung Liao
Wenjie Li
Wenkui Yang
Wenwen Si
Wenwen Yu
Wenwen Zhang
Wenwu Yang
Wenxi Li
Wenxi Yue
Wenxue Cui
Wenzhuo Liu
Widhiyo Sudiyono
Willem Dijkstra
Wolfgang Fuhl
Xi Zhang
Xia Yuan

Xianda Zhang
Xiang Zhang
Xiangdong Su
Xiang-Ru Yu
Xiangtai Li
Xiangyu Xu
Xiao Guo
Xiao Hu
Xiao Wu
Xiao Yang
Xiaofeng Zhang
Xiaogang Du
Xiaoguang Zhao
Xiaoheng Jiang
Xiaohong Zhang
Xiaohua Huang
Xiaohua Li
Xiao-Hui Li
Xiaolong Sun
Xiaosong Li
Xiaotian Li
Xiaoting Wu
Xiaotong Luo
Xiaoyan Li
Xiaoyang Kang
Xiaoyi Dong
Xin Guo
Xin Lin
Xin Ma
Xinchi Zhou
Xingguang Zhang
Xingjian Leng
Xingpeng Zhang
Xingzheng Lyu
Xinjian Huang
Xinqi Fan
Xinqi Liu
Xinqiao Zhang
Xinrui Cui
Xizhan Gao
Xu Cao
Xu Ouyang
Xu Zhao
Xuan Shen
Xuan Zhou
Xuchen Li
Xuejing Lei
Xuelu Feng
Xueting Liu
Xuewei Li
Xueyi X. Wang
Xugong Qin
Xu-Qian Fan
Xuxu Liu
Xu-Yao Zhang
Yan Huang
Yan Li
Yan Wang
Yan Xia
Yan Zhuang
Yanan Li
Yanan Zhang
Yang Hou
Yang Jiao
Yang Liping
Yang Liu
Yang Qian
Yang Yang
Yang Zhao
Yangbin Chen
Yangfan Zhou
Yanhui Guo
Yanjia Huang
Yanjun Zhu
Yanming Zhang
Yanqing Shen
Yaoming Cai
Yaoxin Zhuo
Yaoyan Zheng
Yaping Zhang
Yaqian Liang
Yarong Feng
Yasmina Benmabrouk
Yasufumi Sakai
Yasutomo Kawanishi
Yazeed Alzahrani
Ye Du
Ye Duan
Yechao Zhang
Yeong-Jun Cho

Yi Huo
Yi Shi
Yi Yu
Yi Zhang
Yibo Liu
Yibo Wang
Yi-Chieh Wu
Yifan Chen
Yifei Huang
Yihao Ding
Yijie Tang
Yikun Bai
Yimin Wen
Yinan Yang
Yin-Dong Zheng
Yinfeng Yu
Ying Dai
Yingbo Li
Yiqiao Li
Yiqing Huang
Yisheng Lv
Yisong Xiao
Yite Wang
Yizhe Li
Yong Wang
Yonghao Dong
Yong-Hyuk Moon
Yongjie Li
Yongqian Li
Yongqiang Mao
Yongxu Liu
Yongyu Wang
Yongzhi Li
Youngha Hwang
Yousri Kessentini
Yu Wang
Yu Zhou
Yuan Tian
Yuan Zhang
Yuanbo Wen
Yuanxin Wang
Yubin Hu
Yubo Huang
Yuchen Ren
Yucheng Xing
Yuchong Yao
Yuecong Min
Yuewei Yang
Yufei Zhang
Yufeng Yin
Yugen Yi
Yuhang Ming
Yujia Zhang
Yujun Ma
Yukiko Kenmochi
Yun Hoyeoung
Yun Liu
Yunhe Feng
Yunxiao Shi
Yuru Wang
Yushun Tang
Yusuf Osmanlioglu
Yusuke Fujita
Yuta Nakashima
Yuwei Yang
Yuwu Lu
Yuxi Liu
Yuya Obinata
Yuyao Yan
Yuzhi Guo
Zaipeng Xie
Zander W. Blasingame
Zedong Wang
Zeliang Zhang
Zexin Ji
Zhanxiang Feng
Zhaofei Yu
Zhe Chen
Zhe Cui
Zhe Liu
Zhe Wang
Zhekun Luo
Zhen Yang
Zhenbo Li
Zhenchun Lei
Zhenfei Zhang
Zheng Liu
Zheng Wang
Zhengming Yu
Zhengyin Du

Zhengyun Cheng
Zhenshen Qu
Zhenwei Shi
Zhenzhong Kuang
Zhi Cai
Zhi Chen
Zhibo Chu
Zhicun Yin
Zhida Huang
Zhida Zhang
Zhifan Gao
Zhihang Ren
Zhihang Yuan
Zhihao Wang
Zhihua Xie
Zhihui Wang
Zhikang Zhang
Zhiming Zou
Zhiqi Shao
Zhiwei Dong
Zhiwei Qi
Zhixiang Wang
Zhixuan Li
Zhiyu Jiang
Zhiyuan Yan
Zhiyuan Yu
Zhiyuan Zhang
Zhong Chen
Zhongwei Teng
Zhongzhan Huang
Zhongzhi Yu
Zhuan Han
Zhuangzhuang Chen
Zhuo Liu
Zhuo Su
Zhuojun Zou
Zhuoyue Wang
Ziang Song
Zicheng Zhang
Zied Mnasri
Zifan Chen
Žiga Babnik
Zijing Chen
Zikai Zhang
Ziling Huang
Zilong Du
Ziqi Cai
Ziqi Zhou
Zi-Rui Wang
Zirui Zhou
Ziwen He
Ziyao Zeng
Ziyi Zhang
Ziyue Xiang
Zonglei Jing
Zongyi Xu

Contents – Part VI

TaylorShift: Shifting the Complexity of Self-attention from Squared to Linear (and Back) Using Taylor-Softmax

Tobias Christian Nauen[1,2(✉)], Sebastian Palacio[2], and Andreas Dengel[1,2]

[1] Department of Computer Science, RPTU Kaiserslautern-Landau, Gottlieb-Daimler-Straße, Kaiserslautern, Germany
[2] German Research Center for Artificial Intelligence (DFKI), Trippstadter Str. 122, Kaiserslautern, Germany
{tobias_christian.nauen,andreas.dengel}@dfki.de, sebastian.palacio@de.abb.com

Abstract. The quadratic complexity of the attention mechanism represents one of the biggest hurdles for processing long sequences using Transformers. Current methods, relying on sparse representations or stateful recurrence, sacrifice token-to-token interactions, which ultimately leads to compromises in performance. This paper introduces TaylorShift, a novel reformulation of the Taylor softmax that enables computing full token-to-token interactions in linear time and space. We analytically determine the crossover points where employing TaylorShift becomes more efficient than traditional attention, aligning closely with empirical measurements. Specifically, our findings demonstrate that TaylorShift enhances memory efficiency for sequences as short as 800 tokens and accelerates inference for inputs of approximately 1700 tokens and beyond. For shorter sequences, TaylorShift scales comparably with the vanilla attention. Furthermore, a classification benchmark across five tasks involving long sequences reveals no degradation in accuracy when employing Transformers equipped with TaylorShift. For reproducibility, we provide access to our code under https://github.com/tobna/TaylorShift.

Keywords: Efficient Attention · Transformer · Machine Learning

1 Introduction

Ever since their introduction by Vaswani et al. [25], Transformers have revolutionized numerous domains of deep learning, from Natural Language Processing to Computer Vision, while also underpinning the emergence of novel applications such as Large Language Models. This success stems largely from their ability to capture intricate dependencies and token-to-token interactions.

To extend the utility of Transformers to more complex tasks, they need to be able to process long sequences. However, the computational complexity of

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_1.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 1–16, 2025.
https://doi.org/10.1007/978-3-031-78172-8_1

the attention mechanism scales quadratically in the length of the input sequence $\mathcal{O}(N^2)$. Therefore, computing twice as many sequence elements requires four times the number of computations, which hinders scaling to very long context windows. This makes some practitioners turn to approaches like compressing portions of the input into single states [3,5], which reduces the amount of information available at each step. Despite this progress, exploiting long context windows to significantly improve performance and incorporate new information without retraining remains challenging. Current Transformers encounter limitations when processing long documents, high-resolution images, or a combination of data from multiple domains and modalities. Especially, considering the limited resources of smaller enterprises or individual consumers.

While linearly scaling Transformers have been proposed, these often experience compromised accuracy [19], specialize in a particular domain, like language [30] or images [14], or only convey averaged global information across tokens, neglecting individual token-to-token interactions [1,8]. These models end up being ill-suited for handling longer sequences, leaving the standard Transformer as the preferred choice due to its large capacity and established performance [13].

In this work, we approach this bottleneck of the Transformer by reformulating the softmax function in the attention mechanism after introducing the Taylor approximation of the exponential. While some methods alter the softmax, their goal is to split interactions of queries and keys, computing global average interactions only [1,4]. In contrast, our proposed approach, TaylorShift, preserves individual token-to-token interactions. Combining a tensor-product-based operator with the Taylor approximation of the exponential function allows us to compute full token-to-token interactions in linear time. Moreover, this approach has the added benefit of adhering to concrete error bounds when viewed as an approximation of vanilla attention [11]. We show that a naive implementation of this linearization is numerically unstable and propose a novel normalization scheme that enables its practical implementation. For short sequences, TaylorShift can default back to quadratic scaling to preserve efficiency. We apply TaylorShift to a diverse set of tasks on images, text, and mathematical operations.

Our paper starts with the related work (Sect. 2), providing context for our contributions. In Sect. 3, we introduce two implementations of TaylorShift, efficient for short and long sequences, respectively, and our novel normalization scheme. Beyond the $\mathcal{O}$-notation, we delve into the efficiency analysis of TaylorShift, identifying specific conditions where it excels, both theoretically (Sect. 4) and empirically (Sect. 5). Finally, we conclude in Sect. 6.

2 Related Work

To contextualize TaylorShift, we review work on efficient attention, how Taylor approximations are used in ML, and their application for attention specifically.

Linear Complexity Attention. Various strategies have been proposed to devise attention mechanisms with linear complexity. Sparse attention mechanisms, like Swin [14] (images) or BigBird [30] (text), only selectively enable token-to-token interactions and their effectiveness heavily depends on the input modality.

Kernel-attention methods [1,4], decouple the influence of queries and keys, leading to a global average transformation instead of individual token-to-token interactions. Mechanisms like Linformer [27] apply transformations on the sequence direction, restricting them to a specific input size. For a comprehensive exploration of this topic, readers are referred to [9,19]. While these linear attention mechanisms offer innovative solutions to computational challenges, their performance nuances and lack of adaptability compared to TaylorShift warrant further exploration.

Taylor Approximation in ML. Applying Taylor approximations has proven to be a powerful technique in deep learning. In Explainable AI, the Deep Taylor Decomposition [17] employs a linear Taylor decomposition of individual neurons to propagate the relevancy of each part of the input. Linear Taylor approximations also are utilized in network pruning, where they are leveraged to quantify the influence of individual neurons on a loss value [10,16]. TE-CSR [28] directly utilizes a Taylor series to gather multivariate features in the domain of image fusion. Recently, TaylorNet [31] and Taylorformer [20] treat the factors of a Taylor series, as learnable parameters. The Taylor softmax [26], introduced to enable efficient calculation of loss values, outperformed the traditional softmax in image classification [2]. In this work, we leverage insights from these diverse applications of Taylor series to enable the efficient calculation of attention.

Taylor Approximation in Attention. Recently, [22] adopt the first order Taylor softmax in the attention mechanism. However, this is limited to linear token-interactions. To emphasize local interactions, they add a convolution operation. In contrast, we compute individual non-linear interactions in linear time.

[11], an analysis of efficient attention mechanisms, mentions the theoretical possibility of leveraging higher order Taylor softmax to approximate the attention mechanism in linear time, but with exponential complexity in the order of the Taylor approximation. In this work, we draw inspiration from this theoretical analysis and develop a viable, working implementation based on Taylor series. We analyze the efficiency gains beyond the $\mathcal{O}$-notation, estimating transition points where it outperforms standard attention.

3 TaylorShift

This section describes the formal derivation of TaylorShift and its algorithmic implementation. Starting from a direct, non-efficient formulation, we proceed to mathematically derive a provably efficient alternative. An investigation into scaling behaviors will lead to the incorporation of a novel normalization scheme.

3.1 Direct TaylorShift

Taylor-Softmax approximates the softmax's exponential function by its k-th order Taylor approximation:

$$\exp(x) \approx \sum_{n=0}^{k} \frac{x^n}{n!}.$$

For a vector $\mathbf{x} \in \mathbb{R}^d$, with Hadamard powers $\mathbf{x}^{\odot n}$:

$$\text{softmax}(\mathbf{x}) = \text{normalize}(\exp \mathbf{x}) \approx \text{normalize}\left(\sum_{n=0}^{k} \frac{\mathbf{x}^{\odot n}}{n!}\right) =: \text{T-SM}^{(k)}(\mathbf{x})$$

Here, the normalize operation is division by the ℓ^1-norm: $\mathbf{x} \mapsto \frac{\mathbf{x}}{\sum_i |\mathbf{x}_i|}$. For even k, Taylor-Softmax generates a probability distribution, since it is positive and its terms sum to one. $k = 2$ balances computational cost and expressivity [2].

By using Taylor-Softmax, the attention mechanism for the query, key, and value matrices $Q, K, V \in \mathbb{R}^{N \times d}$, where N is the length of the sequence and d is the internal dimension, takes the form

$$Y = \text{T-SM}\left(QK^\top\right) V \tag{1}$$

with row-wise Taylor-Softmax. We refer to the direct implementation of Eq. (1), which calculates the large $N \times N$ attention matrix $\text{T-SM}\left(QK^\top\right)$ of token-to-token interactions before multiplying it by V, as *direct-TaylorShift*.

3.2 Efficient TaylorShift

Since direct-TaylorShift does not scale well, we derive a more efficient implementation. We can achieve this, by splitting the influence of the taylor approximation of the exponential function among the matrices Q and K and pushing the normalization operation to the end, after multiplying by V. Mathematically the result will still be the same, but by switching up the order of operations, the computational complexity can be reduced from $\mathcal{O}(N^2 d)$ to $\mathcal{O}(N d^3)$.

First, we rewrite the normalization operation by splitting it into nominator and denominator:

$$\begin{aligned} Y_{\text{nom}} &= [1 + QK^\top + \frac{1}{2}(QK^\top)^{\odot 2}]V, \\ Y_{\text{denom}} &= [1 + QK^\top + \frac{1}{2}(QK^\top)^{\odot 2}]\mathbb{1}_N, \\ \Rightarrow Y &= Y_{\text{nom}} \oslash Y_{\text{denom}}, \end{aligned}$$

where $\mathbb{1}_N \in \mathbb{R}^N$ is the vector of ones and $^{\odot 2}$ and $\oslash$ are the Hadamard power and division. This representation allows us to disentangle the influence of the linear, squared, and constant terms of the Taylor approximation into their influence on Q and K, respectively.

The constant and linear influence $[1 + QK^\top]V = Q(K^\top V) + \Sigma_{\text{col}} V$ can trivially be computed in $\mathcal{O}(Nd^2)$, leaving us with $(QK^\top)^{\odot 2}V$. To handle this term efficiently, we define a tensor product on the internal dimension d:

$$\begin{aligned} \boxtimes &: \mathbb{R}^{N \times d} \times \mathbb{R}^{N \times d} \to \mathbb{R}^{N \times d^2} \\ [A \boxtimes B]_n &= \iota(A_n \otimes B_n) \in \mathbb{R}^{d^2} \quad \forall n = 1, ..., N \end{aligned}$$

Here, $A_n, B_n \in \mathbb{R}^d$, and $[A \boxtimes B]_n$ are the n-th entries of A, B, and $A \boxtimes B$ respectively, $\otimes$ is the outer product of vectors[1], and $\iota : \mathbb{R}^{d \times d} \xrightarrow{\sim} \mathbb{R}^{d^2}$ is the canonical isomorphism of reordering the entries of a matrix into a vector. This reordering operation can be described by a bijective map $\pi : \{1, ..., d\} \times \{1, ..., d\} \to \{1, ..., d^2\}$. We define $A^{\boxtimes 2} := A \boxtimes A$. Then we have $[A^{\boxtimes 2}]_{n,\pi(k,\ell)} = A_{n,k} A_{n,\ell}$. This lets us linearize $(QK^\top)^{\odot 2}$ by using the tensor operator $\boxtimes$ to unroll the square of a d-element sum along a sum of d^2 elements. At position ij, we have

$$\begin{aligned}
[(QK^\top)^{\odot 2}]_{ij} &= \left(\sum_{k=1}^{d} Q_{ik} K_{jk} \right)^2 = \sum_{k,\ell=1}^{d} Q_{ik} Q_{i\ell} K_{jk} K_{j\ell} \\
&= \sum_{k,\ell=1}^{d} [Q_i \otimes Q_i]_{k,\ell} [K_j \otimes K_j]_{k,\ell} = \sum_{k,\ell=1}^{d} [Q^{\boxtimes 2}]_{i,\pi(k,\ell)} [K^{\boxtimes 2}]_{j,\pi(k,\ell)} \\
&= [Q^{\boxtimes 2}]_i [K^{\boxtimes 2}]_j^\top .
\end{aligned}$$

for $i, j = 1, ..., N$. And therefore

$$\Rightarrow Y_{\text{squ}} := (QK^\top)^{\odot 2} V = \underbrace{Q^{\boxtimes 2}}_{N \times d^2} \underbrace{(K^{\boxtimes 2})^\top V}_{=: A_{\text{mod}}} \tag{2}$$

This can be calculated in linear time in N by multiplying from right to left. Adding both the linear and the constant terms to the square-term gives:

$$Y_{\text{nom}} = \frac{1}{2} Q^{\boxtimes 2} \left((K^{\boxtimes 2})^\top V \right) + Q(K^\top V) + \Sigma_{\text{col}} V. \tag{3}$$

We calculate the nominator Y_{nom} and denominator Y_{denom} simultaneously using Eq. (3) by setting $V \leftarrow (\mathbb{1}_N \circ V) \in \mathbb{R}^{N \times (d+1)}$, where $\circ$ is the concatenation operation. The result $\hat{Y} \in \mathbb{R}^{N \times (d+1)}$ can then be split back into $Y_{\text{denom}} \in \mathbb{R}^N$ and $Y_{\text{nom}} \in \mathbb{R}^{N \times d}$ to get the final output:

$$Y = \left[\frac{[Y_{\text{nom}}]_{1:}}{[Y_{\text{denom}}]_1}, ..., \frac{[Y_{\text{nom}}]_{N:}}{[Y_{\text{denom}}]_N} \right] \in \mathbb{R}^{N \times d}. \tag{4}$$

We refer to the result of Eq. (4) when calculating $\hat{Y} = Y_{\text{denom}} \circ Y_{\text{nom}}$ using Eq. (3) as *efficient-TaylorShift*. Figure 1 visualizes the differences between direct- and efficient-TaylorShift. The output of direct- and efficient-TaylorShift is the same mathematically, but the later scales linearly in N.

3.3 Normalization

Empirical evaluations reveal the presence of intermediate values with large norms, which ultimately leads to failure to converge during training[2]. Tracking the scaling behaviors (Table 1) of intermediate results in TaylorShift[3] lets us

[1] We identify the basis $\{e_i \otimes e_j\}_{ij}$ of tensor space with the canonical basis $\{e_{ij}\} \subset \mathbb{R}^{d \times d}$ of matrix space, viewed as a vector space. $\{e_i\}_i$ is the canonical basis of $\mathbb{R}^d$.

[2] See Appendix B.1 for further details.

[3] For more details see Appendix B.2.

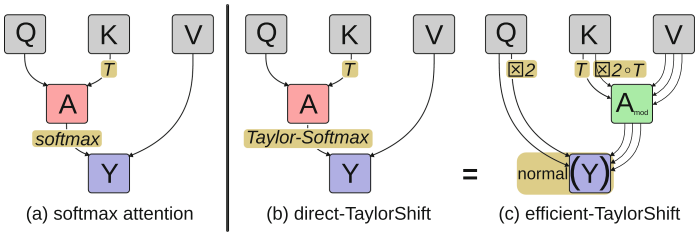


Fig. 1. Order of operations in softmax attention, direct-, and efficient-TaylorShift. Multi-paths for efficient-TaylorShift show squared, linear, and constant influence.

Table 1. Mean size of intermediate expressions in efficient-TaylorShift, when rows of Q, K, and V are sampled uniformly from the unit sphere.

Expr.	$(K^{\boxtimes 2})^\top V = A_{\text{mod}}$	$(QK^T)^2 V$	$QK^\top V$	Y_{denom}	Y
Size	$\frac{N+1}{\sqrt{d}}$	$\frac{N}{d}$	$\sqrt{N}\frac{4d+1}{4d}$	$N\frac{d+2}{2d}$	$\sqrt{\frac{d}{N}}$

define a normalization scheme that keeps these results from growing uncontrollably.

We first normalize the queries and keys and additionally introduce a per-head temperature parameter $\tau \in \mathbb{R}$[4], which ensures a constant input size:

$$q_i \leftarrow \frac{\tau q_i}{\|q_i\|_2}, \qquad k_i \leftarrow \frac{k_i}{\|k_i\|_2} \qquad \text{for } i = 1, ..., N.$$

Then, we counteract the scaling behaviors in Table 1 by multiplying Q and K by $\sqrt[4]{d}$ and V by $\frac{1}{N}$. To obtain the same output, we need to scale the factors of the Taylor series accordingly[5]. To ensure a consistent mean size of the output Y of TaylorShift, independent of N and d, we additionally multiply by $\sqrt{\frac{N}{d}}$[6]. We add the same normalization of the input and output to direct-TaylorShift to keep both implementations interchangeable. Algorithm 1 shows the full procedure to calculate efficient-TaylorShift with normalization.

4 Analysis of Efficiency Transition Points

We have seen that efficient-TaylorShift has a complexity of $\mathcal{O}(Nd^3)$, while its direct version stands at $\mathcal{O}(N^2 d)$. Therefore, the efficient implementation will be faster and more memory efficient for sufficiently large sequence lengths $N \gg d$.

[4] More details on the effect of normalizing compared to dividing by $d^{-\frac{1}{2}}$ (standard softmax attention does this) in Appendix B.3.

[5] From $\frac{1}{2}, 1, 1$ to $\frac{1}{2}, \sqrt{d}, d$ ($\frac{1}{2}, \alpha^2, \alpha^4$ in Line 9 of Algorithm 1), to counteract the factors of $\sqrt[4]{d}$.

[6] To save on computations, we scale the denominator by $\sqrt{\frac{d}{N}}$ in Line 5 of Algorithm 1.

Algorithm 1. Efficient-TaylorShift with normalization

Require: Queries, Keys and Values $Q, K, V \in \mathbb{R}^{N \times d}$

def $(A \boxtimes B)$: {A and B are of shape $N \times d$}
  $C \leftarrow A.\text{reshape}(N \times d \times 1) \odot B.\text{reshape}(N \times 1 \times d)$ {$\odot$ is the broadcasted Hadamard product. C has shape $N \times d \times d$.}
  return $C.\text{reshape}(N \times d^2)$
$\alpha \leftarrow \sqrt[4]{d}$
$V \leftarrow \frac{1}{N}\left(\left(\sqrt{\frac{d}{N}}\mathbb{1}_N\right) \circ V\right) \in \mathbb{R}^{N \times d+1}$
$Q, K \leftarrow \frac{\alpha\tau Q}{\|Q\|_{2,\text{dim}=-1}}, \frac{\alpha K}{\|K\|_{2,\text{dim}=-1}}$
$A_{\text{mod}} \leftarrow (K \boxtimes K)^\top V$
$\hat{Y} \leftarrow (Q \boxtimes Q)A_{\text{mod}}$
$\hat{Y} \leftarrow \frac{1}{2}\hat{Y} + \alpha^2 Q(K^\top V) + \alpha^4 \sum_{i=1}^{N} V_{i,:}$
$Y_{\text{denom}}, Y \leftarrow \hat{Y}_{:,:1}, \hat{Y}_{:,1:}$
$Y \leftarrow Y \oslash Y_{\text{denom}}$ {$\oslash$ is the Hadamard division.}
return Y

However, determining the exact value of N where this transition occurs is crucial for practical scenarios. This section analyzes the theoretical speed characteristics and memory requirements of both implementations to identify the specific point at which one outperforms the other independent of hardware considerations. Furthermore, we analyze additional factors influencing the efficiency of both implementations, providing a deeper understanding of their performance.

4.1 On the Floating-Point Operations

To identify the critical sequence length N_0 at which the efficient implementation surpasses the direct one in a hardware- and implementation-agnostic way, we inspect the number of floating-point operations involved. Starting with direct-TaylorShift, we follow Eq. (1) step by step. We need $2N^2d$ operations to multiply $QK^\top$, $4N^2$ operations to apply $x \mapsto \frac{1}{2}x^2 + x + 1$ element-wise to this $N \times N$ matrix, $2N^2$ operations for normalization, and $2N^2d$ operations for the final multiplication by V. The total FLOPS of direct-TaylorShift thus are

$$\text{ops}_{\text{triv}}[Y] = 2N^2d + 4N^2 + 2N^2 + 2N^2d = 4N^2d + 6N^2. \tag{5}$$

As the only difference between direct-TaylorShift and the standard attention mechanism is the choice of exp or its Taylor approximation, the number of operations needed for calculation of standard attention is slightly higher.

In contrast, for efficient-TaylorShift (Eq. (3)), the primary computation centers around the squared influence Y_{squ}. For $A_{\text{mod}} \in \mathbb{R}^{d^2 \times (d+1)}$ (Eq. (2)) the tensor operation has Nd^2 FLOPS and the subsequent matrix multiplication needs $2Nd^2(d+1)$. Factoring in the operations for the tensor operation on Q and the second matrix multiplication, the total FLOPS for calculating Y_{squ} are

$$\text{ops}[Y_{\text{squ}}] = 4Nd^2(d+1) + 2Nd^2.$$

Table 2. Influence of the hidden dimension d on the transitional points N_0 (speed) and N_1 (memory) based on Eq. (7) and (9) for typical d.

d	8	16	32	64	128
$\boldsymbol{N_0}$	73	273	1057	4161	16513
$\boldsymbol{N_1}$	47	159	574	2174	8446

Given the $4Nd(d+1)$ operations required to compute the linear influence $QK^\top V$, the $N(d+1)$ for summing up the columns of V, and the $3N(d+1)$ FLOPS for the sums and scalar multiplication, the total for calculating $\hat{Y}$ is

$$\begin{aligned}\text{ops}_{\text{eff}}[\hat{Y}] &= \text{ops}[Y_{\text{squ}}] + \text{ops}[QK^\top V] + \text{ops}[\Sigma_{\text{col}} V] + 3N(d+1) \\ &= 4\,Nd^2(d+1) + 2\,Nd^2 + 4\,Nd(d+1) + N(d+1) + 3\,N(d+1).\end{aligned}$$

Including the Nd operations for normalization, the total number of operations for efficient-TaylorShift is

$$\text{ops}_{\text{eff}}[Y] = N(4d^3 + 10d^2 + 9d + 4). \tag{6}$$

Comparing Eqs. (5) and (6) shows that for $N \to \infty$, efficient-TaylorShift outperforms direct-TaylorShift, but for $N \not\gg d$ the latter will still be faster. Let $N_0 = N_0(d)$ be the critical point, where $\text{ops}_{\text{triv}}[Y] = \text{ops}_{\text{eff}}[Y]$. We calculate

$$N_0 = \frac{4d^3 + 10d^2 + 9d + 4}{4d + 6} \leq d^2 + d + \frac{3}{4}. \tag{7}$$

For details on the derivation of N_0, see Appendix A.1. Since the value of d is typically fixed, we can easily compute the transitional input length N_0 for common choices of d. The values for typical d can be found in Table 2.

4.2 On Memory

In addition to the number of operations, the memory footprint plays an important role as excessive memory needs can result in the inability to run a model altogether. To assess it, we examine the largest tensors that have to be stored simultaneously, omitting memory needed for model parameters.

For direct-TaylorShift, maximum memory usage occurs when calculating the attention matrix $\text{T-SM}\left(QK^\top\right)$ from $QK^\top$. Here, we store matrices $QK^\top$ and V, as well as space for the output[7] resulting in a total of

$$\text{entries}_{\text{triv}}[Y] = \underbrace{dN}_{\text{for } V} + \underbrace{2\,N^2}_{\text{for } QK^\top \text{ and result}}.$$

[7] Calculating the sum in $\frac{1}{2}x^2 + x$ requires saving the original value.

Conversely, the efficient version requires maximum memory during the calculation of A_{mod} (Eq. (2)). Here, the matrices $(K^{\boxtimes 2})^\top$, V, and space for the result are needed, along with Q and K for later calculations for a total of

$$\text{entries}_{\text{eff}}[Y] = \underbrace{d^2(d+1)}_{\text{for } A_{\text{mod}}} + \underbrace{2dN}_{\text{for } Q,K} + \underbrace{(d+1)N}_{\text{for } V} + \underbrace{d^2\,N}_{\text{for } K^{\boxtimes 2}} \tag{8}$$

matrix entries. It is evident that $\text{entries}_{\text{triv}}[Y] > \text{entries}_{\text{eff}}[Y]$ for all N bigger than some constant $N_1 = N_1(d)$. This marks the transitional point beyond which efficient-TaylorShift becomes more memory efficient than direct-TaylorShift. By setting $\text{entries}_{\text{triv}}[Y] = \text{entries}_{\text{eff}}[Y]$ for $N = N_1$, we find

$$N_1 = \frac{1}{4}\left[d^2 + 2d + 1 + \sqrt{d^4 + 12d^3 + 14d^2 + 4d + 1}\right] \leq \frac{1}{2}d^2 + 2d + \frac{1}{2}. \tag{9}$$

Refer to Appendix A.4 for a detailed derivation. Notably, from Table 2, we observe that N_1 is considerably smaller than N_0 highlighting the extra memory efficiency of efficient-TaylorShift.

4.3 Changing the Number of Attention Heads h

In an effort to reduce the number of operations while retaining the ability to process the same number of tokens N, one might opt to reduce the internal dimension d. However, this might come at the cost of expressiveness. Given that efficient-TaylorShift has a cubed complexity in d, an alternative strategy involves increasing the number of attention heads in the multi-head-attention mechanism. Let each token be $d_{\text{emb}} \in \mathbb{N}$ dimensional and let $h \in \mathbb{N}$ be the number of attention heads (with $h | d_{\text{emb}}$). Then, in each head, the queries, keys, and values are $d = \frac{d_{\text{emb}}}{h}$-dimensional, with the computational cost of the multi-head self-attention (MHSA) mechanism being h times that of a single attention head. For direct-TaylorShift (Eq. (5)), the cost becomes

$$\text{ops}_{\text{triv}}[\text{MHSA}] = h\,\text{ops}_{\text{triv}}[Y] = h(4N^2d + 6N^2) = 4\,N^2 d_{\text{emb}} + 6\,hN^2,$$

which strictly increases in h. In contrast, using efficient-TaylorShift, we obtain

$$\begin{aligned}\text{ops}_{\text{eff}}[\text{MHSA}] &= h\,\text{ops}_{\text{eff}}[Y] = hN(4d^3 + 10d^2 + 9d + 4)\\ &= N\left(4\frac{d_{\text{emb}}^3}{h^2} + 10\frac{d_{\text{emb}}^2}{h} + 9d_{\text{emb}} + 4h\right).\end{aligned}$$

Given that $\text{ops}_{\text{eff}}[\text{MHSA}]$ diverges for $h \to 0, \infty$, there exists an optimal $\hat{h}_0 = \hat{h}_0(d_{\text{emb}})$ that minimizes the number of operations. Setting the derivative of $\text{ops}_{\text{eff}}[\text{MHSA}]$ with respect to h to zero, we find

$$0 = \frac{\partial}{\partial h}\,\text{ops}_{\text{eff}}[\text{MHSA}] = N\left(4 - 9\frac{d_{\text{emb}}^3}{h^3} - 10\frac{d_{\text{emb}}^2}{h^2}\right) \overset{N>0}{\Leftrightarrow} 4 = 9d^3 + 10d^2. \tag{10}$$

This has a single positive solution of $d \approx 0.52$, minimizing the number of operations at $\hat{h}_0 \approx \frac{1}{0.52} d_{\text{emb}}$. For a detailed derivation refer to Appendix A.2. In particular, the number of operations of efficient-TaylorShift decreases when h increases in the range of possible values $\{1, 2, ..., d_{\text{emb}}\}$ (divisors of d_{emb}).

Examining memory costs provides another perspective on the impact of attention heads. On one hand, for direct-TaylorShift the number of simultaneous entries strictly increases with the number of attention heads h, when calculating heads in parallel:

$$\text{entries}_{\text{triv}}[\text{MHSA}] = h \, \text{entries}_{\text{triv}}[Y] = d_{\text{emb}} N + 2\, N^2\, h$$

On the other, for efficient-TaylorShift, the number of entries is

$$\begin{aligned} \text{entries}_{\text{eff}}[\text{MHSA}] &= h \, \text{entries}_{\text{eff}}[Y] = h(d^3 + (N+1)d^2 + 3Nd + N) \\ &= \frac{d_{\text{emb}}^3}{h^2} + (N+1)\frac{d_{\text{emb}}^2}{h} + 3N d_{\text{emb}} + Nh. \end{aligned}$$

This expression again diverges as $h \to 0, \infty$ and therefore an optimum $\hat{h}_1$ exists. Setting the derivative to zero gives

$$0 = \frac{\partial}{\partial h} \text{entries}_{\text{eff}}[\text{MHSA}] = -2d^3 - (N+1)d^2 + N, \tag{11}$$

which implies $d < 1$ and therefore $\hat{h}_1 > d_{\text{emb}}$. Refer to Appendix A.3 for the detailed derivation. In particular, the memory cost also decreases with increasing h in the allowed range $\{1, ..., d_{\text{emb}}\}$. The same holds true when calculating the attention heads in sequence (Eq. (8) is strictly increasing in d). Our analysis provides insight into the dynamic efficiency interplay between the two implementations and the number of attention heads.

5 Empirical Evaluation

We run a number of experiments that provide an empirical verification of our theoretical analysis of the transitional bounds, scalability, and required computational resources, as well as of the effective capacity of our proposed mechanism.

5.1 Efficiency of the TaylorShift Module

To validate our theoretical analysis of the critical points N_0 and N_1 from Sect. 4, we compare the speed and memory usage of TaylorShift and softmax attention [25] using simulated data. For multiple internal dimensions d and sequence lengths N, we measure inference time and memory consumption of a single attention head on an NVIDIA A100 GPU. For comparison, applications like GPT-2 [23] or ViT [7], use a per-head dimension of $d = 64$.

In Fig. 2 (top), we contrast the speed of TaylorShift and softmax attention. The quadratic growth of softmax attention and direct-TaylorShift and

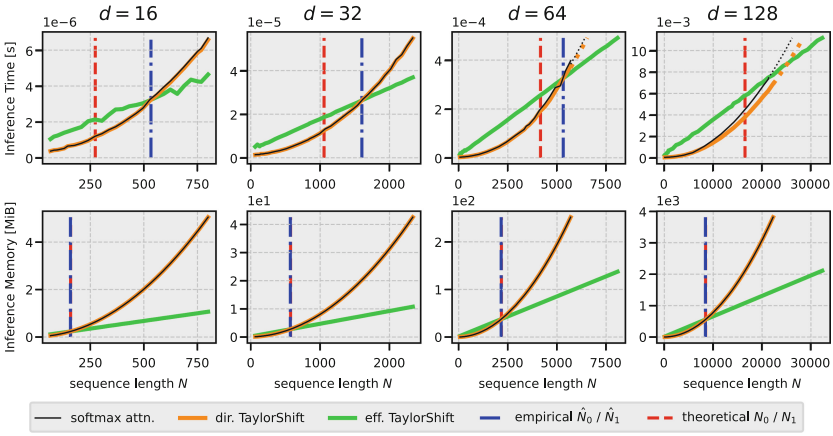


Fig. 2. Inference time in seconds per input (top) and inference memory in MiB (bottom) of the attention mechanism (with $h = 1$) vs. sequence length for both implementations of TaylorShift and softmax attention. Each column uses a different internal dimension d. We mark the theoretical N_0 and N_1 and empirical intersections $\hat{N}_0$ and $\hat{N}_1$. Dotted lines extrapolate values by fitting a parabola.

the linear growth of efficient-TaylorShift are evident. As noted in Sect. 4.1, we observe a slightly higher number of FLOPS for softmax attention than for direct-TaylorShift. Note that the difference between the theoretical N_0 and empirical $\hat{N}_0$ transition points $\hat{N}_0 - N_0 \approx 18d$ is approximately proportional to d. We hypothesize that the more sequential nature of efficient-TaylorShift results in more, costly reads and writes in GPU memory. This indicates possible efficiency gains for eff. TaylorShift from a low-level IO-efficient implementation.[8]

Due to increasing memory requirements for direct-TaylorShift and softmax attention, plotted in the second row, we need to extrapolate the plots for $d = 64$ and $d = 128$ by fitting a parabola (dotted lines) to the data In the regimen of memory (second row of Fig. 2), the theoretical and empirical intersections align closely $\hat{N}_1 \approx N_1$, with an error of at most 0.6%. Comparing both rows shows efficient-TaylorShift becoming memory efficient earlier than it becomes efficient in terms of speed, highlighting its usefulness in low-memory environments, in alignment with our theoretical results from Table 2.

5.2 Efficiency of a Transformer with TaylorShift

We show the efficiency of a full-scale[9] Transformer encoder equipped with TaylorShift in Fig. 3.[10] At a sequence length of 900 tokens efficient-TaylorShift needs less memory and at 1800 tokens it surpasses the standard Transformer in speed.

[8] For more details, see Appendix D.2.
[9] Here, we use the hyperparameters for ListOps from Appendix C, but with 16 heads.
[10] The extended Fig. 3 in Appendix D.4 includes different numbers of heads.

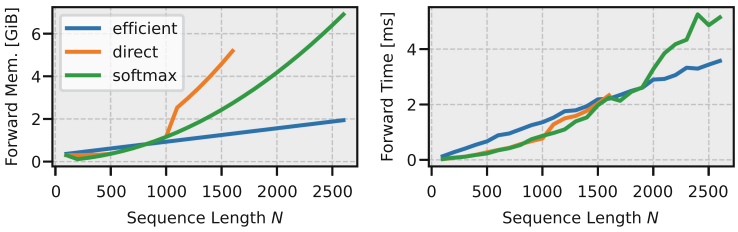


Fig. 3. Memory and inference time of a transformer with efficient- and direct-TaylorShift and the standard softmax, using $d = 32$.

Table 3. Accuracy in percent for models on datasets of different modalities. For the first three datasets, we closely adhere to the setup of [24]. Models with * had to be trained with full instead of mixed precision.

Model	CIFAR (Pixel)	IMDB (Byte)	ListOps	ImageNet (Ti)	ImageNet (S)	Average
Linformer [27]	29.2	58.1	–	64.3	76.3	(57.0)
RFA [21]	44.9	65.8	–	–	–	(55.4)
Performer [4]	34.2*	65.6*	35.4*	62.0*	67.1*	52.9
Reformer [12]	44.8	63.9	**47.6**	73.6	76.2*	61.2
Nystromformer [29]	**49.4**	65.6	44.5	75.0	78.3*	62.6
EVA [32]	46.1	64.0	45.3	73.4	78.2	61.4
Transformer [25]	44.7	65.8	46.0	**75.6**	79.1	62.2
Ours	47.6	**66.2**	46.1	75.0	**79.3**	**62.8**

Note that at 1500 tokens it only needs half and at 2000 tokens only 35% of the Transformer's memory. For shorter sequence length, direct-TaylorShift remains competitive with a standard Transformer in terms of speed and memory.

5.3 Performance of a Transformer with TaylorShift

To assess the effectiveness of TaylorShift, we evaluate it using a Transformer-encoder across various datasets representing different modalities. We track the classification accuracy of a TaylorShift-equipped Transformer across five tasks.

Tasks. We train on three datasets introduced by [24], specially designed to assess performance on long sequences with long-range dependencies. The first is a pixel-level CIFAR10 task, where 8-bit intensity values of grayscale images from CIFAR10 are encoded into a sequence of length 1024. In the domain of text, IMDB Byte [15], is a classification task for text encoded at the character level, resulting in sequences of 4000 tokens. Thirdly, we employ the Long ListOps dataset of mathematical operations [18] of length 500 to 2000 tokens encoded at the character level. Beyond these synthetic tasks, we train for classification on ImageNet [6] at two sizes (Ti & S) to additionally evaluate the scaling behavior of TaylorShift. Refer to Appendix C for model sizes and training hyperparameters. We utilize mixed-precision calculations whenever possible.

Table 4. Accuracy on the CIFAR Pixel task when ablating our novel normalization introduced in Sect. 3.3.

Model	direct	efficient
Plain impl.	47.1	–
impl. +norm.	46.8	46.8
impl. +norm. +output norm.	47.5	47.6

Table 3 shows our method's consistent performance across all datasets. It surpasses all other linear scaling Transformers on a minimum of four out of five datasets. Note, that those models marked with * only work using full precision, slowing down training considerably. TaylorShift also outperforms the standard Transformer on four out of five tasks, remains competitive on the last one. We observe a notable increase of 4.3% when transitioning from size Ti to S on ImageNet, in contrast to 3.5% for the Transformer. These findings highlight the robustness and competitiveness of TaylorShift across diverse datasets and modalities. This demonstrates TaylorShift's usefulness when dealing with very long sequences.

5.4 Ablations

We conduct an ablation analysis, systematically dissecting two key components to establish their impact on the performance of TaylorShift.

Normalization. We train a Transformer equipped with TaylorShift at different stages of normalization to track the impact of our normalization scheme. Table 4 shows that without normalization, direct-TaylorShift demonstrates acceptable performance, while the efficient version fails to converge during training. We attribute this to numerical overflow in intermediate results[11]. Upon introducing input normalization to the attention mechanism, efficient-TaylorShift becomes stable, and both implementations achieve an accuracy of 46.8%, a slight decrease for direct-TaylorShift. Additionally, normalizing the output to a mean size of 1, results in a performance boost for both implementations, bringing them to the accuracy level observed for the direct version before.

Number of Attention Heads h. Finally, we validate our insights from Sect. 4.3 by training a TaylorShift-equipped encoder with varying numbers of attention heads h while maintaining the embedding dimension d_{embed}. Note that the number of parameters stays almost exactly constant, with only the shape of the attention temperature per head (τ) changing. The results in Table 5 align with our theoretical analysis, demonstrating an acceleration and less memory demands as the number of heads increases. Notably, increasing the number of heads often leads to increased accuracy while concurrently speeding up calculations and reducing memory. These efficiency gains will become more significant for sequences longer than the 1024 tokens we tested with. Beyond the point where accuracy increases,

[11] See also Appendix B.1.

Table 5. Accuracy, throughput (TB), and VRAM (Mem) usage of TaylorShift on the CIFAR Pixel task with different number of attention heads h. All models have $d_{\text{embed}} = 256$ with $d = \frac{d_{\text{embed}}}{h}$ in the attention mechanism.

h	d	Acc [%]	direct		efficient	
			TP [ims/s]	Mem [MiB@16]	TP [ims/s]	Mem [MiB@16]
4	64	47.1	12 060	596	2 975	840
8	32	47.5	7 657	1 111	5 749	585
16	16	47.3	4 341	2 135	9 713	459
32	8	46.9	2 528	4 187	14 087	397
64	5	45.9	1 235	8 291	13 480	125

we can still leverage additional heads to trade off accuracy against speed and memory, particularly advantageous for processing longer sequences.

6 Conclusion

We present TaylorShift, an efficient attention mechanism that computes token-to-token interactions in linear time and memory. We lay the theoretical groundwork for using TaylorShift by studying the exact threshold values where it becomes efficient. Empirical validation of our analysis through classification experiments confirms the performance benefits of TaylorShift for long sequences. TaylorShift even outperforms a standard Transformer across diverse datasets and modalities by 0.6% on average, while being faster and using less than half the memory for sequences longer than 2000 tokens. Furthermore, our results on the number of attention heads reaffirm the efficiency gains predicted theoretically. The number of heads can be tuned to improve the model's effective capacity, its speed, and reduce memory requirements, all at once. While efficient-TaylorShift is faster than a standard Transformer for long sequences, we can swap back to the interchangeable direct-TaylorShift variant to keep the model efficient for short sequences. This can be useful when dealing with datasets containing sequences of vastly different length, like text or time-series, or when using a curriculum to build up to very long sequence tasks. By adopting TaylorShift, it will be possible to tackle tasks featuring long sequences such as high-resolution image classification and segmentation, processing long documents, integrating data from multiple modalities, and dynamically encoding lengthy documents into a prompt-specific context for Large Language Models. Overall, our findings underscore the efficiency and versatility of TaylorShift, positioning it as a competitive and scalable option in the landscape of efficient attention-based models.

Acknowledgments. This work was funded by the Carl-Zeiss Foundation under the Sustainable Embedded AI project (P2021-02-009) and by the EU project SustainML (Horizon Europe grant agreement No 101070408).

References

1. Babiloni, F., et al.: Linear complexity self-attention with 3rd order polynomials. TPAMI **45**, 12726–12737 (2023)
2. de Brébisson, A., Vincent, P.: An exploration of softmax alternatives belonging to the spherical loss family. In: Bengio, Y., LeCun, Y. (eds.) ICLR (2016)
3. Bulatov, A., Kuratov, Y., Burtsev, M.S.: Scaling transformer to 1m tokens and beyond with RMT (2023)
4. Choromanski, K.M., et al.: Rethinking attention with performers. In: ICLR (2021)
5. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-XL: attentive language models beyond a fixed-length context. In: ACL, pp. 2978–2988. Association for Computational Linguistics (2019)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: ICPR. IEEE (2009)
7. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale. In: ICLR (2021)
8. El-Nouby, A., et al.: XCiT: cross-covariance image transformers. In: NeurIPS (2021)
9. Fournier, Q., Caron, G.M., Aloise, D.: A practical survey on faster and lighter transformers. ACM Comput. Surv. **55**, 1–40 (2023)
10. Gaikwad, A.S., El-Sharkawy, M.: Pruning convolution neural network (squeezenet) using Taylor expansion-based criterion. In: ISSPIT. IEEE (2018)
11. Keles, F.D., Wijewardena, P.M., Hegde, C., Keles, F.D., Wijewardena, P.M., Hegde, C.: On the computational complexity of self-attention. In: ALT (2023)
12. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: the efficient transformer. ICLR (2020)
13. Lin, T., Wang, Y., Liu, X., Qiu, X.: A survey of transformers. AI Open
14. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: ICCV (2021)
15. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Lin, D., Matsumoto, Y., Mihalcea, R. (eds.) ACL (2011)
16. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: ICLR (2017)
17. Montavon, G., Bach, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recogn. **65**, 211–222 (2015)
18. Nangia, N., Bowman, S.: ListOps: a diagnostic dataset for latent tree learning. In: Cordeiro, S.R., Oraby, S., Pavalanathan, U., Rim, K. (eds.) NAACL (2018)
19. Nauen, T.C., Palacio, S., Dengel, A.: Which transformer to favor: a comparative analysis of efficiency in vision transformers (2023)
20. Nivron, O., Parthipan, R., Wischik, D.: Taylorformer: probabalistic modelling for random processes including time series. In: ICMLW (2023)
21. Peng, H., Pappas, N., Yogatama, D., Schwartz, R., Smith, N.A., Kong, L.: Random feature attention. In: ICLR (2021)
22. Qiu, Y., Zhang, K., Wang, C., Luo, W., Li, H., Jin, Z.: MB-TaylorFormer: multi-branch efficient transformer expanded by Taylor formula for image dehazing. In: ICCV (2023)
23. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)

24. Tay, Y., et al.: Long range arena: a benchmark for efficient transformers. In: ICLR (2021)
25. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) NeurIPS (2017)
26. Vincent, P., de Brébisson, A., Bouthillier, X.: Efficient exact gradient update for training deep networks with very large sparse targets. In: NeurIPS (2015)
27. Wang, S., Li, B.Z., Khabsa, M., Fang, H., Ma, H.: Linformer: self-attention with linear complexity (2020)
28. Xing, C., Wang, M., Dong, C., Duan, C., Wang, Z.: Using Taylor expansion and convolutional sparse representation for image fusion. Neurocomputing **402**, 437–455 (2020)
29. Xiong, Y., et al.: Nyströmformer: a Nyström-based algorithm for approximating self-attention. In: AAAI (2021)
30. Zaheer, M., et al.: Big bird: transformers for longer sequences. In: NeurIPS (2020)
31. Zhao, H., et al.: TaylorNet: a Taylor-driven generic neural architecture (2023)
32. Zheng, L., Yuan, J., Wang, C., Kong, L.: Efficient attention via control variates. In: ICLR (2023)

Balancing Accuracy and Efficiency in Budget-Aware Early-Exiting Neural Networks

Youva Addad(✉), Alexis Lechervy, and Frédéric Jurie

GREYC, Normandy University, UNICAEN, ENSICAEN, UMR CNRS 6072, Caen, France
{youva.addad,alexis.lechervy,frederic.jurie}@unicaen.fr

Abstract. This paper presents an Early Exit Neural Network (EENN) architecture, which enables budgeted classification by dynamically selecting the most relevant exit point for each input sample of a dataset to achieve the best performance while adhering to a pre-defined computational budget. The key contribution of this work is a novel method that jointly learns the classifier model and the sample exiting policy, in contrast to prior approaches that treated these components separately. Specifically, the paper introduces a bi-level optimization framework that simultaneously optimizes the cross-entropy loss of the classifier and the probabilities of each sample exiting at different stages of the network. This joint learning approach allows the classifier parameters and the sample-dependent exiting policy to be mutually optimized, leading to improved classification accuracy under computational constraints. The proposed EENN method is evaluated on three computer vision benchmarks - CIFAR-10, CIFAR-100, and ImageNet - and demonstrates state-of-the-art results in budgeted classification compared to existing early exit strategies. The code for this work will be made publicly available upon acceptance of the paper.

Keywords: Computer Vision · Dynamic Neural Network · Weighting Sample · Early Exit

1 Introduction

Recent image classification algorithms have achieved impressive results on benchmarks such as the Large Scale Visual Recognition Challenge (LSVRC) [20]. However, this progress has come at the cost of increasingly large and computationally intensive models. In traditional deep learning architectures, each input sample follows a fixed path through the entire network, regardless of its inherent

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_2.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 17–31, 2025.
https://doi.org/10.1007/978-3-031-78172-8_2

difficulty. This approach introduces unnecessary computational overhead, especially when dealing with datasets that contain a mix of easy and challenging samples. This limitation makes real-time inference on resource-constrained platforms, such as smartphones, wearable devices, robotics, and other edge devices, almost unattainable.

To address this issue, researchers have explored various approaches to improve the inference efficiency of deep convolutional neural networks (CNNs). These include efficient architecture design [7,14,36], network pruning [11,34], weight quantization [17,18], knowledge distillation [13,21], adaptive inference [10,15,25,32,33], and efficient deployment on hardware [5,30].

Dynamic early-exiting networks, a type of adaptive inference, address the inefficiencies of traditional deep learning architectures by introducing branch points at different depths within the network [29]. The key idea is to intelligently select network segments to execute based on the input sample: easy-to-classify inputs can exit the network earlier, reducing computation, while complex inputs pass through deeper layers. This adaptive decision-making allows dynamic early-exiting networks to balance computational efficiency and predictive accuracy. By selectively executing only necessary segments, these models can significantly reduce computational cost while maintaining overall performance.

Furthermore, dynamic early-exiting networks can tailor the inference process to available resources and latency requirements. In time-constrained scenarios, the model can prioritize early exits for faster predictions; in resource-rich settings, it can explore deeper branches to improve accuracy. This flexibility makes dynamic early-exiting networks compelling for deployment across devices, from resource-constrained edge to powerful server systems. By intelligently allocating resources based on input, these networks can deliver efficient, high-performing inference for diverse real-world applications.

The importance of dynamic early-exiting networks has driven substantial research into improving their mechanisms. Key advancements in this active area include maximizing reuse of computations between classifiers [15], leveraging self-distillation techniques to efficiently transfer knowledge between network exits [9,23,25], and processing of successive small input regions to enable more dynamic exiting decisions [32]. Researchers have also explored resolution-adaptive network architectures [33,37] and developed calibration and sample weighting methods to improve the early exit decisions [10,26]. Furthermore, the field has seen exploration of transformer-based models as an alternative to convolutional networks for dynamic early-exiting [4,9,31]. This steady stream of innovations has led to ever-increasing sophistication in dynamic early-exiting network mechanisms, further enhancing their efficiency and effectiveness.

The fundamental challenge in this domain stems from the inherent train-test mismatch [8]. The classification network is typically trained without considering the constraints of a limited inference budget. Each exit classifier is optimized across the entire training dataset, without accounting for the fact that during inference, not all classifiers may encounter all types of testing data. In scenarios with resource limitations or easily manageable inputs, only the shallow layers and

classifiers are activated, leading to a disparity in the data distribution between training and testing. While "easy" examples may contribute to regularizing deep classifiers during training, overemphasizing these samples can exacerbate the distribution mismatch issue.

In this work, we assume that the inference budget will be allocated by specifying the ratio of samples that must exit at each checkpoint. This means the classifiers must not only infer the correct classes but also rank the samples appropriately to meet the specified exit criteria within the given overall budget. This requirement calls for a specialized training procedure, which we address by introducing a novel gater network that is independent of both the backbone and the classifiers. The gater network takes various confidence measures as input and generates probabilities for selecting the best exit. These probabilities are then transformed into sample weights. Since the predictor network and gater network operate independently, we propose a bi-level optimization approach to co-train them.

The contributions outlined in this paper can be summarized as follows:

- We propose a novel gater network that takes multiple confidence measures as input and produces probabilities for selecting the optimal exit point during training.
- We introduce an alternative approach to modeling the likelihood of early exiting, which plays a crucial role in achieving the desired balance between accuracy and computational efficiency.
- We formulate the training process as a bi-level optimization problem, enabling the simultaneous training of the predictor network and the gater network. This optimization scheme evaluates both the accuracy and the inference cost during the training phase.
- We conduct extensive experiments on three widely-used datasets: CIFAR-10, CIFAR-100, and ImageNet. These comprehensive evaluations demonstrate the effectiveness of our proposed approach in delivering efficient and high-performing dynamic early-exiting inference.

2 Related Work

Dynamic Early Exiting is an emerging technique in deep learning that focuses on improving inference efficiency [9,10,15,25,32,33]. It allows models to exit prediction early for certain input instances, reducing unnecessary computation without significant loss of accuracy. Dynamic Early Exiting uses adaptive criteria, such as confidence thresholds [29] or uncertainty measures [26], to decide whether to exit the inference process early for an input. This approach has several advantages, including reduced inference time [15], improved scalability [8] for resource-constrained environments [22], and potential energy savings [22]. While dynamic early exit has attracted attention for its benefits, challenges remain in finding the right criteria to balance computational gains and accuracy preservation. However, recent research [9,10,15,31] has shown promising results and has

been applied to several domains, including computer vision [3], natural language processing [12] and speech recognition [27]. Some authors have attempted to deviate from the multi-exit framework, for example in [35], where the authors proposed a boosting-like neural architecture, but the performance is comparable to multi-exit approaches such as MSDNet [15].

The *best performing approaches* in multi-exit architectures are based on BranchyNet [29], which was one of the first papers to propose such an efficient architecture, improved later by MSDNet [15], which introduced the concepts of *anytime classification* and *budgeted batch classification* with multiple classifiers applied adaptively during test time. Building on the foundations of this architecture, [1] has proposed a more appropriate way of fusing the network's intermediary outputs. In [33], the Resolution Adaptive Network (RANet) introduced the idea of performing resolution adaptive learning in deep CNNs within this multi-exit framework. In contrast, in L2W [10] the authors observed that MSDNet treats all samples for all exists during training, ignoring the early-exit behaviour that occurs during testing, and proposed to compensate for this by weighting training samples according to their difficulty. In the very recent paper [26], the authors proposed a novel method for estimating uncertainty in dynamic neural networks, which allows to better distinguish between easy and hard examples. This question was also investigated in [2]. It is also worth mentioning the approach presented in [24], which proposes an online knowledge distillation mechanism for multi-exit networks. Another approach using attention is the Dynamic Vision Transformer (DVT) [31] and the Coarse-to-Fine Vision Transformer (CF-ViT) [4]. Both methods share a common principle, emphasising that it is suboptimal to process all samples with the same number of tokens. The most recent model, Dynamic Perceiver (Dyn-Perceiver) [9], advocates disentangling the feature extractor and classifier branches due to the problem of classifier interference.

One limitation of these existing methods is that they do not specifically optimize the backbone network for budgeted inference. In contrast, our approach directly addresses this by jointly training the backbone and gating mechanisms to make efficient early-exiting decisions under resource constraints.

3 Presentation of the Contributions

As mentioned earlier, the key idea of our approach is to jointly train the classification network and the early exits. However, this presents a challenge, as the two components are interdependent: the performance of the early exits depends on the parameters of the classification network, and vice versa. This mutual dependence complicates the optimization process, as the optimal solution for one component cannot be determined independently of the other. Figure 1 provides an illustration of the overall training procedure.

Early Exiting at Inference Time. Once the multi-exit neural network is trained, early exit is performed during inference by defining thresholds $\eta^{(k)}$,

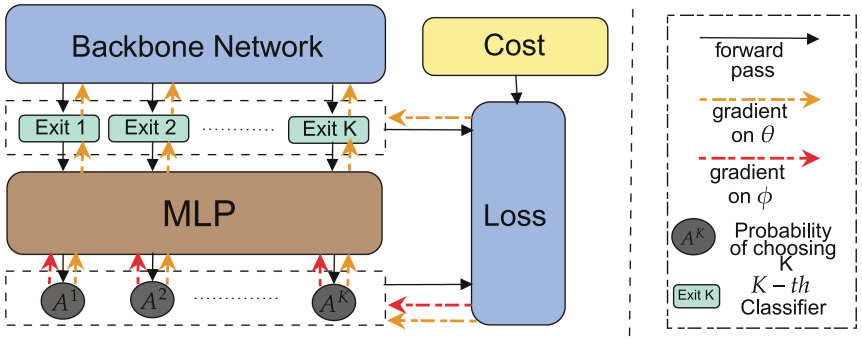


Fig. 1. Our Training Method: An Overview. The yellow arrow in the architecture represents the gradient utilized to update the entire backbone and exits. On the other hand, the red arrow signifies the gradient employed for updating the MLP scorer. The values of A^K are determined using the formula provided in Eq. (5). It's worth noting that the cost can be viewed as a scalar regularization term. (Color figure online)

where k refers to the k-th exit. When processing input examples x_i, the examples are sequentially passed through each classifier exit f^k until the maximum predicted class probability, denoted as $\max_c \text{softmax}\left(f^k(x_i)\right)$, exceeds the corresponding threshold $\eta^{(k)}$. At this point, the network returns the predicted class c.

The threshold values $\eta^{(k)}$ are usually determined using validation data, with the goal of ensuring that the overall processing remains within the specified computation budget. The typical approach, which we also employ in this work, is to set the thresholds such that a fixed fraction q, with $0 < q \leq 1$, of the samples reaching a classifier will obtain a confident enough classification to exit. This fixed probability q is applied consistently across all exits. Effectively, setting q is equivalent to defining the desired computation budget, as it directly determines the expected number of samples that will exit at each stage of the network. The key point is that the classification network should be trained to provide higher classification scores for the examples that should be the first to exit.

Training Multi-exit Networks. We have established that the classification scores provided by the network should be higher for examples that are intended to exit the network earlier. This can be the case for examples that are easier to classify or examples where further propagation through the network would not yield significant performance gains given the computational budget. This means the classification network needs to be trained with the understanding that some examples will be prioritized for earlier exit.

Let $\mathcal{D}$ be a set of training samples (x_i, y_i) for i ranging from 1 to N, where x_i denotes the input features (e.g., images) and $y_i \in \mathcal{C}$ represents the corresponding class labels. The set $\mathcal{C} = \{1, 2, \ldots, C\}$ encompasses all potential classes. The

objective is to train the Multi-Exit Neural Network (MENN) model f parameterized by θ.

Let $k \in [0, K]$ be the index of each exit, where f^k (parameterized by θ^k) represents a sub-network responsible for generating the k-th output and aiming to predict the target using different computational resources. In practice, f^k often shares layers with lower-cost networks, allowing for partial reuse of computations. This design choice enables the network to efficiently allocate computational resources based on the requirements of each input sample.

The objective stated above translates into the loss:

$$\mathcal{L}(\theta; \mathcal{D}) = \mathbb{E}_{x \sim \Pr(x)} \left[\mathbb{E}_{k \sim \Pr(k|x)} \left(\ell^{CE}(\hat{y}^k, y) \right) \right], \tag{1}$$

where $\hat{y}^k$ is the label predicted by f^k, $\mathcal{L}^{CE}(\theta; \mathcal{D}) = \mathbb{E}_{x \sim \Pr(x)} \left[\ell^{CE}(\hat{y}^k, y) \right]$ is the standard cross-entropy loss where $\ell^{CE}(\hat{y}^k, y) = -\log \Pr_{\hat{y}=y}(x; \theta)$, and $\Pr(k|x)$ represents the probability for the input x to exit at the k-th exit.

Traditionally, Early Exit Neural Network (EENN) methods have assumed a fixed probability of exiting at each layer, i.e., $\Pr(k|x) = \frac{1}{K}$. This simplifies the loss function to the average of the cross-entropies at each output. We relax this assumption of a fixed exit probability. Instead, we consider a sample-dependent probability of exiting, $\Pr(k|x)$, which can vary across samples. This sample-dependent exit probability is important because it allows the network to adapt its computational budget more flexibly to each input, while still maintaining the overall budget constraint, at train time.

Addressing Early Exit During Training. As mentioned in the previous section, we want the classification network to be learned by taking into account the fact that not all samples have an equal probability of exiting at each exit and must respect a given budget. Leveraging the work in [16], we integrate the probability of exiting in the loss, which also leads to the introduction of a per-exit cost function, as the early exiting probabilities depends on the overall budget. We propose to translate this into the following loss:

$$\mathcal{L}(\theta; \mathcal{D}) = \mathbb{E}_{x \sim \Pr(x)} \left(\mathbb{E}_{k \sim \Pr(k|x)} \left(\ell^{CE}(\hat{y}^k, y) + \lambda Cost^k \right) \right). \tag{2}$$

where λ is a hyperparameter that determines the weight given to additional costs associated with the model's predictions. It controls the emphasis placed on these costs compared to the cross-entropy loss. We will show later how to compute $\Pr(k|x)$, which is not a constant here.

Regularizing the Loss. We observed that using the loss given in Eq. 2, the model frequently favors a single output based on cost considerations. This tendency limits the model's capacity to generalize effectively due to the lack of diversity in output selection. To address this limitation, we introduce an additional term in the loss function, which is the unweighted cross-entropy, as follows:

$$\mathcal{L}(\theta; \mathcal{D}) = \mathbb{E}_{x \sim \Pr(x)} \left(\mathbb{E}_{k \sim \Pr(k|x)} \left(\ell^{CE}(\hat{y}^k, y) + \lambda Cost^k \right) + K \mathbb{E}_{k \sim \text{Uniform}(K)} \left(\ell^{CE}(\hat{y}^k, y) \right) \right). \tag{3}$$

Given that $Cost^k$ is a scalar value independent of the training parameters, the loss can be written as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} (\Pr(k|x) + 1) \left(\ell^{CE}(\hat{y}_i^k, y_i) + \lambda Cost^k \right). \tag{4}$$

Computing the Probability of Early Stopping. As discussed before, once the network achieves sufficient confidence, it may no longer require further consideration of subsequent exits. This rationale underlies our modeling of $\Pr(k|x_i)$, which is the likelihood for the sample x_i of exiting at exit k:

$$\Pr(k|x_i) = \begin{cases} P_i^k & \text{if } k = 1 \\ P_i^k \prod_{j=1}^{k-1} (1 - P_i^j) & \text{if } 1 < k < K \\ \prod_{j=1}^{K-1} (1 - P_i^j) & \text{if } k = K \end{cases}. \tag{5}$$

where P_i^k is the probability that $p_i^{(k)}$ is greater than the exit threshold.

Equation (5) defines this probability differently for three cases: the first exit ($k = 1$), the last exit ($k = K$), and all other exits in between. The probability of an input exiting at the first exit, denoted as $\Pr(k = 1|x)$, is equivalent to the probability of the input being correctly confident at that exit. Conversely, for the final output, the input will only exit if it has not exit at any of the preceding output. This scenario is captured by multiplying the probabilities of not exiting at each of the previous output. For all intermediate exits, the input will exit if two conditions are met: first, it must be correctly confident at the current exit, and second, it must not have been correctly confident at any of the preceding outputs. This situation is represented by the product of the probability of correct confident at the current exit and the probabilities of not exiting at each of the previous exits. In summary, this equation serves as a probabilistic decision-making framework, determining the optimal exit point for an input. It strikes a balance between the goal of achieving a correct confident and the objective of minimising the computational expense associated with processing the input.

Note that it can be readily confirmed that the summation $\sum_{k=1}^{K} \Pr(k|x) = 1$.

Estimating the Probability of Early Stopping at Train Time. Since we cannot know the values of the thresholds $\eta^{(k)}$ during training, it is not possible to calculate the probabilities P_i^k directly. We propose to replace this term by an action function A_i^k, which selects the outputs according to the multiple confidence scores $s_{i;\theta}^k$ and the backbone parameters represented by θ. This function, parameterised by ϕ, dictates the response of the model to the multiple confidence scores. This is achieved by employing a Multi-Layer Perceptron (MLP) on an aggregated confidence score, represented as $s_{i;\theta}^k$. Therefore, the probability $P_{i;\phi}^k = \sigma\left(\text{MLP}(s_{i;\theta}^k)\right)$, given the aggregated confidence score. σ is the sigmoid function.

The aggregation $s^k_{i;\theta}$, serves as a comprehensive measure of the model's certainty across various aspects, thereby providing a more robust estimate of the probability P^k_i. This approach allows for a more nuanced understanding of the model's performance, as it takes into account a variety of confidence scores, rather than relying on a single one. To accomplish this, we established an aggregation of confidence measures, which encompasses maximum confidence, maximum merging, and entropy. The formulation of these concepts is as follows (inspired by the work of Ilhan et al. [16]):

$$s^{l,max}_{i;\theta} = p^{(k)}_i, \tag{6}$$

$$s^{l,entropy}_{i;\theta} = \sum_{c\prime=0}^{C} \text{softmax}_{c'}\left(f^k(x_i)\right) \log(\text{softmax}_{c'}\left(f^k(x_i)\right)), \tag{7}$$

$$s^{l,margin}_{i;\theta} = p^{(k)}_i - \max_{c\prime \neq c} \text{softmax}_{c'}\left(f^k(x_i)\right). \tag{8}$$

In our practical implementation, we utilize Eq. (9) to regulate the smoothness of probability distributions within our model. This equation embodies the softmax operation with a modification introduced by the temperature parameter (T). Integrating T into the equation provides us with precise control over how probabilities are distributed among the exits, leveraging the input data x_i. The normalization step within the equation guarantees that the resulting probabilities sum up to 1, ensuring a valid probability distribution. In our experimental setup, we set the temperature parameter to $T = 0.5$. We find this temperature value works well through a grid search.

$$A^k_{i;\phi} = \frac{Pr(k|x_i)^{1/T}}{\sum_{j=1}^{K} Pr(j|x_i)^{1/T}} \tag{9}$$

Optimizing Φ and θ

The training of the multi-exit network involves optimizing two sets of parameters: θ, which are the parameters of the classification networks, and Φ, which are the parameters of the action function that estimates the probability for a sample to exit at a particular exit of the network. This constitutes a bi-level optimization problem, where we have two interconnected optimization problems, with one nested within the other.

In this bi-level optimization setup, the solution to the outer problem depends on the resolution of the inner problem. The process of minimizing the bi-level optimization can be described as follows:

$$\begin{aligned} &\min_{\phi} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(\ell^{CE}_{\theta^*}(\hat{y}^k_i, y_i) + \lambda \text{Cost}^k\right) \left(A^k_{i;\phi}(s^k_{i;\theta^*})\right) \\ &\text{s.t.} \min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \left(\ell^{CE}_{\theta}(\hat{y}^k_i, y_i) + \lambda \text{Cost}^k\right) \left(A^k_{i;\phi}(s^k_{i;\theta}) + 1\right) \end{aligned} \tag{10}$$

Our approach to bi-level optimization differs from that of [10], as we optimize the same loss function in both the inner and outer optimization stages, with

only the parameters varying. Now, directing attention to the derivatives of the objectives concerning the parameters θ, and ϕ, the derivatives are as follows:

$$\frac{\partial \mathcal{L}^{inner}}{\partial \theta} = \sum_{i=1}^{N}\sum_{k=1}^{K}\left(A_{i;\phi}^{k}(s_{i;\theta}^{k})+1\right)\frac{\partial \ell_{\theta}^{CE}(\hat{y}_i^k, y_i)}{\partial \theta} + \left(\ell_{\theta}^{CE}(\hat{y}_i^k, y_i)) + \lambda Cost^k\right)\frac{\partial A_{i;\phi}^{k}(s_{i;\theta}^{k})}{\partial \theta}, \tag{11}$$

$$\frac{\partial \mathcal{L}^{outer}}{\partial \phi} = \sum_{i=1}^{N}\sum_{k=1}^{K}(\ell_{\theta^*}^{CE}(\hat{y}_i^k, y_i) + \lambda Cost^k)\frac{\partial A_{i;\phi}^{k}(s_{i;\theta^*}^{k})}{\partial \phi}. \tag{12}$$

4 Experiments

This section presents the experimental validation of the proposed method on CIFAR-10/100 [19] and ImageNet [6] datasets, and provides comparisons with recent methods in the literature. Inline with the literature on the domain, we experiment our approach in the *budgeted batch classification mode* as well as in the *anytime classification mode* (see definitions in [15]).

On the Influence of λ. The classification network is trained using a cost function that manages a trade-off between budget and classification performance. Ideally, we would need to train a different network for each value of λ, i.e., for each level of budget allocated, which would require training and storing numerous networks to cover all the possible trade-offs. However, we have observed that it is possible to find a single value of λ that offers an acceptable compromise, regardless of the budget allocated for inference. This value, $\lambda = 2$, was estimated empirically through a performance analysis on the CIFAR-100 database. All the following experiments have been conducted using a single classification network learned with $\lambda = 2$.

On the Importance of Training the Classification Network with Budget-Aware Constraints. The main idea of this paper is that it is important to train the classification network while considering budget constraints, as opposed to methods that first train the classifier and then add techniques to refine the output rules. To validate this hypothesis, we conducted an experiment on the CIFAR databases, the results of which are part of Fig. 3. This figure compares the performance of our proposed method (called "MSDNet+Ours") with the same method when the classification network is first learned and then frozen (called "MSDNet frozen+Ours"). We observe that the gain is significant, on the order of +2% on CIFAR 100, across various budget levels.

On the Features Used for Estimating the Probability of Early Stopping at Train Time. As discussed in the previous section, during training we cannot know which example will output on which layer during inference.

Instead, we infer this information from certain features that we believe correlate with the early stopping probability, namely: 'confidence', 'margin', and 'entropy' (as described in Sect. 3). Table 1 shows the performance of the classifier for different budgets as a function of the information used to infer the probability of exiting. The best model across all budgets is obtained by using both the 'confidence' score and 'entropy' as inputs. In general, incorporating entropy into the input results in the highest score compared to using other features alone. This is the setting used in the following experiments.

Table 1. Accuracy on CIFAR-100 in budgeted batch classification, comparing different combinations of features used to determine the probability of exiting the network: 'confidence', 'margin', and 'entropy' (as described in Sect. 3).

Input	Top-1 Acc (%)			
	30M	50M	70M	90M
Confidence Only	70.84	76.39	79.32	80.08
Margin Only	71.49	76.07	78.73	79.6
Entropy Only	71.55	76.84	79.43	80.37
Confidence and Margin	71.63	76.48	78.91	79.73
Confidence and Entropy	**72.28**	76.81	**79.53**	**80.41**
Margin and Entropy	71.67	76.56	78.79	79.7
All	71.57	**76.89**	79.19	79.7

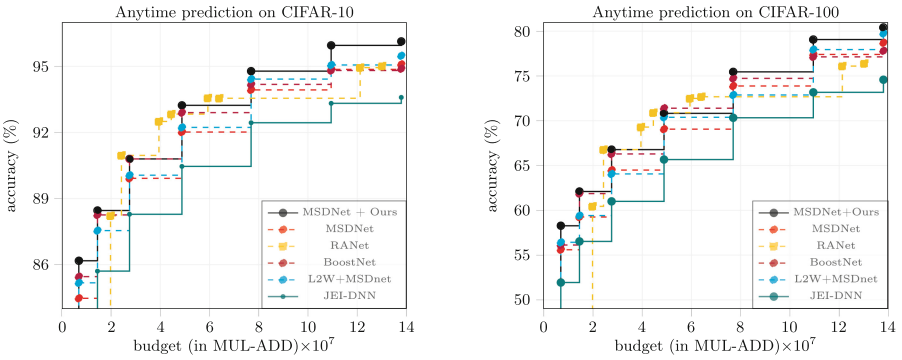


Fig. 2. Accuracy (top-1) of anytime prediction models as a function of computational budget on CIFAR-10 (top) and CIFAR-100 (bottom). Higher is better.

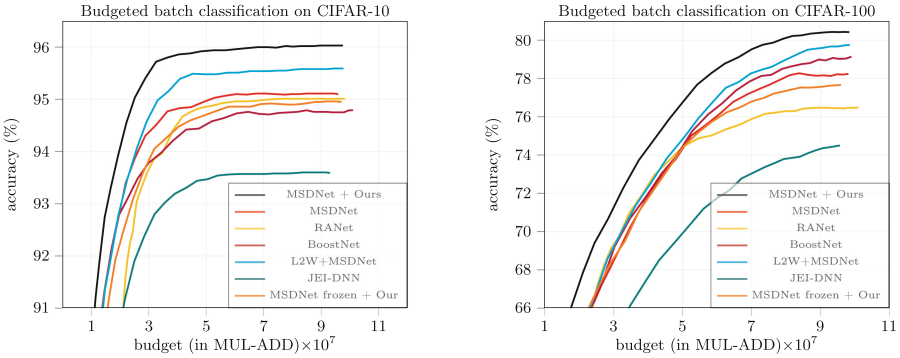


Fig. 3. Accuracy (top-1) of budgeted batch classification models as a function of average computational budget per image on CIFAR-10 (left) and CIFAR-100 (right).

Comparisons with State-of-the-Art Multi-exit Architectures. We compare our model to the best performing related works, namely MSDNet [15], RANet [33], BoostNet [35], L2W [10], and JEI-DNN [28]. In addition, we compare our results with those of post-hoc methods, specifically Calibrated-DNN proposed by Meronen et al. [26] and EENet proposed by Ilhan et al. [16].

For Anytime Prediction, our proposed training method coupled with MSDNet [15] surpasses the previous state-of-the-art for both CIFAR-10 and CIFAR-100 datasets, even for the first exits. The only exception is exit 3 on CIFAR-10, where RANet performs slightly better. Our method is depicted by the black curve in Fig. 2 entitled 'MDSNet+Ours'. On CIFAR-10, our method achieves an accuracy of 96.13% for its last exit with $137M$ FLOPs. For a budget of $27M$ FLOPs, we achieve an accuracy of 90.80% compared to RANet's 90.96% with a similar number of FLOPs. In comparison to JEI-DNN [28] trained under the same conditions, we achieve an improvement of approximately 2% to 3% in accuracy across all exits. On CIFAR-100, our method attains an accuracy of 80.43% with $137M$ FLOPs. Compared to the traditional training of MSDNet, our proposed method achieves an improvement of approximately 1% to 1.5% in accuracy, depending on the exit.

In the context of budgeted batch classification, our proposed method demonstrates superior performance compared to all other methods for both CIFAR-10 and CIFAR-100 datasets. For CIFAR-10, our method achieves a classification rate of 95.03% with a budget of only $25M$ FLOPs, which is significantly more efficient than MSDNet, requiring twice as many FLOPs to achieve the same level of performance. Furthermore, RANet and BoostNet require approximately $75M$ FLOPs to attain the same performance level as our method. To reach the same accuracy as JEI-DNN, which is approximately 93.4%, our method requires only $28M$ FLOPs, which is about 3× fewer FLOPs. For CIFAR-100, our method requires only $90M$ FLOPs to attain an accuracy of 80.43%, and only $58M$ FLOPs to achieve the same level of performance as MSDNet, which takes approximately 1.5× more FLOPs.

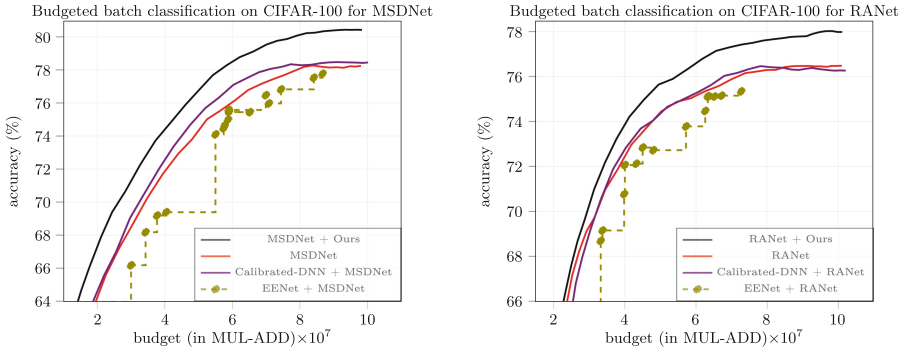


Fig. 4. Compare MSDNet and RANet using both the post-hoc method and our training approach on CIFAR 100 dataset. Higher is better.

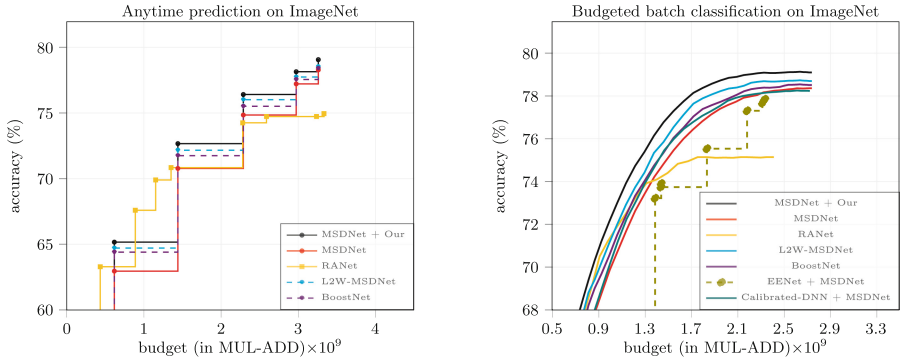


Fig. 5. Top-1 accuracy on ImageNet, plotted as a function of computational budget.

In addition to comparing our method with other approaches, we have also evaluated it against post-hoc methods, which involve applying post calibration on the scores. Figure 4 illustrates the comparison with two such methods, namely Calibrated-DNN [26] and EENet [16]. The figure presents two models, with MSDNet [15] on the left and RANet [33] on the right. Our proposed method demonstrates significant improvement in the performance of both architectures and surpasses the current state-of-the-art methods in both cases. In the scenario of MSDNet [15], Calibrated-DNN noticeably improves inference performance, particularly in budgets exceeding $60M$ Flops, although still falling short of our method's enhancement. When MSDNet [15] is combined with EENet [16], the resulting method exhibits inferior performance compared to the classic approach. As for RANet [33], our approach enhances its performance by a margin of 0.5% to 1.5% across all budgets. Notably, Calibrated-DNN achieves nearly equivalent performance to traditional training. Similarly, applying EENet [16] to RANet [33] yields poor result as classical training.

Finally, Fig. 5 reports the performance of the proposed method on the ImageNet dataset. Once again, our proposed method demonstrates superior performance compared to other state-of-the-art approaches, whether for Anytime Prediction or Budgeted Batch Classification. In Anytime Prediction, our method outperforms L2W-MSDNet and BoostNet, two approaches that bear the closest resemblance to ours. In the realm of Budgeted Batch Classification, our primary focus, our approach achieves a slightly higher accuracy of around 0.5% to 1% compared to alternative methods. Noteworthy is our method's efficiency, requiring only 1.95×10^9 Flops to achieve peak accuracy, representing a 25% reduction in Flops compared to L2W-MSDNet. Regarding BoostNet, our method achieves the maximum performance reached by BoostNet, approximately 78.54%, with only 1.87×10^9 Flops, marking a reduction of approximately 30% in Flops.

5 Conclusions

This paper introduces a training approach for Early Exit Neural Networks (EENN) specifically designed for budgeted classification tasks. The primary goal of our method is to align the behavior of the training and inference steps. By enhancing traditional Early Exit models with the integration of the forward cost and a network that calibrates sample difficulty, we achieve improved classification accuracy while respecting computational constraints. Extensive evaluations on CIFAR-10, CIFAR-100, and ImageNet benchmarks demonstrate that our approach sets a new state-of-the-art for budgeted classification, consistently outperforming existing early exit strategies.

Acknowledgments. Research reported in this paper was supported by the ANR under award number ANR-19-CHIA-0017 and was performed using computing resources of CRIANN.

References

1. Addad, Y., Lechervy, A., Jurie, F.: Multi-exit resource-efficient neural architecture for image classification with optimized fusion block. In: ICCV Workshops (2023)
2. Agarwal, C., D'souza, D., Hooker, S.: Estimating example difficulty using variance of gradients. In: CVPR (2022)
3. Bi, Y., Xue, B., Mesejo, P., Cagnoni, S., Zhang, M.: A survey on evolutionary computation for computer vision and image analysis: past, present, and future trends. IEEE Trans. Evol. Comput. **27**(1), 5–25 (2023)
4. Chen, M., et al.: CF-ViT: a general coarse-to-fine method for vision transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37 (2022)
5. Chen, T., et al.: TVM: end-to-end optimization stack for deep learning. arXiv preprint arXiv:1802.04799 (2018)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)

7. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: GhostNet: more features from cheap operations. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020
8. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **44**(11), 7436–7456 (2022). https://doi.org/10.1109/TPAMI.2021.3117837
9. Han, Y., et al.: Dynamic perceiver for efficient visual recognition, June 2023
10. Han, Y., et al.: Learning to weight samples for dynamic early-exiting networks. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022, Part XI. LNCS, vol. 13671, pp. 362–378. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20083-0_22
11. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
12. Hedderich, M.A., Lange, L., Adel, H., Strötgen, J., Klakow, D.: A survey on recent approaches for natural language processing in low-resource scenarios. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2545–2568. Association for Computational Linguistics, Online, June 2021
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015)
14. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications, April 2017. arXiv:1704.04861 [cs]
15. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.: Multi-scale dense networks for resource efficient image classification. In: International Conference on Learning Representations (2018)
16. Ilhan, F., et al.: Adaptive deep neural network inference optimization with EENet. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 1373–1382, January 2024
17. Jacob, B., et al.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018
18. Jung, S., et al.: Learning to quantize deep networks by optimizing quantization intervals with task loss. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019
19. Krizhevsky, A.: Learning multiple layers of features from tiny images, pp. 32–33 (2009)
20. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 25 (2012)
21. Lan, x., Zhu, X., Gong, S.: Knowledge distillation by on-the-fly native ensemble. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc. (2018)
22. Laskaridis, S., Kouris, A., Lane, N.D.: Adaptive inference through early-exit networks: design, challenges and directions. In: Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning (2021)
23. Lee, H., Lee, J.S.: Students are the best teacher: exit-ensemble distillation with multi-exits, April 2021. https://doi.org/10.48550/arXiv.2104.00299

24. Lee, H., Lee, J.S.: Rethinking online knowledge distillation with multi-exits. In: Wang, L., Gall, J., Chin, T.J., Sato, I., Chellappa, R. (eds.) ACCV 2022, vol. 13846, pp. 408–424. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-26351-4_25
25. Li, H., Zhang, H., Qi, X., Yang, R., Huang, G.: Improved techniques for training adaptive deep networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019
26. Meronen, L., Trapp, M., Pilzer, A., Yang, L., Solin, A.: Fixing overconfidence in dynamic neural networks, April 2023. arXiv:2302.06359 [cs]
27. Prabhavalkar, R., Hori, T., Sainath, T.N., Schlüter, R., Watanabe, S.: End-to-end speech recognition: a survey, March 2023. arXiv:2303.03329 [cs, eess]
28. Regol, F., Chataoui, J., Coates, M.: Jointly-learned exit and inference for a dynamic neural network. In: The Twelfth International Conference on Learning Representations (ICLR) (2024)
29. Teerapittayanon, S., McDanel, B., Kung, H.T.: BranchyNet: fast inference via early exiting from deep neural networks. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 2464–2469 (2016)
30. Thakkar, M., Thakkar, M.: Introduction to core ml framework. Beginning Machine Learning in iOS: CoreML Framework (2019)
31. Wang, Y., Huang, R., Song, S., Huang, Z., Huang, G.: Not all images are worth 16x16 words: dynamic transformers for efficient image recognition. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
32. Wang, Y., Lv, K., Huang, R., Song, S., Yang, L., Huang, G.: Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020
33. Yang, L., Han, Y., Chen, X., Song, S., Dai, J., Huang, G.: Resolution adaptive networks for efficient inference. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020
34. Yang, L., et al.: CondenseNet V2: sparse feature reactivation for deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3569–3578, June 2021
35. Yu, H., Li, H., Hua, G., Huang, G., Shi, H.: Boosted dynamic neural networks. In: Williams, B., Chen, Y., Neville, J. (eds.) Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, pp. 10989–10997. AAAI Press (2023)
36. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018
37. Zhu, M., et al.: Dynamic resolution network. In: Advances in Neural Information Processing Systems, vol. 34, pp. 27319–27330. Curran Associates, Inc. (2021)

An Evolutionary Search-Based Operator Fusion Method with Binary Representation for Deep Learning Inference Acceleration

Yu Yang[1,2], Boyu Diao[1(✉)], Hangda Liu[1,2], Qiyun Chen[1,2], Qi Wang[1], and Yongjun Xu[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
{yangyu211,chenqiyun21}@mails.ucas.ac.cn,
{diaoboyu2012,liuhangda21s,wangqi08,xyj}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beijing, China

Abstract. The deployment of deep neural network (DNN) models on various devices presents a significant challenge due to the diverse characteristics of deployment environments. Operator fusion, which enhances DNN model inference efficiency by combining multiple operators into a single one, is critical in this context. However, most deep learning compilers use a uniform fusion scheme, failing to consider environment-specific optimizations.

We propose a novel approach to determine optimal operator fusion schemes tailored to different deployment environments. By representing fusion schemes with fixed-length binary sequences and using model inference time as feedback, we apply an evolutionary search strategy to identify effective fusion schemes. Our experiments with multiple DNN models across diverse environments show substantial performance gains over static fusion schemes used by TVM and DNNFusion. Specifically, we observed an average performance improvement of 17% on the NVIDIA Tesla V100 GPU, with a maximum improvement of 79%. On the NVIDIA Orin NX GPU, our method achieved an average improvement of 13%. On CPUs, the average performance increase was 25%, peaking at 44%. These results underscore the effectiveness of our method in enhancing DNN model inference across various deployment environments.

Keywords: Operator Fusion · Evolutionary Search · Deep Learning Compilation

1 Introduction

Deep learning, a specialized branch of machine learning, has experienced swift progress with the integration of multi-layer neural networks and large-scale datasets. This area of research has not only driven progress in established fields like computer vision and natural language processing but has also achieved

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 32–45, 2025.
https://doi.org/10.1007/978-3-031-78172-8_3

notable advancements in emerging domains such as model interpretability [12,25], compression [7,24], and the prevention of technology misuse [23]. As deep learning models grow in complexity and performance, they require increasingly more computational resources. This surge in demand poses a substantial challenge for both research and practical applications, particularly in terms of efficient deployment.

Operator fusion [4,5] is a crucial technique for accelerating model inference. It merges multiple operators into a single equivalent operator, thereby reducing the number of memory reads and writes for intermediate results and enhancing the execution efficiency of DNN models [11]. Since operator fusion does not alter the parameters of the DNN model, the output remains consistent before and after optimization, ensuring that the model's accuracy is not compromised. In the deep learning compiler optimization process, the operator fusion scheme is first determined at the graph level, followed by the operator level generating tensor programs for each fused operator, tailored to various devices. Therefore, the optimization effect of operator fusion is not only influenced by the fusion scheme but also closely related to the deployment environments [28].

Synergistic optimization that combines operator fusion schemes with tensor program generation holds the key to maximizing the benefits of operator fusion, thus boosting the inference efficiency of DNN models. While certain frameworks rely on traditional algorithms for devising fusion schemes. This reliance, however, often results in settling for local optima, which means the full optimization potential of operator fusion is not realized. Exploring innovative strategies that circumvent these drawbacks is essential for enhancing optimization.

We introduce a novel method to identify the optimal operator fusion scheme for various deployment environments. Initially, we represent operator fusion schemes as fixed-length binary sequences using a DAG Analyzer module. This setup enables the application of evolutionary algorithms to match the most suitable fusion schemes with specific deployment environments, using the model's inference time as a fitness measure. For broader compatibility with other optimization techniques, the Sequence Decoder transforms these binary sequences into a more universally applicable format during the compilation phase.

Experimental results indicate that our approach yields a performance improvement ranging from 7% to 44% on server-level CPU, averaging 25%, when compared to the static fusion schemes of TVM and DNNFusion. For NVIDIA Tesla V100 GPU [16], we observed improvements between 7% and 79%, with an average of 17%. On NVIDIA Orin NX GPU [17], improvements varied from 5% to 34%, with an average of 13%.

This paper makes the following contributions:

1. We represent the operator fusion scheme of DNN models as fixed-length binary sequences. This novel representation creates an extensive solution space for operator fusion, simplifying the identification of invalid schemes and allowing for easy shifts among different fusion strategies.

2. We employ an evolutionary search strategy to identify the optimal operator fusion schemes, specifically tailored to each deployment environment. This approach effectively avoids the pitfalls of local optima.
3. Through evaluations conducted on a diverse array of models and computing platforms, we have demonstrated that our approach significantly boosts the inference performance of DNN models. This evidence underscores our method's ability to enhance computational efficiency across various deployment environments.

2 Related Works

Research on operator fusion schemes is currently divided into two distinct approaches: classification-based and algorithm generation-based.

The classification-based approach organizes operators based on specific criteria and outlines rules for their fusion. For example, TVM [3] categorizes operators into four types: injective, reduction, complex-out-fusable, and opaque, and then identifies three scenarios where these can be combined through computation graph traversal. DNNFusion [14] takes a different angle by classifying operators into five categories based on the relationship between their input and output tensors, allowing for more versatile fusion rules. This approach facilitates both forward and backward fusion from simple operators within the computation graph. The Apollo [26] system advances this concept further by implementing a dynamic rule-updating mechanism to better match the nuances of different hardware platforms, thus optimizing fusion strategy effectiveness and adaptability.

Conversely, the algorithm generation-based approach employs models and algorithmic optimizations to pinpoint the most effective fusion strategy. FusionStitching [29] assesses fusion strategy benefits through metrics like decreased memory access volume and reduced CPU-GPU context switching, utilizing a beam search algorithm to identify the optimal fusion scheme. Optimus [2], on the other hand, creates a memory access cost model and applies dynamic programming to discover fusion strategies that lower memory access volume the most.

These studies typically employ static fusion schemes or conduct searches within constrained solution spaces, often leading to local optima. To address this issue, we construct a comprehensive solution space and employ evolutionary search methods to enhance search efficiency and avoid the pitfall of local optima.

3 Background

Operator fusion focuses on optimizing the computational graph $G = (V, E)$, derived from deep learning models. This graph is a Directed Acyclic Graph (DAG) consisting of nodes V and directed edges E. In this context, each node represents a tensor operation, and the directed edges represent the data dependencies between operators. The process of operator fusion involves combining

multiple operators into a singular new fused operator, similar to partitioning the graph. This optimization technique maintains the correctness of data dependencies, guaranteeing that the modified, fused graph is functionally equivalent to the original.

Traditionally, operator fusion schemes are considered as a series of disjoint node sets $\mathcal{F} = \{V_1, V_2, \ldots, V_k\}$, satisfying specific conditions to ensure that there are direct or indirect data dependency relationships among nodes within each fused operator, and collectively covering all nodes in the original computational graph. However, as illustrated in Fig. 1, this representation introduces dual uncertainties in the number of node sets and the size of each set. This complexity hinders the fluidity of transitions between different fusion schemes, due to the need of multiple operations on sets and nodes, such as moving nodes between sets or merging and splitting sets. Such constraints limit the application of effective search algorithms within the operator fusion solution space.

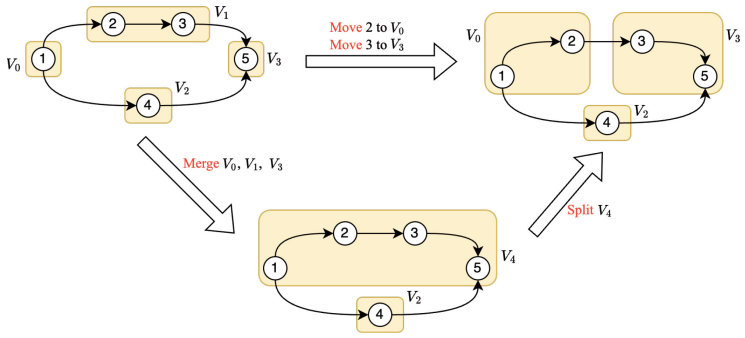


Fig. 1. The transition between fusion schemes lacks flexibility, involving three distinct operations: Move, Merge, and Split.

Therefore, seeking a more efficient representation and search method for operator fusion schemes, to overcome the shortcomings of existing methods, holds significant research significance and practical value for the optimization of deep learning compilers.

4 Method

4.1 Overview

As shown in Fig. 2, our methodology consists of two primary components: the DAG Analyzer and the Sequence Decoder. The DAG Analyzer examines the computational graph to identify information pertinent to the fusion scheme's fixed-length binary sequence representation. Utilizing this data, an evolutionary search explores the solution space. For every fusion scheme, the Sequence Decoder module adjusts and interprets it, leading to the creation of a respective tensor program. With the aim of reducing inference time, we evaluate the efficiency of fusion schemes by the execution time of the tensor programs.

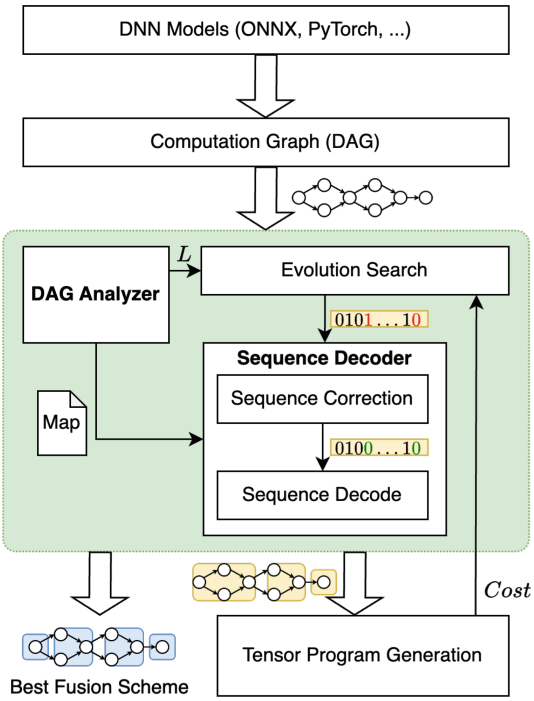


Fig. 2. Overview. The green part represents our proposed approach's composition, yellow highlights the fusion schemes identified during the evolutionary search, and blue marks the final fusion scheme derived by the evolutionary algorithm. (Color figure online)

4.2 DAG Analyzer

Key Insight. In deep learning computational graphs, directed edges serve a dual purpose: depicting data flow paths between operators and revealing dependencies where the output tensor of one operator becomes the input for another. We categorize these dependencies into two main types: intra-fusion (within the same fused operator) and inter-fusion (across different fused operators), as illustrated in Fig. 3. This bifurcation is crucial for the construction of operator fusion schemes, essentially requiring a clear classification of each directed edge's dependency type within the graph.

In exploring operator fusion schemes within a computation graph $G = (V, E)$, we employ a representation that precisely captures the type of dependency relationships. We associates each edge $e_i \in E$ with a binary value b_i, where $b_i = 1$ denotes intra-fusion dependencies and $b_i = 0$ signifies inter-fusion dependencies. Consequently, the entire fusion scheme is encoded as a binary sequence $S = b_1 b_2 ... b_{|E|}$, where the length of the sequence is equal to the number of edges. This representation not only simplifies the depiction of fusion schemes but also significantly facilitates their comparison, optimization, and automation.

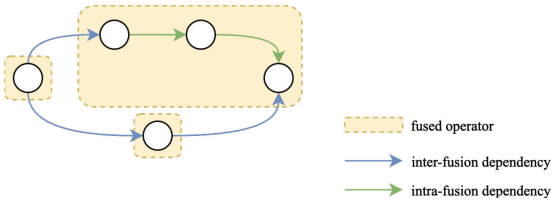


Fig. 3. Dependency categories in operator fusion.

The completeness and uniqueness of this representation are based on the following considerations:

For any computation graph $G = (V, E)$ and its operator fusion scheme $P = \{G_1, G_2, ..., G_n\}$, each edge $e_i \in E$ can be denoted as (v_x, v_y). That is, v_x and v_y either reside within the same subgraph G_k (i.e., $v_x, v_y \in G_k$, where $b_i = 1$) or in distinct subgraphs (i.e., $v_x \in G_p, v_y \in G_q$, and $p \neq q$, with $b_i = 0$). Hence, a unique binary bit b_i corresponds to each edge in E, forming the binary sequence $S = b_1 b_2 ... b_{|E|}$ that is uniquely associated with P.

Conversely, for any given binary sequence S of length $|E|$, the association of nodes to each edge e_i is determined by the bit b_i in S. Specifically, if $b_i = 0$, the nodes of e_i belong to different subgraphs. If $b_i = 1$, they are part of the same subgraph. Given that the edge set E defines all node connections in G, this method enables the unique reconstruction of the operator fusion scheme P.

By distinctly classifying each edge's dependency type as either intra-fusion or inter-fusion and using a binary sequence to uniquely represent these relationships, we not only streamline the visualization and handling of fusion schemes, but also ensure the representation's completeness and uniqueness. This provides a solid foundation for further research into operator fusion.

Search Space Pruning. In the construction of current deep learning models, operators performing simple logical operations, such as addition and ReLU, play a crucial role. These operators are characterized by their one-to-one mapping between the output tensor and input elements, executing a simple operation on each input element independently and preserving the tensor shape. Due to this characteristic, such operators do not require complex optimization techniques (e.g., tiling) and can be fused with other operators with data dependencies, leading to performance improvements.

By fusing simple operators directly with their producer operators, we fix these direct dependencies in the computation graph and eliminate these dependency bits from the binary sequence representation, effectively pruning the solution space. Taking the common structure in YOLOv4 [1] as an example, as shown in Fig. 4, pruning reduces the solution space from 2^6 to 2^3, focusing on fusion scenarios with significant potential for performance improvement, instead of exploring all possible fusions.

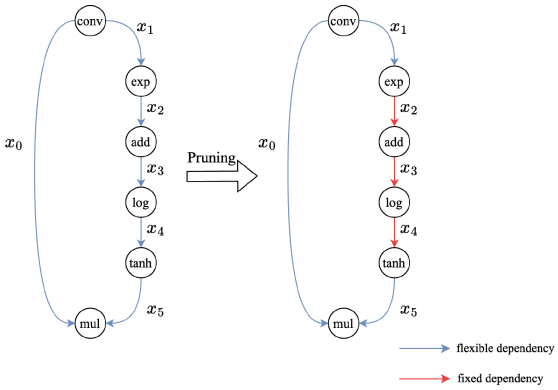


Fig. 4. Pruning demo in YOLOv4. Pruning reduces the search space from 2^6 to 2^3.

Subsequence Sharing. Deep learning architectures often feature the use of repeated modular units, such as the residual blocks found in ResNet-18 [6]. These modules are arranged in a deliberate sequence to facilitate complex data processing. However, when these architectural structures are encoded into fixed-length binary sequences for operator fusion, redundancy becomes a significant issue due to the repetition of these modules. This is particularly true when identical modules at different positions in the model lead to longer than necessary sequences, compromising the efficiency of representation.

Recognizing the need for a more efficient method that accounts for the repetition of modules under identical execution conditions, we present a sequence sharing strategy based on subgraph isomorphism. This method seeks to detect and assign functionally identical modules within the computational graph to identical segments of the binary sequence, thereby enhancing representation efficiency.

4.3 Sequence Decoder

Sequence Repair. As shown in Fig. 5, the solution space contains many illegal solutions, which result in loop data dependency within the computation graph. Our analysis under the binary sequence representation reveals that the reason of these illegal solutions leading to data loops is the presence of both "reachable" paths (where all edges along the path have a value of 1, indicating intra-fusion dependency) and "unreachable" paths (where there is at least one edge with a value of 0, indicating inter-fusion dependency) between a node and its post-dominator.

Based on this analysis, we verify the legality of each fusion scheme encountered during evolution search. For nodes that cause a fusion scheme to be illegal, we transform the scheme into a legal, executable one by setting the values of all incoming edges to their post-dominators to 0.

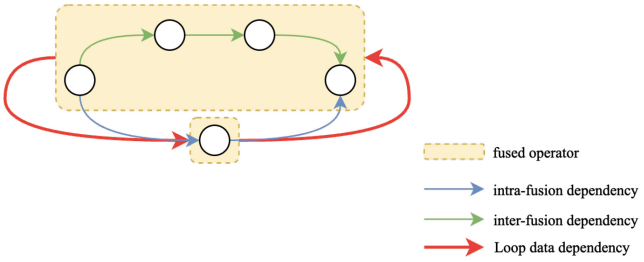


Fig. 5. Loop data dependency.

Sequence Decode. To be compatible with specific tensor program generation methods, we convert the binary sequence representation of a fusion scheme into a traditional form. As described in Algorithm 1, it iteratively propagates through each node in the computation graph based on the fusion relationships with producers and consumers. Each node propagates only once until it reaches the boundary of the fusion operators. Ultimately, each fusion operator is represented as a set of nodes, constituting the traditional representation.

Algorithm 1. Decode fusion scheme into node sets.

Input: $G = (V, E)$, computation graph.
Input: S, a fusion scheme represented by binary sequence.
Input: $Mapper$, a function mapping each edge $e \in E$ to an index of S.
Output: Vs, a fusion scheme represented by node sets.

```
Initialize Vs as an empty dictionary.
Initialize visited as an empty set.
for each node n in V do
  tempV ← Vs[n] if n in Vs else initialize ∅ for Vs[n]
  Initialize queue q with n.
  while q is not empty do
    v ← q.pop()
    if v in visited then
      Continue to the next iteration of the loop.
    end if
    Add v to visited.
    fusedProducers ← getFusedProducers(G, S, Mapper, v)
    fusedConsumers ← getFusedConsumers(G, S, Mapper, v)
    for each fusedNode in fusedProducers + fusedConsumers do
      Vs[fusedNode] ← tempV
      Insert fusedNode into tempV.
      q.push(fusedNode) if fusedNode not in visited.
    end for
  end while
end for
return Vs
```

4.4 Evolutionary Search

Fixed-length binary sequences are used to represent operator fusion schemes in deep learning models, enabling the definition of a comprehensive solution space. We utilize evolutionary search to efficiently explore this space. For fitness evaluation, all operator fusion schemes in the population, encoded as fixed-length binary sequences, are decoded through the Sequence Decoder module. Each individual generates a tensor program, whose performance is tested in a specified deployment environment, with inference time serving as the fitness value.

We employ a tournament selection strategy to choose a subset of individuals for the next generation, followed by crossover and mutation operations. The individual with the minimum fitness value wins the tournament. Additionally, we implement an elite selection strategy, where the individual with the smallest fitness value is directly transferred to the next generation without selection, crossover, or mutation, enhancing convergence speed. If the inference time of the optimal individual remains unchanged after 20 iterations, we consider the search process converged.

5 Experiments

5.1 Experimental Settings

Models. In addressing the diverse requirements of deep learning tasks, we adopted a comprehensive selection strategy for model evaluation. Initially, for image classification, we chose a range of representative models known for their outstanding performance in image recognition. This set includes DenseNet [8], GoogleLeNet [20], InceptionV2 [10], and members of the ResNet series, specifically ResNet-18 and ResNet-152. Recognizing scenarios where computational resources are limited, we also included a selection of lightweight models designed for efficiency without significantly compromising performance. These models include EfficientNet-lite4 [21], MobileNetV2 [19], ShuffleNetV2 [13], and SqueezeNet [9], all of which offer reduced computational demands. For object detection, we chose the YOLOv4 model due to its effectiveness. Finally, in the realm of Natural Language Processing (NLP), we selected BERT-Small [22] and GPT-2 [18].

Tensor Program Generation. We employed the Ansor [27] auto-tuning framework to generate tensor programs for fused operators. Ansor is an advanced auto-tuning system capable of optimizing a diverse range of fused operators. Moreover, its integration with the deep learning compiler TVM is very tight, providing a set of user-friendly interfaces.

Environment Setup. We evaluated our approach on server-level CPUs and GPUs, as well as on GPUs in edge device. The configurations of the experimental environment are shown in Table 1.

Table 1. Experimental Environment Setup.

Metric	Cloud Server	Edge Device
CPU	Intel Xeon Gold 6248R	ARMv8 Processor rev 1 (v8l)
GPU	NVIDIA Tesla V100	NVIDIA Orin NX GPU
CUDA Driver	11.4	11.4
LLVM Version	15.0.5	16.0.0
Operating System	Ubuntu 18.04.06 LTS	Ubuntu 20.04

Implementation Details. We have integrated our methodology into TVM. In setting up the evolutionary algorithm, we start with a randomly initialized population. Fusion schemes that encounter execution failures for unspecified reasons are assigned a significantly high inference time, allowing for their exclusion via the selection. For models characterized by lengthy binary sequences, we improve search efficiency by implementing a multi-point crossover technique. To guarantee the search's eventual convergence, we employ an elitist reservation strategy. Furthermore, to reduce resource waste, an early stopping mechanism is activated.

5.2 End-to-End Model Performance

We evaluated our approach across various computing platforms. Our goal was to measure the improvements in end-to-end model inference performance relative to TVM's default strategy. For further comparison, we also considered the DNNFusion, specifically focusing on its fusion scheme rather than the entire framework. The results of these evaluations are illustrated in Fig. 6 and Fig. 7 for the CPU and GPU on cloud server, respectively, and in Fig. 8 for the GPU on edge device (In the results, the vertical axis "Relative Performance" refers to the relative improvement in inference efficiency compared to the TVM).

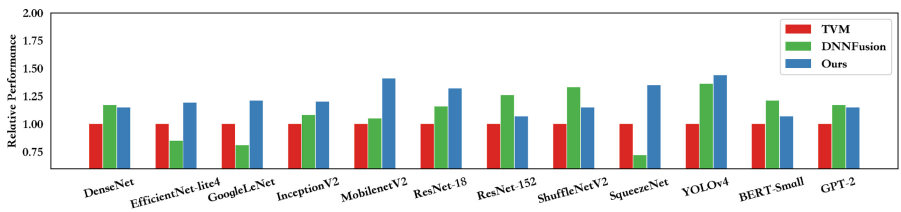


Fig. 6. Performance of different DNN models on CPU (y-axis: relative performance compared to TVM).

The horizontal axis represents various DNN models, and the vertical axis illustrates the performance improvements relative to TVM's fusion scheme. Our method outperformed the strategies of both TVM and DNNFusion in all tests on

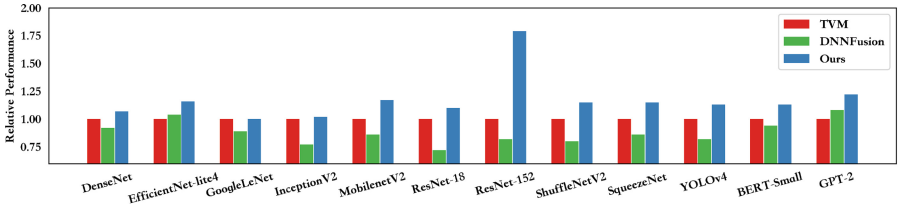


Fig. 7. Performance of different DNN models on NVIDIA Tesla V100 GPU (y-axis: relative performance compared to TVM).

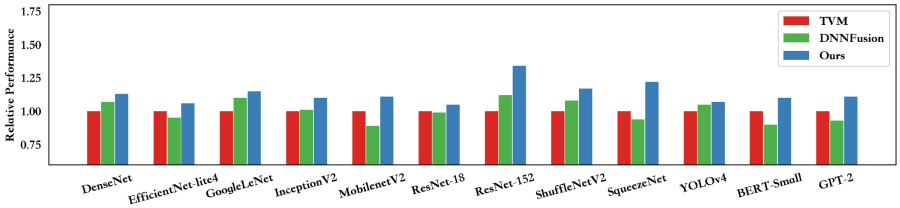


Fig. 8. Performance of different DNN models on NVIDIA Orin NX GPU (y-axis: relative performance compared to TVM).

the NVIDIA Tesla V100 GPU, achieving an average efficiency improvement of 17% and peaking at 79%. On the NVIDIA Orin NX GPU, our approach realized a 13% average efficiency improvement over TVM. Similarly, on CPUs, it achieved an average efficiency gain of 25%. These results affirm the broad applicability and superior effectiveness of our approach.

5.3 Memory Analysis

To delve deeper into the efficacy of our approach on computing platforms, we utilized the ResNet-18 model as our primary neural network architecture. We employed NVIDIA Nsight Compute [15] for a detailed assessment of GPU memory bandwidth utilization, comparing our approach against the fusion schemes of TVM and DNNFusion. Figure 9 highlights our findings, showing that our approach enhances memory bandwidth utilization by 13% over TVM and 20% over DNNFusion. This increase in memory bandwidth utilization significantly cuts down on data transfer overhead, leading to greater efficiency in compute core performance (In the results, the vertical axis "Relative Performance" refers to the relative improvement in memory bandwidth utilization compared to the TVM).

5.4 Compilation Time

Our approach, based on evolutionary search, led to significantly longer search times within the solution space compared to the static fusion schemes used by

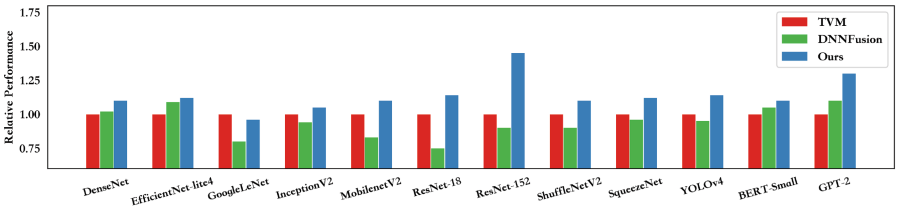


Fig. 9. Memory bandwidth utilization of different DNN models on NVIDIA Tesla V100 GPU (y-axis: relative performance compared to TVM).

TVM. During testing on an NVIDIA Tesla V100 GPU, our method's average compilation time was 17.3 times that of TVM. On CPU, the compilation time increased to 27.1 times, and on the NVIDIA Orin NX GPU, it extended to 19.7 times longer. The increased compilation times are mainly due to the necessity of evaluating various fusion schemes during the evolutionary search. Despite the extended compilation duration, the time investment is justified, as DNN model deployment is generally a "once-for-all" process (In the results, the vertical axis "Relative Performance" refers to the relative improvement in compilation efficiency compared to the TVM).

5.5 Case Study

In this study, we illustrate the variation in the inference time of EfficientNet-lite4 on NVIDIA Tesla V100 GPU through the iterative process of evolution search. As shown in Fig. 10, the performance of the initial population was inferior to the fusion schemes of TVM and DNNFusion. However, a significant and rapid decline in inference time was observed in the first 7 generations, with a superior fusion scheme emerging after the 5th iteration that surpassed those of TVM and DNNFusion. From the 7th to the 24th generation, the performance saw a

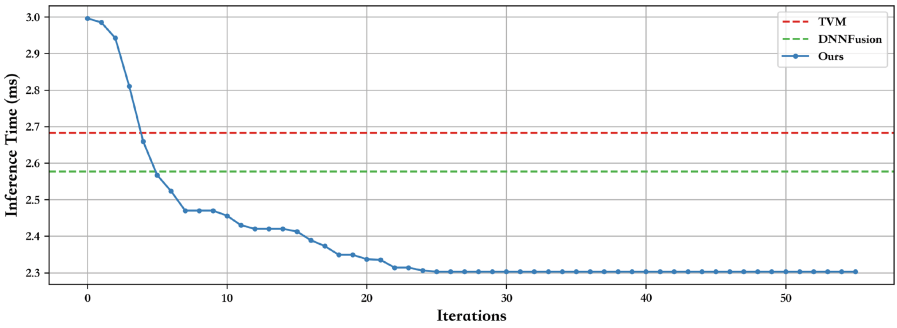


Fig. 10. EfficientNet-lite4 inference time reduction with evolutionary algorithm iterations.

gradual, stepwise improvement and ultimately converged in the 25th generation, achieving a 16% performance increase compared to TVM.

6 Conclusion

Our approach utilizes evolutionary search to identify superior operator fusion schemes tailored to specific deployment environments. To overcome the inflexibility of traditional representations of fusion schemes, we adopt fixed-length binary sequences to depict operator fusion schemes, thereby delineating the entire solution space. Experimental validation has shown that our method significantly enhances inference speeds across a variety of DNN models compared to static fusion schemes. Notably, we achieved an average inference acceleration of 17% on server-level CPU and 25% on NVIDIA Tesla V100 GPU. Additionally, a 13% increase in inference speed was observed on NVIDIA Orin NX GPU. These results highlight the effectiveness of our methodology in boosting model inference speeds across different computing platforms.

References

1. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.: Yolov4: optimal speed and accuracy of object detection. Cornell University - arXiv, Cornell University - arXiv, April 2020
2. Cai, X., Wang, Y., Zhang, L.: Optimus: an operator fusion framework for deep neural networks. ACM Trans. Embed. Comput. Syst. **22**(1), 1–26 (2022)
3. Chen, T., et al.: {TVM}: an automated {End-to-End} optimizing compiler for deep learning. In: 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2018), pp. 578–594 (2018)
4. Dai, L., Gong, L., An, Z., Xu, Y., Diao, B.: Sketch-fusion: a gradient compression method with multi-layer fusion for communication-efficient distributed training. J. Parallel Distrib. Comput. **185**, 104811 (2024)
5. Dong, D., Jiang, H., Diao, B.: AKGF: automatic kernel generation for DNN on CPU-FPGA. Comput. J. **67**, 1619–1627 (2023)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
7. Huang, L., Zeng, Y., Yang, C., An, Z., Diao, B., Xu, Y.: eTag: class-incremental learning with embedding distillation and task-oriented generation. arXiv preprint arXiv:2304.10103 (2023)
8. Iandola, F., Moskewicz, M., Karayev, S., Girshick, R., Darrell, T., Keutzer, K.: DenseNet: implementing efficient convnet descriptor pyramids. arXiv preprint arXiv:1404.1869 (2014)
9. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
10. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)

11. Li, M., et al.: The deep learning compiler: a comprehensive survey. IEEE Trans. Parallel Distrib. Syst. **32**(3), 708–727 (2020)
12. Li, Q., Zhang, Z., Diao, B., Xu, Y., Li, C.: Towards understanding the effect of node features on the predictions of graph neural networks. In: Pimenidis, E., Angelov, P., Jayne, C., Papaleonidas, A., Aydin, M. (eds.) ICANN 2022. LNCS, vol. 13530, pp. 706–718. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15931-2_58
13. Ma, N., Zhang, X., Zheng, H.-T., Sun, J.: ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 122–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_8
14. Niu, W., Guan, J., Wang, Y., Agrawal, G., Ren, B.: DNNFusion: accelerating deep neural networks execution with advanced operator fusion. In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 883–898 (2021)
15. NVIDIA Corporation: NVIDIA nsight compute. https://developer.nvidia.com/nsight-compute. Accessed 07 Apr 2024
16. NVIDIA Corporation: NVIDIA tesla v100. https://www.nvidia.com/en-gb/data-center/tesla-v100/. Accessed 14 Mar 2024
17. NVIDIA Corporation: NVIDIA Jetson Orin NX card (2023). https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-orin/. Accessed 24 Jan 2024
18. Radford, A., et al.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
19. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510–4520 (2018)
20. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
21. Tan, M., Le, Q.: EfficientNet: rethinking model scaling for convolutional neural networks
22. Turc, I., Chang, M.W., Lee, K., Toutanova, K.: Well-read students learn better: on the importance of pre-training compact models. arXiv preprint arXiv:1908.08962v2 (2019)
23. Wang, Q., et al.: How to prevent malicious use of intelligent unmanned swarms? Innov. **4**(2), 100396 (2023)
24. Xu, J., Diao, B., Cui, B., Yang, K., Li, C., Hong, H.: Pruning filter via gaussian distribution feature for deep neural networks acceleration. In: 2022 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2022)
25. Yao, K., Wang, J., Diao, B., Li, C.: Towards understanding the generalization of deepfake detectors from a game-theoretical view. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2031–2041 (2023)
26. Zhao, J., et al.: Apollo: automatic partition-based operator fusion through layer by layer optimization. Proc. Mach. Learn. Syst. **4**, 1–19 (2022)
27. Zheng, L., et al.: Ansor: generating {High-Performance} tensor programs for deep learning. In: 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2020), pp. 863–879 (2020)
28. Zheng, Z., et al.: AStitch: enabling a new multi-dimensional optimization space for memory-intensive ML training and inference on modern SIMT architectures. In: Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 359–373 (2022)
29. Zheng, Z., et al.: FusionStitching: boosting memory intensive computations for deep learning workloads. arXiv preprint arXiv:2009.10924 (2020)

SemFaceEdit: Semantic Face Editing on Generative Radiance Manifolds

Shashikant Verma and Shanmuganathan Raman(✉)

CVIG Lab, Indian Institute of Technology Gandhinagar, Ahmedabad, India
{shashikant.verma,shanmuga}@iitgn.ac.in

Abstract. Despite multiple view consistency offered by 3D-aware GAN techniques, the resulting images often lack the capacity for localized editing. In response, generative radiance manifolds emerge as an efficient approach for constrained point sampling within volumes, effectively reducing computational demands and enabling the learning of fine details. This work introduces SemFaceEdit, a novel method that streamlines the appearance and geometric editing process by generating semantic fields on generative radiance manifolds. Utilizing latent codes, our method effectively disentangles the geometry and appearance associated with different facial semantics within the generated image. In contrast to existing methods that can change the appearance of the entire radiance field, our method enables the precise editing of particular facial semantics while preserving the integrity of other regions. Our network comprises two key modules: the Geometry module, which generates semantic radiance and occupancy fields, and the Appearance module, which is responsible for predicting RGB radiance. We jointly train both modules in adversarial settings to learn semantic-aware geometry and appearance descriptors. The appearance descriptors are then conditioned on their respective semantic latent codes by the Appearance Module, facilitating disentanglement and enhanced control. Our experiments highlight SemFaceEdit's superior performance in semantic field-based editing, particularly in achieving improved radiance field disentanglement.

Keywords: Neural Radiance Fields · Neural Rendering · 3D-aware GANs

1 Introduction

In recent years, there has been significant interest in vision and graphics in generating visually captivating and photo-realistic images. Notably, 3D aware Generative Adversarial Networks incorporating adversarial learning [11] and Neural

This work is supported by Jibaben Patel Chair in AI.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_4.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 46–62, 2025.
https://doi.org/10.1007/978-3-031-78172-8_4

Radiance Fields [25] have made remarkable progress in producing multiple-view images that closely resemble real photographs. Despite advancements, 3D aware Generative Adversarial Networks (GANs) [6,12,28] still struggle to match the resolution of state-of-the-art 2D GAN models. This limitation arises from the high computational demands of learning volumetric representations. Recently proposed methods [5,8] try to efficiently learn radiance fields by a tri-plane representation and learnable radiance manifolds, respectively. Nevertheless, these methods lack control over learned geometry, thus posing challenges in decoupling appearance from geometry.

To decouple facial geometric and appearance attributes, FENeRF [35] adopts π-GAN [6] and learns facial semantic volume aligned with geometry. The learned facial semantics act as an intermediary, offering control over geometry and appearance. However, implementing local editing with FENeRF necessitates time-consuming and resource-intensive retraining. Addressing this limitation, [15] proposes an AdaIN-based Controllable Appearance Module (CAM) and a geometry decoder. This approach allows for independent control of appearance from geometry. Unlike FENeRF, this approach requires training a geometry decoder on learned tri-plane representations from [5] to obtain the semantic volume for editing. Consequently, the CAM module in [15] grants control over the appearance of the entire generated portrait. While introducing the geometry decoder branch facilitates disentangled face editing, it does not provide semantic control over appearance.

In this work, we introduce SemFaceEdit, building upon efficient Generative Radiance Manifolds (GRAM) [8]. GRAM employs 2D manifolds to constrain point sampling and radiance field learning, jointly trained with GAN. This constrained approach reduces the computational workload and enables fine detail learning by confining point sampling and radiance learning within a reduced space. In contrast to GRAM, our approach learns both semantically disentangled geometry and appearance in a unified manner. We introduce two modules: the Geometry Module and the Appearance Module. The Geometry Module predicts the semantic radiance field within a volume, which the Appearance Module subsequently uses to condition the appearance semantically. Unlike existing methods that can entirely change the appearance or geometry of a radiance field, our proposed approach enables semantic-specific editing while preserving the integrity of other semantic information in terms of geometry or appearance. Our method facilitates changes in geometry and appearance within a single photograph and enables seamless transfer of hairstyle or other semantic facial attributes from one source reference to another.

The main contributions of this work are summarized as follows.

- We present a novel framework based on generative radiance manifolds, allowing for semantic control over the geometry and appearance by manipulating latent spaces specific to each semantic attribute.
- We design a differentiable Semantic Volume masking layer, which learns to segregate each point in radiance volume into semantic groups. This layer

enables targeted appearance transfer concerning aspects like colours and hues and selective geometry transfer for various semantic regions.
– Our method excels in geometry and appearance control, allowing for the seamless transfer of geometry or appearance associated with one semantic label in a radiance field to another.

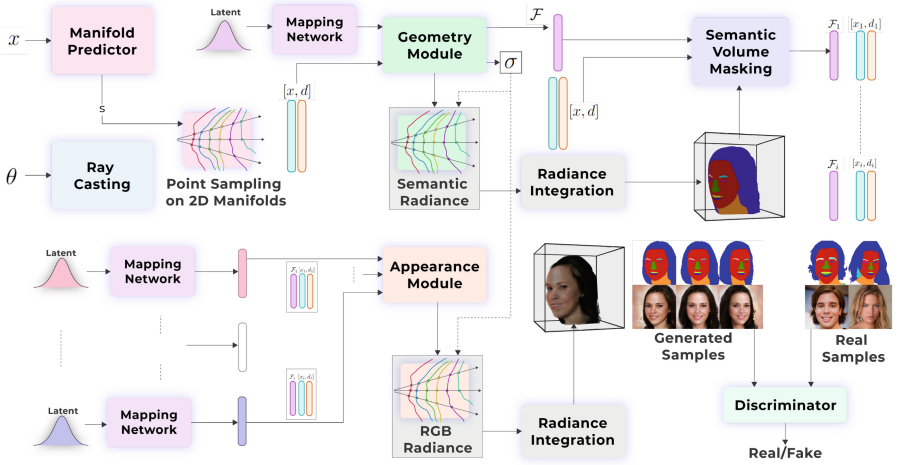


Fig. 1. An overview of our proposed framework. We sample points in volume by determining intersections of casted rays with isosurfaces predicted by Manifold Predictor [8]. Subsequently, the Geometry Module conditions these points using a latent vector sampled from a Gaussian distribution, resulting in diverse predictions for occupancy (σ), semantic radiance, and Appearance Descriptor $\mathcal{F}$. To segregate points and their appearance descriptors based on semantic classes, we employ the Semantic Volume Masking layer, which relies on semantic radiance information. The Appearance Module then utilizes different latent codes to condition each set of points and predicts RGB Radiance. Selectively conditioning Appearance Descriptors $\mathcal{F}$ based on semantics enables the Appearance Module to independently modify a specific semantic appearance.

2 Related Works

Neural Implicit Representations. Neural Implicit functions have been widely applied to various vision problems such as novel view synthesis [32,36], and image manipulations [1,40]. Furthermore, they have gained significant traction in diverse 3D-related tasks, including 3D scene reconstruction [25], and occupancy field or signed distance function estimation [24,28]. Recent works have explored alternative representations of implicit functions, such as octrees and tri-planes [5,37], for faster inference and improved expressive power. Additionally, GRAM [8] introduces an effective point sampling method and radiance field learning

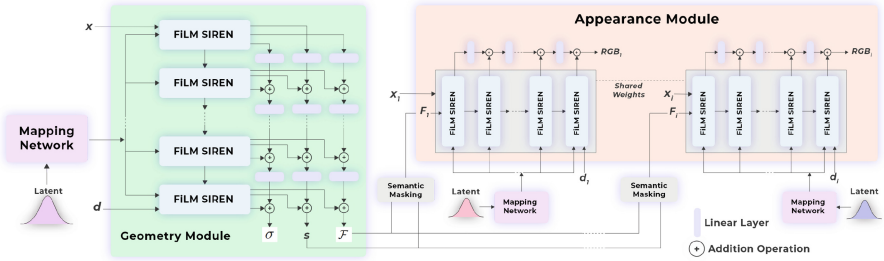


Fig. 2. The network architecture of Geometry Module and Appearance Module.

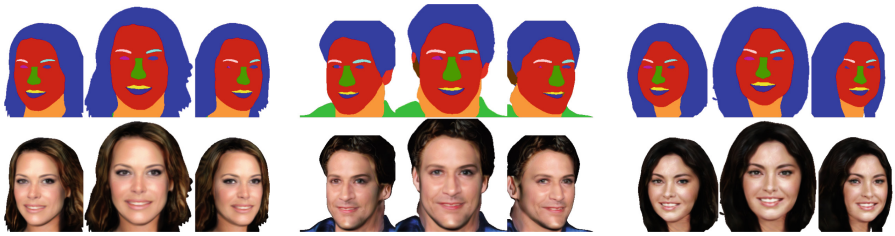


Fig. 3. Renderings of Semantic-radiance and RGB-radiance on image space generated by our approach by random latent code $\mathbf{z} \in \mathrm{R}^d$ and $\mathbf{z}_i \in \mathrm{R}^d$.

on 2D manifolds, enabling fine detailed learning with reduced computational complexity. This paper emphasizes learning semantic information on 2D manifolds through a geometry module rather than directly learning about the image space. However, learning the semantic field alone encounters challenges due to the absence of depth information in the semantic mask data. To address this, the learning of the semantic field is further supervised by an additional appearance module. Unlike [15,35], our appearance module uses the semantic information from the geometry module to control the appearance of each semantic point on these manifolds in the volume.

3D-Aware Neural Face Image Synthesis. The combination of Generative Adversarial Networks (GANs), as originally proposed [11], and neural implicit radiance fields (NeRF) [25], has been widely explored for the generation of 3D-consistent faces. Despite utilizing 3D aware features and 2D CNN-based renderers to generate synthetic images, several methods such as [6,12,22,26] face challenges in accurately representing geometry, resulting in lower quality representations. Recent studies [2,5,8,28] investigate novel representations and models, enabling the synthesis of geometrically consistent fine details.

Editing Faces with 2D GANs and DDPMs. Generative adversarial networks (GANs), and more recently, Denoising Diffusion Probabilistic Models (DDPMs) have widely been applied for generating realistic images, with face editing as a prominent area of research in this context. Various methods have

been proposed for face editing using semantic maps. These approaches leverage conditional 2D GANs [7,20,29,42], and DDPMs [14,17,33] which are conditioned on semantic masks. Unlike explicit field/occupancy-based neural scene representations, SofGAN [7] utilizes implicit fields for semantic occupancy generation. However, SofGAN relies on semantically labelled 3D scans for training and involves a separate process to synthesize images through 2D projections of semantic labels. Diffusion-Rig [9] first learn facial priors by fitting a morphable face model for conditioning diffusion model to generate view-consistent 2D facial images. In contrast, our proposed framework directly modifies appearance in the generated implicit neural representation without the need for 2D projections or facial priors, and is trained on a collection of 2D images instead of 3D scans.

Editing Faces in Neural Radiance Fields. To enable control over the geometry and appearance of synthesized images by 3D-aware GANs, subsequent works employ the embedding of explicit control into the generation process. Certain methods, including [3,39,43], condition radiance fields with information from 3D Morphable models (3DMM) [21,30] to exert control over the generated fields. This conditioning is achieved through embeddings of latent spaces representing pose, shape, and expression parameters of the 3DMM models. FENeRF [35] utilizes semantic masks for geometry control and a separate latent vector for appearance control in synthesized images. CG-NeRF [16] introduces soft conditions, including sketches as external inputs, to influence radiance field generation. Following [5], IDE-3D [34] enables editing on tri-plane representations while NeRFFaceEditing [15] disentangles geometry and appearance using pre-trained tri-plane representations, eliminating the need for retraining.

While the aforementioned methods provide control over the geometry and appearance of synthesized images, they fall short in offering semantic-specific control. Our proposed method provides seamless semantic-specific control over geometry and appearance attributes in facial images, as shown in Fig. 6. This allows for effortlessly transferring specific facial attributes to other images, including overall geometry and appearance.

3 Methodology

Our primary objective is to learn semantic information, occupancy, and colour attributes for points sampled on 2D Manifolds. The Manifold Predictor model predicts these manifolds [8]. To separate the representation of geometry and texture during the image generation process, we introduce two distinct modules: the Geometry module and the Appearance module, as illustrated in Fig. 1. The Geometry module employs a shape latent code for each point on the manifolds, influencing occupancy and semantic information. Simultaneously, it predicts high-dimensional Appearance Descriptors $\mathcal{F}$ for each point. These descriptors are segregated based on semantic information using the Semantic Volume Masking Layer. The Appearance module then conditions points belonging to a specific semantic with different appearance latent codes and predicts colour information. The entire method is trained in two stages using adversarial learning.

Initially, we train the entire network end-to-end, simultaneously learning weights for the Geometry and Appearance modules. Later, we freeze the weights of the Geometry module and fine-tune the Appearance module to achieve high-quality volumetric renderings of generated images.

3.1 Network Architecture

Our generator consists of the Geometry Module and the Appearance Module, as illustrated in Fig. 1. Both modules are implemented as Multi-Layer Perceptrons (MLPs) to produce occupancy σ, semantic radiance s, and colour $c = (r, g, b)$ values for a given point $x \in \mathrm{R}^3$ with view direction $d \in \mathrm{R}^3$.

$$\Psi_g : (\mathbf{z}, \mathbf{x}, \mathbf{d}) \mapsto (\sigma, \mathbf{s}),\ \Psi_a : (\{\mathbf{z_i}\}_{i=0}^{l}, \mathbf{x}, \mathbf{d}, \mathbf{s}) \mapsto (\mathbf{c}) \tag{1}$$

where, $\mathbf{z} \in \mathrm{R}^d \sim p_z$ represents the geometric latent code and $\mathbf{z_i} \in \mathrm{R}^d \sim p_z$ represents the appearance latent code for each semantic labels l.

Manifold Predictor. We sample N points along each ray cast in the direction $\mathbf{d}$, given a camera pose $\theta \in \mathrm{R}^3 \sim p_\theta$. The Manifold Predictor $\mathcal{M}$ is implemented as a scalar field function using light-weight MLPs, predicting a scalar value s for the sampled points. From the predicted scalar field, we extract K iso-surfaces with different levels l_i and perform final point sampling by finding the nearest point of intersection on these iso-surfaces, following [8].

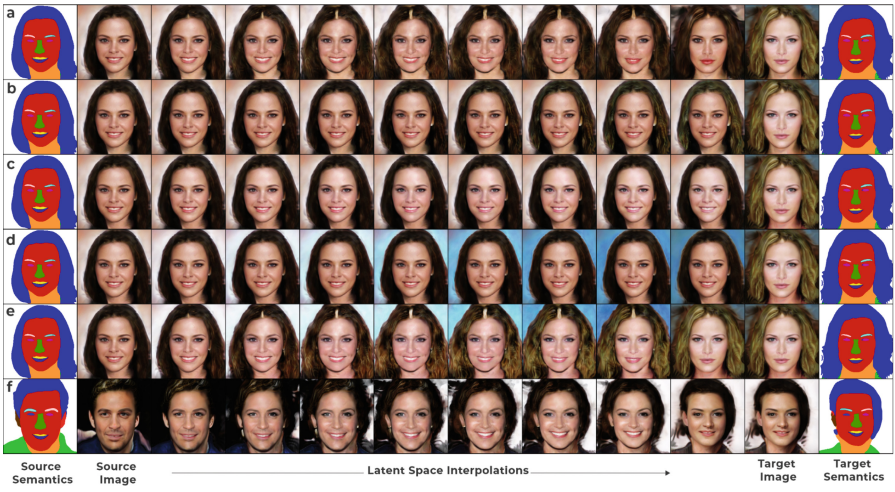


Fig. 4. Latent space disentanglement and interpolations. In (a), linear interpolation is performed on the geometric latent space z while keeping all appearance latent z_i fixed. (b–d) demonstrate changes in appearance latent variables corresponding to hair, face, and background semantics. All latent codes are linearly interpolated for (e) and (f). Additionally, (f) showcases the change in the appearance of garment semantics from the source to the target.

Geometry Module. Given a latent code z, the Geometry module Ψ_g generates the radiance for each point on the predicted 2D manifolds in volume. The architecture of Ψ_g consists of FiLM SIREN backbone, inspired from [6] with some modifications, as presented in Fig. 2. Instead of directly synthesizing images as in [8], our geometric module first learns to predict occupancy σ and semantic radiance field, **s**. Our primary goal is to disentangle geometry from appearance by learning geometry independently of image synthesis. However, attempting to learn semantic radiance solely using semantic masks in an adversarial setting is bound to fail, as the perceptual appearance of a semantic mask lacks depth information present in images due to the absence of shading effects caused by varying shades of light.

To address this limitation, we additionally predict high-dimensional feature descriptors $\mathcal{F}$ for each point, which is then conditioned by the Appearance module for image synthesis. By backpropagating gradients through $\mathcal{F}$ from the appearance module into the geometry module, we ensure the correct learning of the occupancy field σ. This approach helps in better disentangling geometry and appearance, leading to improved results in our framework.

Semantic Volume Masking Layer. Given Semantic radiance field **s** and occupancy field σ, we perform volumetric rendering [27,41] on all 2D manifolds. When a ray r is cast and intersects K surface manifolds at points x_j sorted by proximity, we can describe the semantics of a point on the k-th manifold out of the total K manifold:

$$\mathrm{S}_k(r) = \arg\max \sum_{j=k}^{K} T(\mathrm{x}_j)\sigma(\mathrm{x}_j)\mathrm{s}(\mathrm{x}_j, \mathrm{d}), \quad \textbf{for} \;\; k \leq K \tag{2}$$

Here, $T(x_j) = \prod_{i<j;i\geq k}(1-\sigma(\mathrm{x}_j))$, and Eq. 2 allows us to classify each sampled point to the semantic class it belongs to. After obtaining the semantic labels, we segregate the points and their corresponding appearance descriptors $\mathcal{F}$ based on their specific semantic categories. Using the Appearance module, this segregation allows us to condition them on respective semantic-specific latent codes.

Let there be n semantic classes, and the Semantic Volume Masking layer segregates all sampled points in the volume into n collections, denoted by $\mathrm{x_i}$ along with their corresponding casted direction $\mathrm{d_i}$ and appearance descriptor $\mathcal{F}_i$. These collections are created such that every point $x \in \mathrm{x}_i$ belongs to a specific semantic category i, defined among the n available semantic classes.

Appearance Module. The architecture of our appearance module closely resembles that of the Geometry Module, as depicted in Fig. 2. However, it is distinguished by the presence of n mapping networks, each responsible for conditioning the i-th collection obtained from the Volume Masking layer. These mapping networks are utilized to condition the Appearance Descriptors $\mathcal{F}_i$, which then pass through a SIREN-based neural network for RGB radiance prediction. As a result, our appearance module Ψ_a generates a radiance field c, where the i-th latent code controls the appearance of the i-th semantic among all n semantic classes. Importantly, it should be noted that while different mapping networks

Fig. 5. Appearance Latent Codes (z_i) influence on Geometric Shapes. The image (i, j) in grid (a), (b), and (c) are generated with same z_i's, highlighting resemblance in appearance across hair, background, and facial regions with (i, j)th image in each grid.

are used for conditioning the collections, the weights of the SIREN backbone in the appearance module remain consistent across all collections. In the Results Section, we present exhaustive ablation studies conducted on the Appearance Module to identify the optimal network architecture. The Appearance Module takes segregated Appearance Descriptors from the geometry module and a latent code for each semantic as input, producing the final RGB radiance. Subsequently, we perform volumetric radiance integration with the occupancy field σ, obtained from the Geometry Module, following a similar approach as in [8,27], to obtain the final image.

4 Experiments and Results

4.1 Training

We train SemFaceEdit on the CelebAMask-HQ [19] dataset. The dataset contains 30K high-resolution face images and annotated masks in $N = 19$ classes. In our experiments, we club N classes in 4 semantics, denoted as n: hairs, face (containing eyes, nose, mouth, neck, and ears), garment, and background. We train our network in two stages in an adversarial learning setting, using non-saturating GAN loss with R1 regularization [23]. During training, we randomly sample geometric latent code z, n appearance latent codes z_i and pose θ from Gaussian distributions $p_z^g, p_{z_i}^a$, and p_θ, respectively. The generator, consisting of Ψ_g and Ψ_a, produces semantic s, occupancy σ, and RGB c, radiance fields. In the first stage, we jointly train Ψ_g and Ψ_a with two discriminators, Semantic map discriminator D_s and Facial image discriminator D_c. We provide further details on data preprocessing, discriminator architecture and loss functions in *Suppl. material*. In Fig. 3, We present multi-view images generated from randomly sampled latent codes after training the model.

Implementation Details. Our experiments utilize the Adam Optimizer [18] with learning rates of 2×10^{-5} for the generator and 2×10^{-4} for the discriminator. Utilizing 2 NVIDIA GeForce RTX 4090 GPUs, each with 24 GB of memory, the model is trained for 120K iterations (3 epochs) during the first stage, followed by 30K iterations (1 epoch) in the second stage. To generate neural rendering of radiance field on to image space $\mathcal{I} \in \mathrm{R}^{W\times H\times 3}$, we cast $W \times H$ rays from the camera and sample K number of points along each ray, where $K = 24$ is number of iso-surfaces on manifolds. Specifically, we sample radiance values of 3.14 million ($256 \times 256 \times 24 \times 2$) points in total from semantic-radiance field s and rgb-radiance field c. To overcome computational challenges, we utilize PyTorch's Automatic Mixed Precision (AMP) to reduce memory cost. During inference, our approach generates 256^2 and 128^2 image in 0.86 and 0.15 s with peak GPU memory usage of approx. 15 GB and 6 GB, respectively.

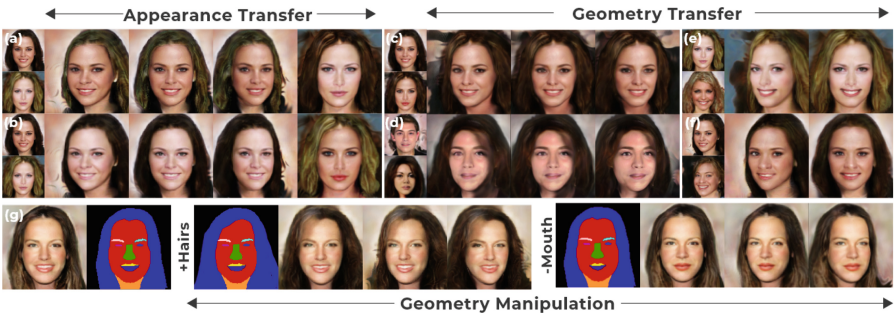


Fig. 6. (a–f) displays two smaller images, with the top image representing the source attribute and the bottom image representing the target attribute of appearance/geometry. **Appearance Transfer**: In (a), we present three different views of the source image showcasing the transferred hairstyle appearance from the target to the source image. Additionally, we present a single view of a converse transfer of appearance, i.e. from the source to the target image. Similarly, in (b), we showcase the transferred facial appearance. **Geometry Transfer**: (c) and (d) demonstrate the transfer of target hair geometry to the source image. In contrast, (e) and (f) showcase the smile and nose geometry transfer from the target to the source image. Note that in (c) and (d), alterations in attributes at regions other than hair semantics are minimal. In contrast, the transferred hair adopts the appearance style of the target image through manipulation of the latent code governing hair semantics. The focus is solely on transferring geometry for finer semantics like the nose or smile in (e, f). Therefore, we constrain appearance transfer by maintaining the unchanged state of appearance latents. **Geometry Manipulation**: (g) Shows the effect of manipulation in the semantic mask by expanding the hair region semantics and the effect of shrinking at the mouth region.

4.2 Editing Faces in Radiance Fields

To edit an input image obtained either from $\mathrm{I} \sim p_{rim}$ (real images) or synthetically generated using Generators Ψ_g and Ψ_a, we perform image inversion

into the $\mathcal{W}$ space, represented as w, utilizing pivotal tuning inversion [31]. To enable image editing based on semantic mask manipulations, we require the original mask S and the user-edited mask S'. For real face images, we utilize the method proposed by [38] to obtain semantic mask S. On the other hand, for synthetic images generated through our pipeline, the Geometry module Ψ_g naturally provides us with the required semantic mask. Our experiments utilize trained mapping networks to obtain the average frequency and phase shifts over 10k sampled latent codes for geometry z and appearance z_i. For brevity, we denote these averaged frequencies and phase shifts as a latent code $w \in \mathcal{W}$ and $w_i \in \mathcal{W}_i$ for geometry and appearance, respectively. To achieve editing, we optimize an editing offset vector $\delta w^+ \in \mathcal{W}$ such that the generated semantic mask S' from Ψ_g approximates the original mask S. During optimization, we keep w_i for Ψ_a fixed while optimizing for the geometric latent δw^+. We directly employ the corresponding z_i's for the appearance of synthetically generated images. For real images, we further optimize for δw_i^+ to make the appearance resemble the given image I. Figure 6 showcases editing in geometry based on semantic masks, and Fig. 7(b, d) demonstrates inversion and latent code manipulations on real images from CelebAMask-HQ Dataset.

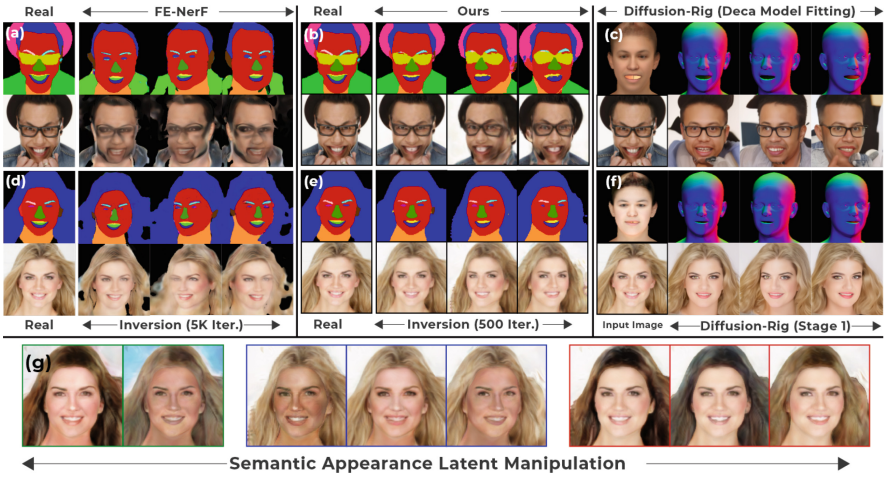


Fig. 7. Comparison of pivotal inversion results between our proposed approach (b, e) and FE-NerF [35] (a, d). Our method achieves rapid convergence within 500 iterations, contrasting with FE-NerF's 5K iterations. Additionally, we compare with Diffusion-Rig [9] (c, f), which first estimate facial prior by fitting a Deca model [10] on given image. Subsequently it employs a diffusion-based approach to generate view consistent facial images. (g) demonstrates semantic appearance variation achieved by modifying latent codes specific to distinct semantic regions $\mathbf{z_i}$ while maintaining geometry latent $\mathbf{z}$ of (d). Green outline signifies full appearance alteration; blue outlines indicate facial appearance changes and red outlines represent hair appearance modifications. Real images for inversion are taken from CelebAMask-HQ Dataset [19].

Geometry and Appearance Transfer. We can directly synthesize appearance manipulations targeting a specific semantic $k \in n$ using Ψ_a. Unlike [15], Ψ_a controls the change in appearance of regions corresponding only to the k-th semantic. We illustrate these results qualitatively in Figs. 5 and 4. However, as discussed previously, image inversion for Geometry transfer into the $\mathcal{W}$ space is required. Given images I_1 and I_2 with semantics S_1 and S_2, respectively, let (w^1, w^2) represent geometric latents and (w_i^1, w_i^2) be appearance latent codes. We want to transfer the semantic region $k \in n$, represented by $\mathcal{M}_k$, from image I_2 to I_1. For appearance transfer, we directly interchange the k-th appearance semantic latent code in w_i^1 with w_k^2, and then optimize for δw^+, such that $\Psi_g(w^1 + \delta w^+, \theta)$ approximates the target semantic $S_1 \odot (1 - \mathcal{M}_k) + S_2 \odot \mathcal{M}_k$, and $\Psi_a(w_i^1, \theta)$ approximates the target image $\mathrm{I}_1 \odot (1 - \mathcal{M}_k) + \mathrm{I}_2 \odot \mathcal{M}_k$. Though w_i^1 for appearance is fixed, inversion requires a backward pass through the Ψ_g and Ψ_a to update δw^+. This is because Ψ_g depends on Ψ_a for the correct estimation of appearance feature descriptors $\mathcal{F}$ and occupancy σ. In Fig. 6 (c–g), we show the geometry transfer from target to source of various semantics. More details can be found in the *Suppl. Material*.

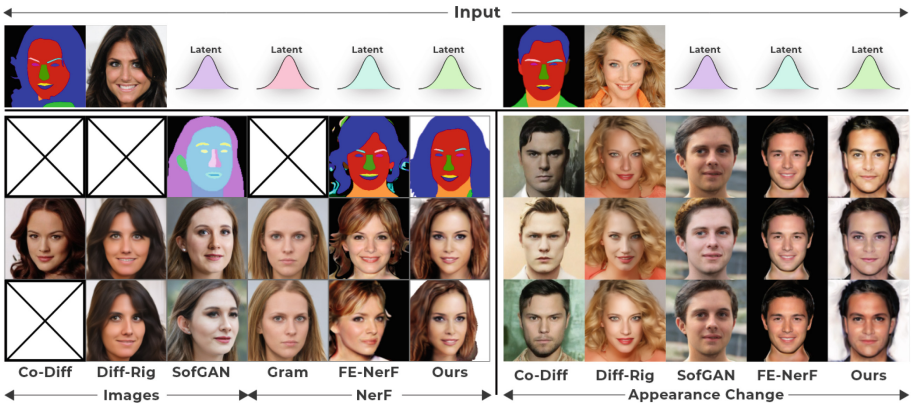


Fig. 8. Qualitative comparison of SemFaceEdit with Co-Diff [14], Diff-Rig [9], SofGAN [7], GRAM [8], and FE-NerF [35]. ☒ represents that the respective approach don't support or yield the required result. Input mask in [14] and image in [9] are taken from CelebAMask-HQ Dataset [19].

4.3 Results

We conduct a comparative analysis of SemFaceEdit against five state-of-the-art techniques: Co-Diff [14], Diff-Rig [9], GRAM [8], FE-NerF [35], and SofGAN [7]. GRAM and FE-NerF are oriented towards generating 3D-aware images, while SofGAN employs a 2D projection of the 3D semantic occupancy field for dynamic styling through a 2D GAN methodology. In contrast to SofGAN, Diff-Rig [9]

employs a diffusion-based approach for view-consistent 2D image generation. This method lacks semantic-guided editing capability, instead utilises expression basis and priors from a morphable face model for appearance changes. We also include Co-Diff [14], a 2D facial image generation approach, which generates realistic facial images given a semantic map but lacks multiple view generation.

Qualitative Comparison. Our framework estimates semantic radiance in addition to the occupancy field and RGB radiance. In Fig. 8 we showcase the generated semantic maps and multi-view images produced by our method compared to the aforementioned approaches. Additionally, on the right, we display the appearance manipulation achieved by our method in comparison with the other techniques. Co-Diff takes a semantic map as input and generates a facial image that fits the provided map, but it cannot generate multi-view images. Diff-Rig first estimates facial priors by fitting Deca model [10] on the given image, then employs a diffusion-based approach to generate view-consistent facial images in 2D space. Changing these buffers along with diffusion noise allows for appearance manipulation and editing, but this methodology lacks semantic-guided editing by design. Sof-GAN produces high-resolution, multi-view facial appearances using a 2D GAN approach, but its training relies on semantically labeled 3D scans. However, the use of 2D GANs prevents it from achieving 3D interpretability. The method presented in GRAM introduces an effective point sampling technique on learnable 2D manifolds. Nonetheless, it falls short in estimating semantic radiance and achieving disentanglement between geometry and appearance. FE-NerF predicts the semantic field but falls short in terms of disentanglement and control over semantic appearance.

Our 3D inversion outcomes, showcased in Fig. 7(b, e), are compared against FE-NerF depicted in Fig. 7(a, d), respectively. Notably, the semantic inversion by FE-NerF struggles with intricate semantics and produces noisy results, as demonstrated in Fig. 7(a). In contrast, our method achieves successful 3D inversion for such cases. Our approach further enables semantic appearance control on the generated RGB-radiance fields. After obtaining optimized geometry latent $\mathbf{z}$ and appearance latents $\mathbf{z}_i$ through inversion process, we can directly manipulate appearances by changing latent codes $\mathbf{z}_i$ corresponding to a semantic region i, as demonstrated in Fig. 7(g). In contrast to our approach which relies on pivotal inversion, Diff-Rig generate the images, shown in Fig. 7(c, f) by first estimating facial priors by fitting Deca face model [10] on the input image. Note that for Diff-Rig, we only present the output generated after Stage-1 in Fig. 7. More accurate results can be obtained by fine-tuning the model on a personalized album, however, this requires multi-view and diverse images of the same face.

Quantitative Comparison. Table 1 presents the assessment of image quality using the Frechet Inception Distance (FID) [13] and Kernel Inception Distance (KID) [4]. This evaluation is performed between 10K generated images from randomly sampled latent codes and an equal number of randomly selected real images. We significantly improve the neural rgb-radiance generation compared to that of FE-NerF, as evident in Table 1 while jointly predicting the semantic radiance field. For Diffusion-Rig, we use 10K real images from the CelebAMask-

Table 1. Comparing FID and KID between 10K generated and 10K real CelebA-MaskHQ [19] Dataset images with GRAM [8], FE-NerF [35], SofGAN [7], Diffusion-Rig [9], and Co-Diff [14]. Lower values (↓) indicate better performance. Superfixes * and † indicates that methods do not generate neural rgb-radiance fields but employ 2-dimensional deep neural networks (GANs/DDPMs) to synthesize multi-view consistent images (*) or only a single view image (†). Abbreviations used: GAC - Global Appearance Control, LAC - Local Appearance Control, and SGE - Semantic Guided Editing. Note that Diffusion-Rig do not support SGE but can perform editing utilizing expression basis and physical buffers of fitted morphable DECA model.

Methods	3D RGB Radiance	3D Semantic Radiance	GAC	LAC	SGE	FID ↓ (256^2)	KID ($\times 10^3$) ↓ (256^2)	#Pars
GRAM	✓	✗	✗	✗	✗	17.36	30.82	1.95 M
FE-NerF	✓	✓	✓	✗	✓	28.97	39.33	31.09 M
SemFaceEdit (Ours)	✓	✓	✓	✓	✓	19.81	32.81	8.78 M
SemFaceEdit (Ours) (Single Stage)	✓	✓	✓	✓	✓	21.27	34.73	8.78 M
SofGAN*	✗	✓	✓	✓	✓	25.72	30.86	30.68 M
Diffusion-Rig*	✗	✗	✓	✗	✗	16.84	30.64	144.6 M
Co-Diff†	✗	✗	✓	✗	✓	14.63	27.43	459.6 M

HQ dataset. We estimate buffers (pose, normal, and albedo) by fitting the Deca Model [10] and generate outputs from stage-1, as proposed by [9]. In contrast, for Co-Diff, we input 10K semantic masks from the CelebAMask-HQ dataset for facial image generation to calculate metrics. To assess the 3D inversion capability of SemFaceEdit, we perform inversions on 1K images from the CelebAMask-HQ Dataset, calculating the mean Intersection over Union (mIoU) for all n semantics. Across all inverted images, our approach converges to 0.85 mIoU within 50 iterations. The per-iteration mIoU and semantic class-wise mIoU progressions are depicted in Fig. 9.

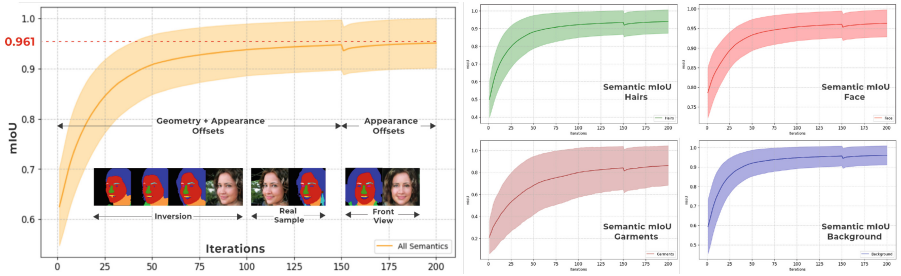


Fig. 9. During the optimization process for 3D inversion on 1000 CelebAMask-HQ Dataset images, we track the Mean Intersection over Union (mIoU). Embedded images qualitatively illustrate the inversion progression of an image in a challenging pose. Furthermore, we track the Mean Intersection over Union (mIoU) of all $n = 4$ semantics, namely: Hairs, Face, Garments, and Background.

Ablation Studies. We conducted several ablation studies to determine the optimal choice for SemFaceEdit architecture. In Table 1, we show improved generated image metrics after second-stage training. This process involves refining

Appearance Module weights while keeping Geometry module weights frozen. The *Suppl. Material* contains extra ablation studies examining (1) the impact of shared Appearance Modules and (2) Depth Ablation.

Limitations. Our appearance module uses the generated semantic radiance from the geometry module through a semantic volume masking layer. The generation of RGB-radiance is constrained to adhere to the densities of the semantic radiance, which limits the creation of detailed and fine-grained geometry. Additionally, any discrepancies in the semantic geometry propagates to final facial appearance. We include illustrations of such cases in *supp. material*. The high computational requirements for training generative NeRFs limit the proposed method's ability to generate high-quality images, resulting in image quality that lags behind traditional methods using 2D GANs and DDPMs. In the future, more efficient point sampling and implicit representations could be explored to reduce computational overhead and achieve higher quality radiances.

5 Conclusion

This work introduces a novel approach for semantic face editing on generative radiance manifolds. We enable effective editing capabilities by simultaneously estimating each point's semantic information and RGB radiance. The key to our approach lies in utilizing a differentiable Semantic Volume Masking layer, which effectively partitions points according to their predicted semantics. This partitioning allows us to condition the RGB radiance with latent codes, resulting in a finely tuned level of control over the editing process. With SemFaceEdit, we successfully attain comprehensive control over geometry and appearance, including transferring distinct facial attributes between images. Through experiments, we demonstrate the capability of our method to effectively control the visual attributes of distinct semantics while preserving other regions by manipulating latent codes. Moreover, SemFaceEdit achieves enhanced GAN inversion and faster convergence compared to existing state-of-the-art methods.

References

1. Abdal, R., Zhu, P., Mitra, N.J., Wonka, P.: StyleFlow: attribute-conditioned exploration of styleGAN-generated images using conditional continuous normalizing flows. ACM Trans. Graph. (ToG) **40**(3), 1–21 (2021)
2. An, S., Xu, H., Shi, Y., Song, G., Ogras, U., Luo, L.: PanoHead: geometry-aware 3D full-head synthesis in 360. arXiv preprint arXiv:2303.13071 (2023)
3. Athar, S., Shu, Z., Samaras, D.: Flame-in-NeRF: neural control of radiance fields for free view face animation. In: 2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG), pp. 1–8. IEEE (2023)
4. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying MMD GANs. arXiv preprint arXiv:1801.01401 (2018)
5. Chan, E.R., et al.: Efficient geometry-aware 3D generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16123–16133 (2022)

6. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: PI-GAN: periodic implicit generative adversarial networks for 3D-aware image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5799–5809 (2021)
7. Chen, A., Liu, R., Xie, L., Chen, Z., Su, H., Yu, J.: SofGAN: a portrait image generator with dynamic styling. ACM Trans. Graph. **41**(1), 1–26 (2022)
8. Deng, Y., Yang, J., Xiang, J., Tong, X.: GRAM: generative radiance manifolds for 3D-aware image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10673–10683 (2022)
9. Ding, Z., Zhang, X., Xia, Z., Jebe, L., Tu, Z., Zhang, X.: DiffusionRig: learning personalized priors for facial appearance editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12736–12746 (2023)
10. Feng, Y., Feng, H., Black, M.J., Bolkart, T.: Learning an animatable detailed 3D face model from in-the-wild images. ACM Trans. Graph. (ToG) **40**(4), 1–13 (2021)
11. Goodfellow, I., et al.: Generative adversarial nets. Adv. Neural Inf. Process. Syst. **27** (2014)
12. Gu, J., Liu, L., Wang, P., Theobalt, C.: StyleNeRF: a style-based 3D-aware generator for high-resolution image synthesis. arXiv preprint arXiv:2110.08985 (2021)
13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. Adv. Neural Inf. Process. Syst. **30** (2017)
14. Huang, Z., Chan, K.C., Jiang, Y., Liu, Z.: Collaborative diffusion for multi-modal face generation and editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6080–6090 (2023)
15. Jiang, K., Chen, S.Y., Liu, F.L., Fu, H., Gao, L.: NeRFFaceEditing: disentangled face editing in neural radiance fields. In: SIGGRAPH Asia 2022 Conference Papers, pp. 1–9 (2022)
16. Jo, K., Shim, G., Jung, S., Yang, S., Choo, J.: CG-NeRF: conditional generative neural radiance fields. arXiv preprint arXiv:2112.03517 (2021)
17. Kim, G., Kwon, T., Ye, J.C.: DiffusionClip: text-guided diffusion models for robust image manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2426–2435 (2022)
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
19. Lee, C.H., Liu, Z., Wu, L., Luo, P.: MaskGAN: towards diverse and interactive facial image manipulation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
20. Leimkühler, T., Drettakis, G.: FreestyleGAN: free-view editable portrait rendering with the camera manifold. arXiv preprint arXiv:2109.09378 (2021)
21. Li, T., Bolkart, T., Black, M.J., Li, H., Romero, J.: Learning a model of facial shape and expression from 4d scans. ACM Trans. Graph. **36**(6), 194–1 (2017)
22. Liao, Y., Schwarz, K., Mescheder, L., Geiger, A.: Towards unsupervised learning of generative models for 3D controllable image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
23. Mescheder, L., Geiger, A., Nowozin, S.: Which training methods for GANs do actually converge? In: International Conference on Machine Learning, pp. 3481–3490. PMLR (2018)
24. Michalkiewicz, M., Pontes, J.K., Jack, D., Baktashmotlagh, M., Eriksson, A.: Implicit surface representations as layers in neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4743–4752 (2019)

25. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (2021)
26. Niemeyer, M., Geiger, A.: Giraffe: representing scenes as compositional generative neural feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11453–11464 (2021)
27. Oechsle, M., Peng, S., Geiger, A.: UNISURF: unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5589–5599 (2021)
28. Or-El, R., Luo, X., Shan, M., Shechtman, E., Park, J.J., Kemelmacher-Shlizerman, I.: StylesDF: high-resolution 3D-consistent image and geometry generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13503–13513 (2022)
29. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2337–2346 (2019)
30. Paysan, P., Knothe, R., Amberg, B., Romdhani, S., Vetter, T.: A 3D face model for pose and illumination invariant face recognition. In: 2009 sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, pp. 296–301. IEEE (2009)
31. Roich, D., Mokady, R., Bermano, A.H., Cohen-Or, D.: Pivotal tuning for latent-based editing of real images. ACM Trans. graph. **42**(1), 1–13 (2022)
32. Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: DeepVoxels: learning persistent 3D feature embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2437–2446 (2019)
33. Sun, J., Deng, Q., Li, Q., Sun, M., Ren, M., Sun, Z.: AnyFace: free-style text-to-face synthesis and manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18687–18696 (2022)
34. Sun, J., Wang, X., Shi, Y., Wang, L., Wang, J., Liu, Y.: IDE-3D: interactive disentangled editing for high-resolution 3D-aware portrait synthesis. ACM Trans. Graph. (ToG) **41**(6), 1–10 (2022)
35. Sun, J., et al.: FENERF: face editing in neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7672–7682 (2022)
36. Tucker, R., Snavely, N.: Single-view view synthesis with multiplane images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 551–560 (2020)
37. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: PlenOctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5752–5761 (2021)
38. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: BiSeNet: bilateral segmentation network for real-time semantic segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 334–349. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_20
39. Zheng, Y., Abrevaya, V.F., Bühler, M.C., Chen, X., Black, M.J., Hilliges, O.: IM avatar: implicit morphable head avatars from videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13545–13555 (2022)

40. Zhou, P., Xie, L., Ni, B., Tian, Q.: CIPS-3D: a 3D-aware generator of GANs based on conditionally-independent pixel synthesis. arXiv preprint arXiv:2110.09788 (2021)
41. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018)
42. Zhu, P., Abdal, R., Qin, Y., Wonka, P.: SEAN: image synthesis with semantic region-adaptive normalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5104–5113 (2020)
43. Zhuang, Y., Zhu, H., Sun, X., Cao, X.: MoFaNeRF: morphable facial neural radiance field. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13663, pp. 268–285. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20062-5_16

D^{2}Styler: Advancing Arbitrary Style Transfer with Discrete Diffusion Methods

Onkar Susladkar[1], Gayatri Deshmukh[2], Sparsh Mittal[3(✉)], and Parth Shastri[2]

[1] Yellow.AI, Bengaluru, India
[2] Mumbai, India
shastripp18.extc@coeptech.ac.in
[3] Indian Institute of Technology, Roorkee, Roorkee, India
sparsh.mittal@ece.iitr.ac.in

Abstract. In image processing, one of the most challenging tasks is to render an image's semantic meaning using a variety of artistic approaches. Existing techniques for arbitrary style transfer (AST) frequently experience mode-collapse, over-stylization, or under-stylization due to a disparity between the style and content images. We propose a novel framework called D^{2}Styler (Discrete Diffusion Styler) that leverages the discrete representational capability of VQ-GANs and the advantages of discrete diffusion, including stable training and avoidance of mode collapse. Our method uses Adaptive Instance Normalization (AdaIN) features as a context guide for the reverse diffusion process. This makes it easy to move features from the style image to the content image without bias. The proposed method substantially enhances the visual quality of style-transferred images, allowing the combination of content and style in a visually appealing manner. Experimental results demonstrate that D^{2}Styler produces high-quality style-transferred images and outperforms **twelve** existing methods on nearly all the metrics. The qualitative results and ablation studies provide further insights into the efficacy of our technique. The code is available at https://github.com/Onkarsus13/D2Styler.

Keywords: Neural Style Transfer · Vector Quantization · Latent Diffusion

1 Introduction

Given a content image and a style image, Style transfer (ST) synthesizes a new image by transferring the style from the style image while preserving the substance from the content image. Neural style transfer (NST) has been studied extensively in recent years [14]. NST seeks to learn how humans perceive images, as transferring style without significantly altering the semantic content in the target image requires a single network to disentangle the style and the content.

G. Deshmukh and P. Shastri—Independent Researcher.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 63–82, 2025.
https://doi.org/10.1007/978-3-031-78172-8_5

Fig. 1. Results from $D^2Styler$, our proposed method. The content image (the one on the top of the tree) is converted to different stylized versions based on the corresponding style image (shown in the inset).

After the introduction of pioneering work by Gatys et al. [14], many works have proposed improvements in use of a single feed-forward method [21], loss function, use of regularization, and normalization techniques.

While GANs have been used to perform NST [23], GANs are challenging to train and offer no control over the style and content of the output image. Researchers have used standalone flow-based models and GAN plus flow-based models for image generation. These models allow control over the output image attributes [2,13]. Recently, diffusion models have become popular for image generation [33,36,37]. These models allow generating new images based on text prompts, image in-painting, and conditional image-to-image translation. These models also allow arbitrary style transfer by giving an image and a text prompt specifying the style we need to transfer. However, these models face a strict trade-off between style transfer and content preservation.

We present D^2Styler, a technique to perform arbitrary style transfer for a given content image and a style image. Figure 1 illustrates D^2Styler output. D^2Styler uses a pretrained VQ-GAN encoder [11] to encode the content and style images. Then, it models their combined latent space by a conditional diffusion model. This diffusion model is conditioned on the features extracted by matching the statistics of the content and style images. To achieve this, an AdaIN [20] layer with a pre-trained convolutional encoder network is used. Our key idea is that this approach provides a context for the diffusion decoder to predict the masked inputs correctly. The resultant learned latent code is passed through the VQ-GAN decoder to obtain the style-transferred image. Our key contributions are:

1. We introduce a pioneering approach that combines discrete diffusion with AdaIN, uniquely addressing the prevalent issues of mode-collapse and over-stylization in existing style transfer methods. This integration not only stabilizes the training process but also ensures the preservation of content integrity while applying diverse artistic styles, a critical improvement over prior methodologies.
2. We propose using AdaIN [20] features to guide the diffusion decoder, conditioning the model on matched statistics between style and content features. This approach enables more precise control over the style transfer process,

ensuring that the output closely aligns with the desired stylistic attributes while maintaining the integrity of the content.
3. We propose a new loss function ($L_{feature}$) that drives the output image to match the features from the AdaIN [20] layer. This loss, coupled with style and content losses, makes the model generate plausible stylized images.
4. A rigorous evaluation on multiple benchmarks show that D^{2}Styler outperforms twelve established style transfer techniques [6,13,14,25,28,32,34,36,44–46] on key metrics such as SSIM and LPIPS. This not only demonstrates the effectiveness of our method but also highlights its robustness across various artistic and content scenarios. D^2Styler has 78M parameters.
5. We have shown that D^{2}Styler can effectively apply multiple styles to a single content image (Sect. 4.3). This capability significantly expands artistic possibilities, allowing for the creative blending of diverse styles within a single output, which is especially valuable in complex digital art and design projects.
6. D^{2}Styler is designed to achieve high-quality results in fewer diffusion steps than conventional diffusion architectures and auto-regressive methods [18] (Sect. 5). This significantly reduces computational costs and improves the feasibility of deploying style transfer in real-time scenarios, thus addressing a major bottleneck in the adoption of NST technologies.
7. Ablation studies illustrate the contributions of each component of D^{2}Styler, providing clear evidence to the community on the effectiveness of methodology.

2 Related Work

Neural Style Transfer (NST): NST has been extensively studied in non-photorealistic rendering and texture generation [10]. Gatys et al. [14] employ a Gram matrix to extract features from a pre-trained DNN with an iterative optimization network to produce stylized images using multi-level feature correlations. Since then, many works have addressed key NST issues, including speed, control, quality, photorealism and temporal style transfer.

As a workaround for slow iterative optimization strategies in NST, feed-forward networks are trained to minimize the same losses based on Gram matrices [14]. These feed-forward frameworks are faster than iterative optimization and appropriate for real-time deployment [21]. Ulyanov et al. [40] propose feed-forward network enhancements to improve example quality and variety. However, these techniques can transfer a limited number of styles because of their training process. To overcome this constraint, Dumoulin et al. [9] suggest a framework based on a conditional instance normalization layer that can transfer 32 styles. Li et al. [31] propose a framework that can transfer 300 styles. However, these methods cannot transfer arbitrary styles.

AdaIN [20] uses a feed-forward network to match style and content feature statistics in an intermediate layer. However, AdaIN does not generalize well and faces mode collapse. The WCT technique [30] matches content and style covariance utilizing whitening and color transform. AdaAttN [32] considers both

high-level and low-level features through adaptive attention normalization. The AST methods involving encoder-decoder architectures are prone to information loss due to pooling layers of the encoder network. This causes deformation of the output content.

Generative Models: The Variational Auto-encoder (VAE) [24] proposes maximizing a lower bound on data probability to learn latent space representation. It learns the manifold representation of the input data distribution and generates new samples from the learned continuous latent space. Despite the impressive results of GANs in image-to-image translation and style transfer, they still suffer from mode collapse, complicated training, and instability. To overcome these problems, VQ-VAE [41] learns a discrete latent space instead of a continuous one. This inspired the development of VQ-GAN [11], which uses transformers and discrete latents. Other works, such as DALL.E [35] and Cogview [8], have also leveraged discrete latent and auto-regressive methods to achieve remarkable results in image generation. However, models based on discrete representations face challenges, such as increased accumulated errors, decreased speed for high-resolution images, and directional bias.

Diffusion Models: The denoising diffusion probabilistic model is a generative model inspired by thermodynamics's "diffusion" process. Discrete diffusion has been applied to text-generation [19] and image-generation [3], however, these models were restricted to generating 32×32 images. The recently proposed VQ-diffusion [15] technique enables efficient text-to-image generation and provides results comparable with the continuous paradigm.

Kwon et al. [27] propose utilizing style and structure losses to direct the sampling process for text-guided image translation. Wang et al. [43] fine-tune diffusion models by integrating CLIP, enabling them to learn style references through text prompts. Everaert et al. [12] recommend fine-tuning Stable Diffusion with a new noise distribution that mimics the style images' distribution. Diffusion-Enhanced PatchMatch [16] incorporates patch-based techniques with whitening and coloring transformations in the latent space. StyleDrop [39] model, which is based on the generative vision transformer Muse [5] rather than text-to-image diffusion models, produces content in diverse visual styles. Models like DreamStyler [1] exhibit advanced textual inversion, utilizing techniques such as BLIP-2 [29] and an image encoder to generate content by inverting text and content images while associating style with text. Kim et al. [22] fine-tune a pre-trained DDIM to generate images based on text descriptions, introducing a local directional CLIP loss that ensures the direction between the generated image and the original image closely matches the direction between the reference (original domain) and target text (target domain). Chandramouli et al. [4] employ a deterministic forward diffusion approach, achieving the desired manipulation by using the target text to condition the reverse diffusion process. Prompt-to-Prompt [17] aims to preserve some original image content by modifying the cross-attention maps.

3 D^2Styler: A Novel AST Framework

D^2Styler harnesses the power of discrete diffusion and AdaIN to address the inherent challenges of style transfer. Building upon the discrete representational capabilities of VQ-GANs, *D^2Styler* pioneers a unique approach to arbitrary style transfer that promises enhanced visual quality and reduced mode collapse by intelligently navigating the latent space of content and style images. Unlike traditional feed-forward methods, *D*2Styler uses the high-fidelity image generation powers of diffusion models and the fine-tuned control provided by Vector-Quantized feature spaces introduced by VQ-GAN [15]. Figure 2 shows the architecture of D^2Styler. This architecture functions in two distinct stages: stage 1 and stage 2. During stage 1, the style and content images are taken as inputs and turned into condensed features using latent discrete diffusion. Stage 2 leverages these condensed features to generate the final stylized image.

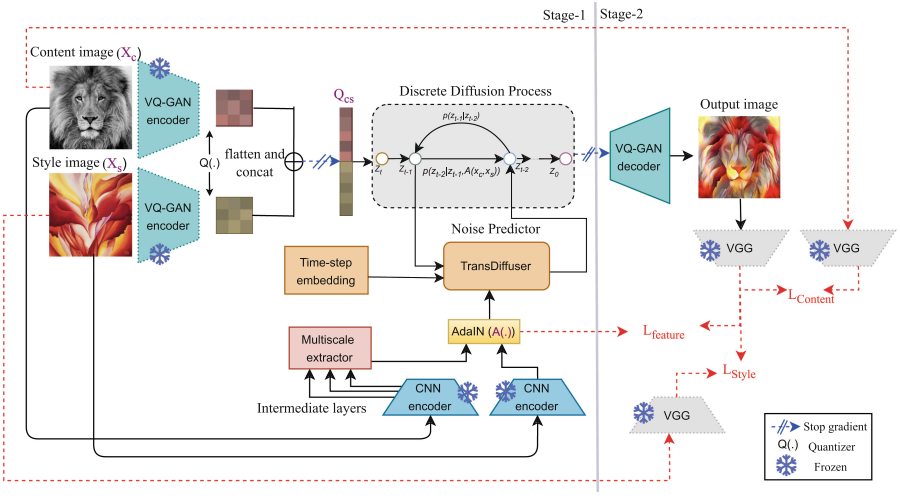


Fig. 2. The architecture of the proposed method. The content and style images are encoded using a pretrained VQ-GAN encoder. The encoded input is passed through the diffusion prior conditioned on the AdaIN [20] features. VQ-GAN decoder is then used to obtain the resultant image. The dotted line indicates that the diffusion prior is trained separately from the decoder.

Stage 1: In stage 1, both style and content images are encoded into continuous latent vectors using a VQ-GAN encoder trained on the OpenImages dataset [26]. These vectors are then projected to the closest codebook item in the discrete latent space. This mapping of continuous vectors to the adjacent discrete codebook vectors is known as quantization (illustrated as Q(.) in Fig. 2). The discretized nature of these vectors facilitates the grouping of similar data points, enhancing the subsequent diffusion sampling within a confined vector space.

Once quantized, vectors are then flattened, concatenated, and sent to the TransDiffuser (Sect. 3.1). Inside the TransDiffuser, these discrete vectors go through the diffusion process, which is influenced by AdaIN features (Sect. 3.2). The outcome of this process is refined denoised features. These features proceed to the next stage (stage 2), where they play a pivotal role in reconstructing the final stylized image.

Stage 2: It uses a pre-trained VQ-GAN decoder trained on OpenImages. This decoder takes improved discrete features from the TransDiffuser as input and creates a stylized picture as output. In particular, when the Stage 2 decoder is being fine-tuned, gradients from Stage 1 are blocked. This is shown in Fig. 2 by the blue arrow. The Stage-2 decoder incorporates a perceptual loss mechanism. This involves calculating the L1 distance between the hidden representation of the ground truth image and the image generated by the decoder. To do this, a pre-trained VGG model is used to derive these hidden representations for both the ground truth and the generated image. A perceptual loss term, denoted as L_{style}, is computed between a style image and the generated image. Similarly, another term, $L_{content}$, is computed between a content image and the generated image. Together, these losses ensure that small differences at the pixel level have less effect on the network. This strikes a balance between keeping the information and sharing the desired style. This method avoids excessive stylization or understylization in the final image. We also compute a $L_{feature}$ loss between AdaIN features and the generated image. This loss ensures that the stylized image retains its original content while adding flair. Formulation of these losses is given in Sect. 3.3

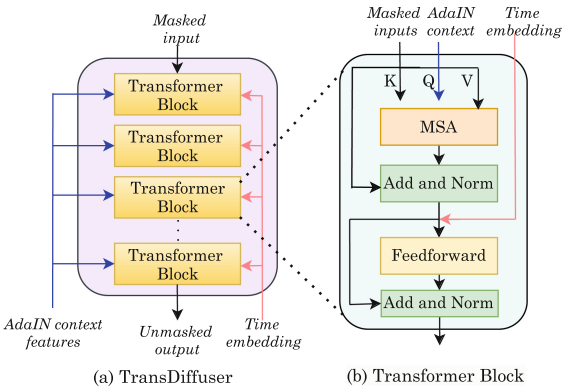


Fig. 3. (a) The proposed TransDiffuser architecture consists of transformer blocks stacked on each other. The attention query is obtained from the AdaIN block (Sect. 3.2). (b) Transformer blocks follow the traditional architecture [42] except for the querying of the AdaIN features.

3.1 TransDiffuser

Figure 3 shows the architecture of TransDiffuser. It is designed to model discrete diffusion processes on quantized vectors. Gu et al. [15] propose use of diffusion to model the discrete vector-quantized latent space of VQ-GAN. We extend this to style transfer. In TransDiffuser, we take a quantized vector (Q_{cs}) as input.

During forward diffusion, this input vector (Q_{cs}) undergoes a gradual corruption process orchestrated by Markov chain $p(z_{t-1}|z_{t-2})$. To achieve this, tokens in z_{t-2} are randomly masked. This iterative process unfolds across a fixed number of time steps (t), generating a sequence of latents (z_1, ..., z_t) that progressively accumulate noise. After the forward process, as visualized in Fig. 2, the reverse process comes into play, starting with the noisy latent variable z_t. This reverse process sequentially removes noise from the latent variables, eventually reconstructing the original data (z_0). Throughout this reverse process, AdaIN features ($A(X_c, X_s)$) are injected into each network block ($p_\theta(z_{t-2}|z_{t-1}, A(X_c, X_s))$). These AdaIN features act as conditional cues for the diffusion process, effectively managing the equilibrium between content and style.

To train TransDiffuser, we utilize the MLM and ELBO loss functions [7, 15]. The MLM loss facilitates the reconstruction of masked tokens, while the ELBO loss captures the probabilistic nature of content and style representations. This dual-loss approach ensures that the generated images exhibit a diverse and meaningful range of possible outcomes, striking a balance between style and content features. The resulting images maintain their original content while adopting the desired style.

3.2 AdaIN Feature Extraction

Within the context of our approach to reverse diffusion, we establish the necessary conditions by harnessing both the input style image (x_s) and content image (x_c). A pivotal step involves crafting a feature map that amalgamates insights from both the style and content images. This is achieved through the utilization of pretrained CNN encoders, VGG-16, in our case. To extract relevant content information, we subject the content image to the VGG encoder, extracting feature maps from specific layers such as 'conv1_2', 'conv2_2', 'conv3_2', and 'conv4_2'. We employ a multiscale extractor to adapt these extracted feature maps for compatibility with the subsequent AdaIN block. This selection of layers is particularly significant due to their ability to encapsulate high-level content details within the image [14]. Likewise, the style image is also processed through the VGG encoder to extract feature maps from its final layer. These extracted feature maps from both the content and style images are then input into the AdaIN block. AdaIN aligns the statistical characteristics of the style and content features. These processed features are subsequently utilized during the reverse process in the TransDiffuser decoder, as detailed in Sect. 3.1. Notably, the introduction of a parameter α, in combination with the AdaIN features, enables precise control over the infusion of style into the resulting stylized image. The influence of using alternative style-content CNN encoders, besides VGG-16, is explored in Sect. 5.

3.3 Loss Functions

We optimized our network using a combination of four loss functions as follows.

1. Diffusion Loss (L_{diff}): The diffusion loss measures the ability of the model to reverse the diffusion process and generate a realistic quantized vector from noise. This loss is often implemented as a denoising score matching or noise prediction loss. During the forward diffusion process, we diffuse the quantized vector from VQ-GAN to get $p(x_t|x_0)$. This vector is passed through the Trans-Diffuser module (P_θ), which is conditioned on the AdaIN features obtained from $A(X_{se}, X_{ce})$. Here, X_{se} and X_{ce} are the representations from the pre-trained CNN model when we pass the style I_s and content I_c images, respectively. At stage 1, we compute the diffusion loss as follows:

$L_{diff} = -\log(P_\theta(p(x_t|X_0), t, A(X_{se}, X_{ce})))$

In stage 2, the predicted quantized vector from P_θ is passed to the VQ-GAN decoder D to get the final stylized image ($\hat{x}$). During this training stage, we block the gradient from Stage 2 to Stage 1 and only train the decoder ($D(.)$) using the following losses:

2. Style Loss (L_{style}): Measures the difference between the style representations of the style image and the representations from the generated image through pretrained vgg-network. This ensures the output image adopts the stylistic elements (e.g., textures, colors, patterns) of the style image.

$L_{style} = \|vgg(I_s) - vgg(\hat{x})\|$

3. Content Loss (L_{cont}): Measures the difference between the feature representations of the content image and the generated image.

$L_{cont} = \|vgg(I_c) - vgg(\hat{x})\|$

4. Feature Loss (L_{feat}): Computes the L1 loss between the AdaIN features and the VGG representations of the generated image. AdaIN dynamically adjusts the normalization parameters (mean and variance) of the feature maps based on the statistics of the input image. This ensures that the normalization process is tailored to the specific content and style images used. This dynamic adjustment helps better align the feature statistics of the generated image with those of the style image, leading to a more faithful style transfer.

$L_{feat} = \|A(X_{se}, X_{ce}) - vgg(\hat{x})\|$

During the training stage 2, we freeze the VGG encoder and AdaIN A(.) encoder is unfrozen.

4 Experiments and Results

Dataset: We select 100,000 images from the COCO dataset as content images and 78,669 images from the WikiArt dataset [38] as style images. We employ a many-to-many strategy to create 1 million content and style image pairs. From these pairs, we randomly select 900,000 pairs for training and use 20,000 and 80,000 pairs for validation and testing, respectively. We employed following metrics: (1) GM for faithful style adoption (2) SSIM for image similarity (3) LPIPS for perceptual resemblance and (4) PD for measuring perceptual dissimilarity.

Experimental Settings: We trained stage-1 and stage-2 models separately on two NVIDIA A100 GPUs with 40 GB VRAM each. The stage-1 is trained using the AdamW optimizer with a learning rate of 1e−5. Stage-2 is trained using the Adam optimizer with a learning rate of 1e−4. The stage-1 TransDiffuser utilizes 25 diffusion steps. The network uses six TransDiffuser blocks. Training is conducted for 140,000 steps with a batch size of 32 for both the stage-1 and stage-2. Inference latency is reported on RTX 3080 Ti GPU.

4.1 Quantitative Results

As shown in Table 1, D^{2}Styler method outperforms all previous techniques on nearly all the metrics, demonstrating its robust capability in style transfer tasks. D^{2}Styler also shows relatively small inference time. D^{2}Styler successfully combines effectiveness and efficiency, making it a leading solution in style transfer.

The DiT method, while slightly lagging behind D^{2}Styler in most metrics, achieves a slightly better performance in the Gram Matrix metric. This might be due to its training on a larger and more diverse dataset and its greater complexity in terms of model parameters. We have taken a pretrained DiT and finetuned it on our dataset. D^{2}Styler uses a pretrained VQ-GAN, however, the dataset on which DiT has been pretrained is larger than the one on which VQ-GAN has been pretrained.

Table 1. Quantitative results

Methods	SSIM ↑	PD ↓	GM ↑	LPIPS ↓	Inference Time (Sec) ↓
CVPR'16 [14]	0.5412	21.22	13.21	0.34	13.56
CVPR'21 [25]	0.6702	19.97	14.09	0.29	1.72
StyleFlow [13]	0.7211	18.76	15.66	0.27	1.12
ACM MM'21 [44]	0.7376	19.09	15.98	0.21	1.42
Cartoon-Flow (MM'22 [28])	0.7422	17.48	16.07	0.18	1.37
StyTr2 (CVPR'22 [6])	0.7778	16.32	17.46	0.19	0.93
SIGGRAPH'22 [45]	0.7501	17.37	16.01	0.17	1.19
InST (CVPR'23) [46]	0.7347	18.09	15.21	0.10	1.72
AdaAttN (ICCV'21) [32]	0.7656	16.77	17.21	0.09	**0.88**
ArtFlow (CVPR'21) [2]	0.7512	16.91	17.29	0.13	1.18
Stable-Diffusion (CVPR'22) [36]	0.7789	17.77	18.21	0.07	4.21
DiT (ICCV'23) [34]	0.7801	17.87	**18.98**	0.06	5.78
Ours w/o Encoder (Fig. 8(d))	0.7757	17.82	16.98	0.14	2.85
Ours w/o AdaIN features (Fig. 8(e))	0.7592	19.92	14.21	0.21	2.54
Ours (Fig. 8(c))	**0.7886**	**15.66**	18.89	**0.04**	2.92

4.2 Qualitative Results

As depicted in Fig. 4, D^{2}Styler effectively preserves the original content of the images while imbuing them with the desired artistic styles from the style

images. The method maintains the original image's underlying structure and details while adopting the reference image's style, producing high-quality, visually appealing images. This capability is evident across various examples shown, where the essence and details of the original images are preserved, yet the style is convincingly and beautifully applied. This includes maintaining structural information without sacrificing the tiny details found in the style images.

StyTr2 is limited in accurately mapping the reference image's style onto the content image, producing unsatisfactory results. Cartoon-Flow, while excelling at maintaining the visual integrity of the content image, occasionally fails to adapt the reference image's aesthetic to the output image. For instance, Cartoon-Flow can preserve the facial structure of Brad Pitt (content image) in the output image, but it cannot account for the sketch style present in the style image, hence its output image is devoid of the desired artistic effect.

Both Cartoon-Flow and StyTr2 struggle to incorporate the color information present in the style image, as demonstrated in the third and fourth rows of Fig. 4. They are unable to generate the color information of the lion images in the style image. Similarly, for the images in the fifth row, both models produce an image

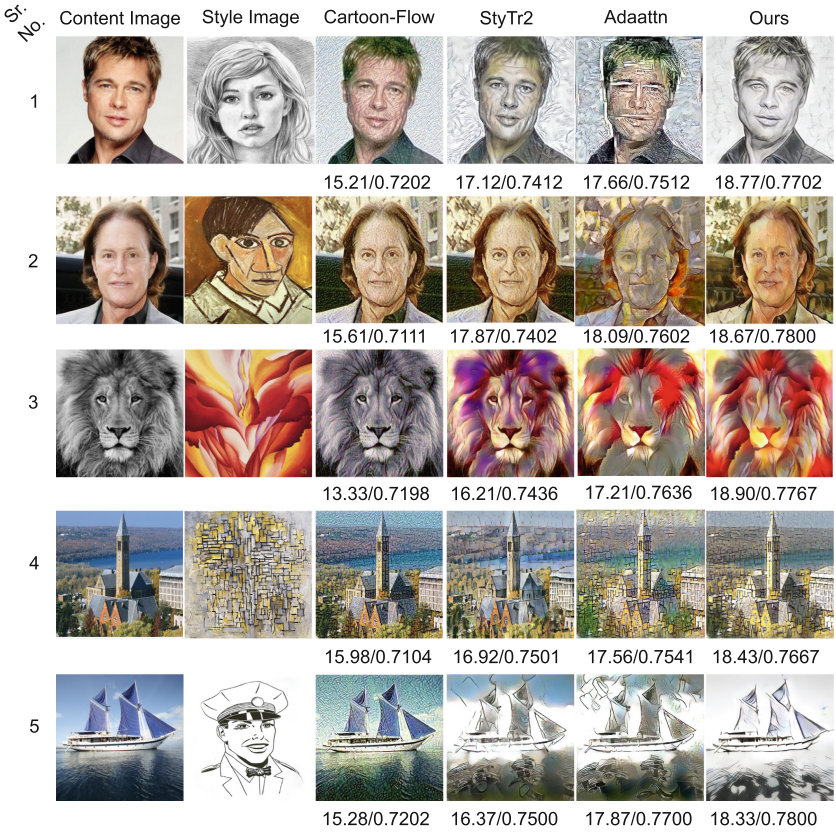


Fig. 4. Qualitative results. The numbers below images show GM (↑) and SSIM (↑).

with the same structural quality as the content image but fail to produce the yellow box texture present in the style image on the output image.

AdaAttN provides the cross attention between the style and the content features, however, it cannot properly retain similar content due to feature collapse. The use of AdaIN features as a condition to the Transdiffuser block helps in a significant boost in performance. AdaIN features are crucial for the effectiveness of style transfer methods because they separate content and style by adjusting the mean and variance of content features to match those of style features. This dynamic adjustment allows the network to adapt to a wide range of styles efficiently, enhancing the quality and visual appeal of the transfer. AdaIN simplifies the training process by focusing on normalization parameters rather than complex style representations, providing better control over the degree of stylization. Without AdaIN, style transfer methods would struggle to achieve the same level of performance, flexibility, and scalability. From Fig. 4, we note that for most images, AdaAttN cannot preserve the content features. In contrast, D^{2}Styler proves to be superior at capturing tiny details in the style image and accurately transferring them to the content image without sacrificing the structural information of the original image. Additionally, D^{2}Styler can robustly handle a wide range of styles and produce high-quality outputs that accurately represent the style of the reference image. Our method achieves comparable text alignment to the Diffusion-based methods for generating content images, i.e., StyTr2. This indicates that our method does not compromise the original style control capabilities of SD while learning the style of the reference images. The substantial advantage reflected in the image quality metric compared to all other methods corroborates the practicality of our approach. In summary, D^{2}Styler achieves an optimal balance between style image fidelity and content image similarity with the most pleasing image quality.

Figure 5 presents the qualitative results of D^{2}Styler on the COCO dataset, effectively demonstrating the capabilities of D^{2}Styler. For instance, the second row illustrates the application of a classical painting style, which imparts a rustic, textured effect reminiscent of brushstrokes and the color palette of the original artwork. Similarly, the pencil sketch style, shown in another row, transforms content images into monochromatic drawings, emphasizing lines and shading to create a hand-drawn appearance. These results underscore the versatility and robustness of D^{2}Styler in blending various artistic styles with diverse content images, while preserving the essential characteristics of both. The model's ability to maintain the structural integrity of the content while accurately reflecting the stylistic nuances of the artistic images demonstrates its potential applications in artistic creation and advanced image processing.

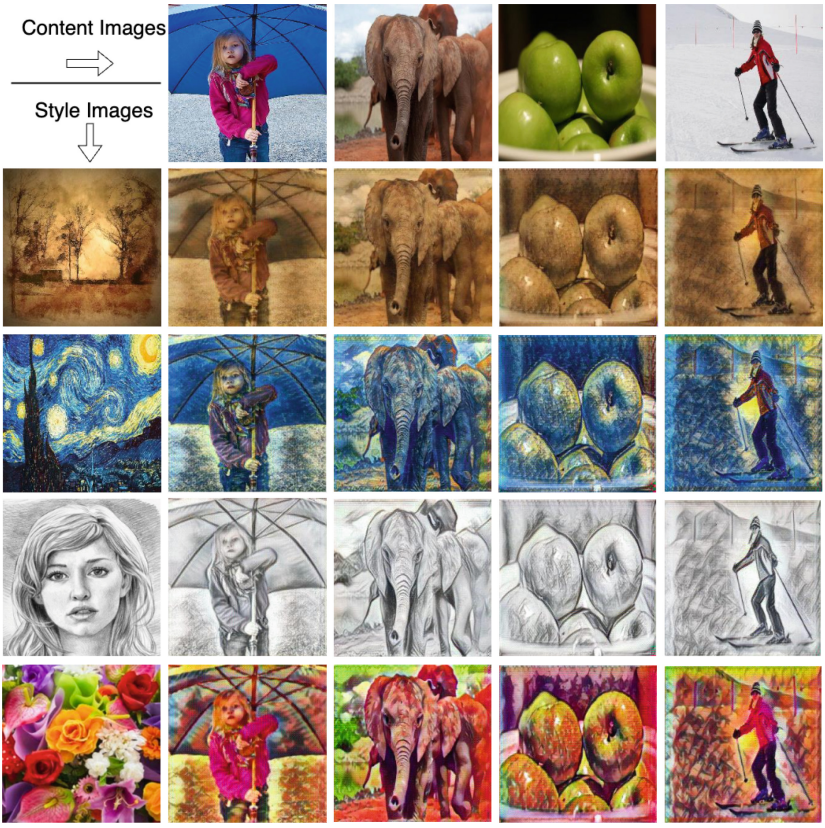


Fig. 5. Qualitative results of D^{2}Styler on COCO dataset

Figure 6 further compares qualitative results of various techniques. InST (column #3) [46] often lacks coherence in preserving the content structure. It fails to semantically transfer the color in a one-to-one correspondence. They used an additional tone transfer module [20] to align the color of the content and reference images. Different methods have different preferences for retaining the colors of the content image.

StyleTr2 (column #5) provides reasonable style transfer results but sometimes over-emphasizes the style, leading to over-stylization. This happens because of an imbalance of content loss and style loss. If the weighting for the style loss is too high relative to the content loss, the model might prioritize transferring stylistic elements over preserving the original content structure, resulting in over-stylization. On the other hand, AdaAttN utilizes cross-attention mechanisms for style transfer but occasionally suffers from feature collapse, affecting content preservation (e.g., column #6 row #8). The per-point basis of AdaAttN leads to style degeneration, thus the stylized output is not consistent with the input reference. The content leak issue usually occurs in the stylization process

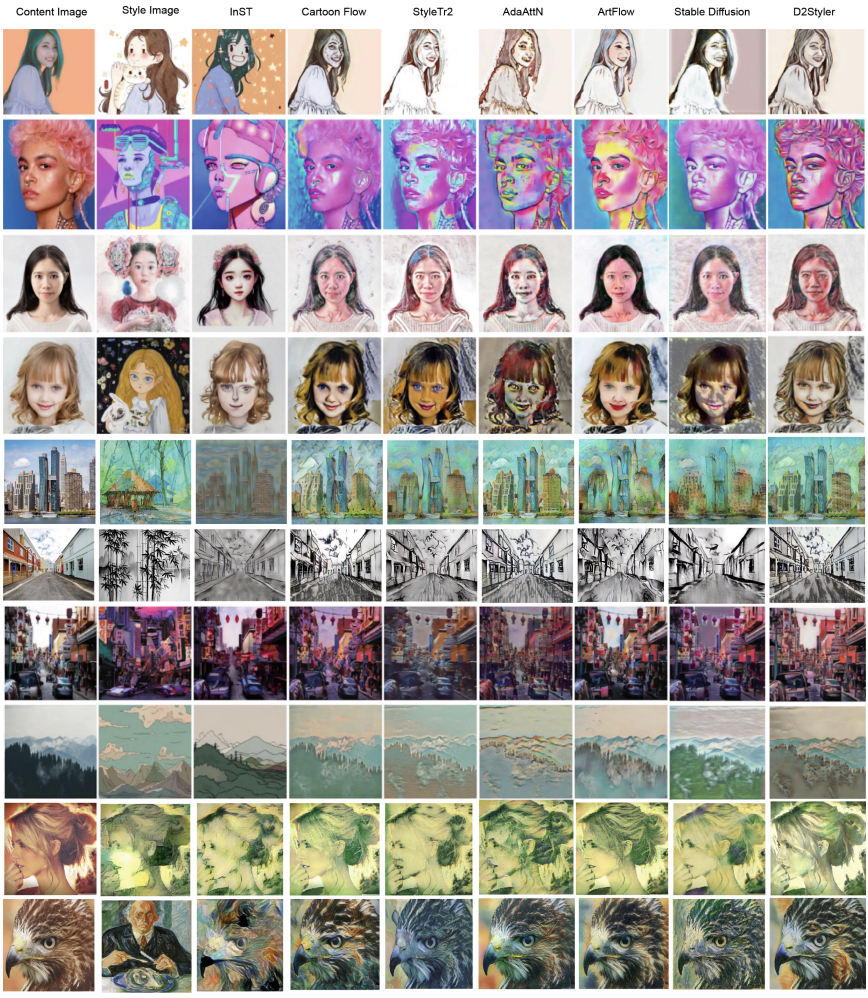


Fig. 6. Qualitative comparison

because CNN-based feature representation may not sufficiently capture details in the image content.

Furthermore, the robustness and generated visual effects of CartoonFlow (column #4) [28] may degrade due to the limited capability of the feature representation. By contrast, D^{2}Styler leverages the capability of transformer-based architecture to capture long-range dependencies, hence, it significantly alleviates the content leak issue. The flow-based model has limited capability of feature representation, hence, the ArtFlow (column #7) [2] results generally suffer from insufficient or inaccurate style. The border of stylized images may present undesirable patterns due to numerical overflow.

Stable Diffusion (column #8) generates high-quality images, albeit with increased computational time (Table 1), due to the significant effort required to interpolate style and content features in its latent space to achieve perfect stylization. In contrast, D^{2}Styler demonstrates superior performance in both style fidelity and content preservation. This superiority stems from our use of diffusion in the Vector Quantized (VQ) space, where content and style features are more easily searchable and can be interpolated more effectively. By leveraging the VQ space, D^{2}Styler efficiently balances style and content, resulting in high-quality images with reduced computational complexity.

4.3 Versatility of D^{2}Styler

Controlling the Style. To control the amount of style output, D^2Styler introduces a weight parameter named α ($\alpha \in [0, 1]$) for the AdaIN context features. As shown in Fig. 7, the amount of style in the image is proportional to the value of the α parameter.

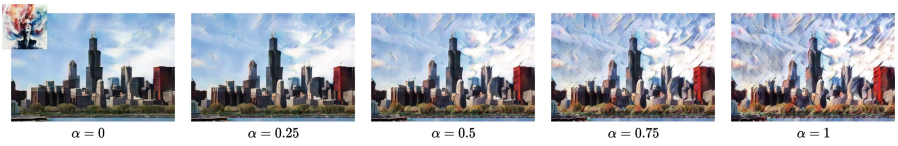


Fig. 7. D^2Styler has a parameter α. By changing α, the AdaIN context features can be varied to control the output image style, as shown in this image. The style image is in the inset.

Use of the VQ-GAN Encoder at Inference. At inference, we can use either (1) the samples from the diffusion mask prior or (2) the VQ-GAN encoded version of the input content and style image. The former gives more priority to the style texture, whereas the latter retains more content, leading to more visually pleasing images, as shown in Fig. 8.

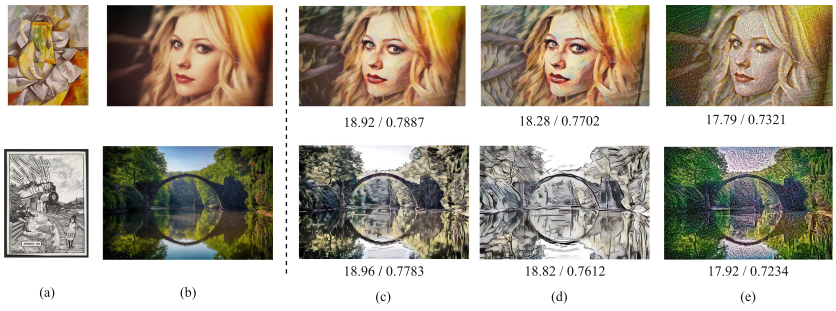


Fig. 8. Input (a) style and (b) content images. D^2Styler results (c) on using the encoder during inference. (d) on starting from the noise distribution during inference. (e) without using the AdaIN features as a context to the TransDiffuser.

Using the encoder also results in a slightly higher (i.e., better) SSIM metric as opposed to starting from the diffusion mask prior. Using the encoder gives a head start to the denoising process, as it contains some prior information about the content. By contrast, this prior information is absent when we start from the mask tokens themselves. Table 1 reports GM and SSIM scores for both types of inference strategies.

Multi-style Transfer: We realize multi-style transfer by passing a linear combination of AdaIN features for each style-content combination. Each image's contribution can be controlled by using a weight for each style-content feature being combined. An example generation is shown in Fig. 9. Evidently, the AdaIN layer serves as a context for the diffusion model, and its features can be used to control the generation of a stylized output image.

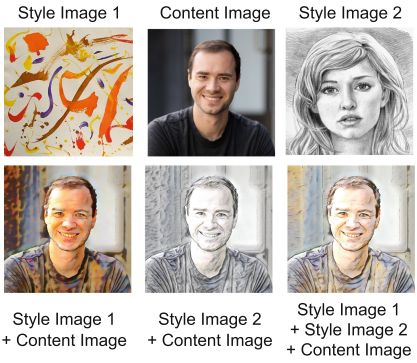


Fig. 9. Transferring the style of a single image and multiple images while preserving the content of a single content image.

5 Ablation Study

1. Effect of CNN Encoders: Table 2 shows the results with different CNN encoders in D^{2}Styler. Notice that using CNN encoders such as VGG, ResNet and EfficientNet leads to better results than use of transformers. NST transfers the style of a style image to a content image while keeping its spatial information. Since transformers do not incorporate locality bias inherently, they often fail to learn hierarchical representations from an input image effectively. Due to this, the model faces challenges in decoupling style and content information from an image, leading to poor performance in NST.

Table 2. Ablation based on CNN Encoder

	SSIM ↑	GM ↑	LPIPS ↓
VGG-16	**0.7886**	**18.89**	**0.04**
Resnet-32	0.7721	16.32	0.08
EfficientNet-B0	0.751	17.01	0.08
ViT	0.7631	15.21	0.09
PVT	0.7419	16.98	0.12

2. Effect of Loss Functions: Table 3 shows the impact of various loss functions. The best scores are obtained by using a combination of $L_{feature}$, L_{style}, and $L_{content}$ losses, because this combined loss considers both style data from the encoder model and high-level features of the content. In contrast, using only L_{style} (style loss) neglects content preservation, and using only $L_{content}$ (content loss) overlooks style consistency. Adding $L_{feature}$ (feature loss) to L_{style} and $L_{content}$ makes the results even better by forcing a close match between the output features and the normalization data of the Adaptive Instance Normalisation (AdaIN) layer. Notably, using only $L_{feature}$ loss leads to better performance than most previous techniques [13,14,25,28,44,45]. This validates the efficacy of our feature-matching approach. The above insights can be confirmed from Fig. 10, which illustrates the output images with various loss functions.

Table 3. Ablation based on loss functions

	SSIM ↑	GM ↑	LPIPS ↓
$L_{feature} + L_{style} + L_{content}$	**0.7886**	**18.89**	**0.04**
$L_{style} + L_{content}$	0.7677	17.23	0.07
$L_{content}$	0.7501	16.09	0.09
L_{style}	0.7421	17.11	0.10
$L_{feature}$	0.7709	17.32	0.12
$L_{style} + L_{feature}$	0.7732	18.00	0.08
$L_{content} + L_{feature}$	0.7821	17.67	0.08

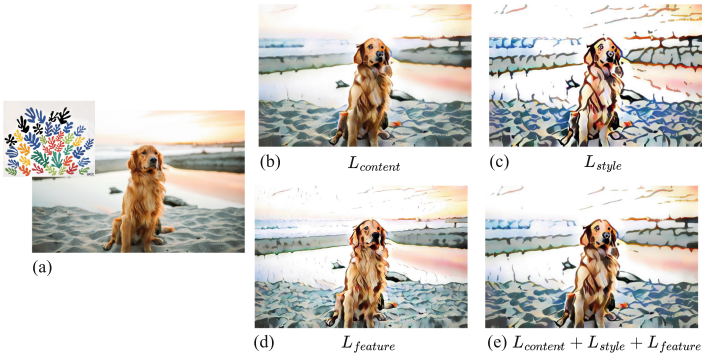


Fig. 10. (a) Input style and content images. D^2Styler results on using (b) only content loss (c) only style loss (d) only the proposed feature loss. (e) all three losses.

3. Effect of the number of TransDiffuser Blocks: We evaluate three versions of D^{2}Styler, namely small, medium, and large, which utilize two, four and six TransDiffuser blocks, respectively. With the increasing number of blocks, the network can more accurately model the style transfer process, which improves the scores (Table 4). Interestingly, even small and medium versions of D^{2}Styler can outperform most previous techniques (refer Table 1). This proves the efficacy of our technique.

Table 4. Effect of changing the number of TransDiffuser blocks

	SSIM ↑	GM ↑	LPIPS ↓
N = 6	**0.7886**	**18.89**	**0.04**
N = 4	0.7798	18.32	0.06
N = 2	0.7762	17.66	0.07

4. Effect of Changing the Number of Diffusion Steps: Table 5 shows the results with different numbers of diffusion steps. Clearly, increasing the number of diffusion steps consistently improves image quality and structural similarity, although the returns become marginal after 25 steps. Our method demonstrates that with just 5 diffusion steps, it matches the inference times of GAN and flow-based baselines (refer to Table 1), and also outperforms them in all quantitative metrics. This indicates our method's robustness and superior performance compared to previous baselines. Figure 11 shows the output images with different numbers of diffusion steps.

Table 5. Ablation Based on Diffusion Steps

#Steps	SSIM ↑	GM ↑	LPIPS ↓	Inference Time (sec) ↓
5	0.7441	17.02	0.07	**1.93**
10	0.7512	17.78	0.06	2.31
15	0.7663	18.00	0.06	2.54
20	0.7702	18.88	0.05	2.74
25	0.7886	**18.89**	**0.04**	2.92
50	**0.7891**	**18.89**	**0.04**	3.28

Fig. 11. (a) Input style and content images. $D^2Styler$ results with (b) 5, (c) 15 and (d) 25 diffusion steps.

6 Conclusion

We propose a novel AST technique, named D^{2}Styler, by combining the benefits of discrete diffusion with discrete representational capacity of VQ-GANs. We propose a novel way of guiding the diffusion process by incorporating Adaptive Instance Normalisation (AdaIN) features. This allows transferring features from the style image to the content image without bias. D^{2}Styler produces style-transferred images that are both visually appealing and accurate to the original content image in terms of their semantic significance. D^{2}Styler outperforms previous techniques and solves the problems of mode collapse, over-stylization, and under-stylization. Future work will focus on generalizing our technique to a wide range of image-processing tasks, e.g., editing and synthesis.

References

1. Ahn, N., et al.: DreamStyler: paint by style inversion with text-to-image diffusion models. AAAI **38**(2), 674–681 (2024)
2. An, J., et al.: ArtFlow: unbiased image style transfer via reversible neural flows. In: CVPR, pp. 862–871 (2021)
3. Austin, J., et al.: Structured denoising diffusion models in discrete state-spaces. In: NeurIPS, vol. 34, pp. 17981–17993 (2021)
4. Chandramouli, P., Gandikota, K.V.: LDEdit: towards generalized text guided image manipulation via latent diffusion models. In: BMVC, vol. 1, p. 2 (2022)
5. Chang, H., et al.: Muse: text-to-image generation via masked generative transformers. In: ICML, pp. 4055–4075 (2023)

6. Deng, Y., et al.: StyTr2: image style transfer with transformers. In: CVPR, pp. 11326–11336 (2022)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. ArXiv **abs/1810.04805** (2019)
8. Ding, M., et al.: CogView: mastering text-to-image generation via transformers. NeurIPS **34**, 19822–19835 (2021)
9. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. arXiv preprint arXiv:1610.07629 (2016)
10. Elad, M., Milanfar, P.: Style transfer via texture synthesis. IEEE Trans. Image Process. **26**(5), 2338–2351 (2017)
11. Esser, P., et al.: Taming transformers for high-resolution image synthesis. In: CVPR, pp. 12873–12883 (2021)
12. Everaert, M.N., et al.: Diffusion in style. In: ICCV, pp. 2251–2261 (2023)
13. Fan, W., Chen, J., Ma, J., Hou, J., Yi, S.: StyleFlow for content-fixed image to image translation. Arxiv (2022)
14. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR, pp. 2414–2423 (2016)
15. Gu, S., et al.: Vector quantized diffusion model for text-to-image synthesis. arXiv preprint arXiv:2111.14822 (2021)
16. Hamazaspyan, M., Navasardyan, S.: Diffusion-enhanced patchmatch: a framework for arbitrary style transfer with diffusion models. In: CVPR, pp. 797–805 (2023)
17. Hertz, A., et al.: Prompt-to-prompt image editing with cross attention control. arXiv preprint arXiv:2208.01626 (2022)
18. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Adv. Neural. Inf. Process. Syst. **33**, 6840–6851 (2020)
19. Hoogeboom, E., et al.: Argmax flows and multinomial diffusion: learning categorical distributions. In: NeurIPS, vol. 34, pp. 12454–12465 (2021)
20. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV, pp. 1510–1519 (2017)
21. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43
22. Kim, G., et al.: DiffusionClip: text-guided diffusion models for robust image manipulation. In: CVPR, pp. 2426–2435 (2022)
23. Kim, J., Kim, M., Kang, H., Lee, K.: U-GAT-IT: unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. arXiv preprint arXiv:1907.10830 (2019)
24. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
25. Kotovenko, D., Wright, M., Heimbrecht, A., Ommer, B.: Rethinking style transfer: from pixels to parameterized brushstrokes. In: CVPR, pp. 12196–12205 (2021)
26. Kuznetsova, A., et al.: The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. Arxiv (2018)
27. Kwon, G., Ye, J.C.: Diffusion-based image translation using disentangled style and content representation. arXiv preprint arXiv:2209.15264 (2022)
28. Lee, J., et al.: Cartoon-flow: a flow-based generative adversarial network for arbitrary-style photo cartoonization. In: International Conference on Multimedia (2022)

29. Li, J., et al.: BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. ICML, pp. 19730–19742 (2023)
30. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. Adv. Neural Inf. Process. Syst. **30** (2017)
31. Li, Y., et al.: Diversified texture synthesis with feed-forward networks. In: CVPR, pp. 3920–3928 (2017)
32. Liu, S., et al.: AdaAttN: revisit attention mechanism in arbitrary neural style transfer. In: ICCV, pp. 6649–6658 (2021)
33. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning, pp. 8162–8171. PMLR (2021)
34. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: ICCV, pp. 4195–4205 (2023)
35. Ramesh, A., et al.: Zero-shot text-to-image generation. ICML, pp. 8821–8831 (2021)
36. Rombach, R., et al.: High-resolution image synthesis with latent diffusion models. In: CVPR, pp. 10684–10695 (2022)
37. Saharia, C., et al.: Palette: image-to-image diffusion models. In: ACM SIGGRAPH, pp. 1–10 (2022)
38. Saleh, B., Elgammal, A.: Large-scale classification of fine-art paintings: learning the right metric on the right feature. arXiv preprint arXiv:1505.00855 (2015)
39. Sohn, K., et al.: StyleDrop: text-to-image synthesis of any style. NeurIPS **36** (2024)
40. Ulyanov, D., et al.: Texture networks: feed-forward synthesis of textures and stylized images. arXiv:1603.03417 (2016)
41. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Adv. Neural Inf. Process. Syst. **30** (2017)
42. Vaswani, A., et al.: Attention is all you need. In: NeurIPS, vol. 30 (2017)
43. Wang, Z., Zhao, L., Xing, W.: StyleDiffusion: controllable disentangled style transfer via diffusion models. In: ICCV, pp. 7677–7689 (2023)
44. Zhang, R., et al.: Image re-composition via regional content-style decoupling. In: ACM International Conference on Multimedia, pp. 3–11 (2021)
45. Zhang, Y., et al.: Domain enhanced arbitrary image style transfer via contrastive learning. In: ACM SIGGRAPH (2022)
46. Zhang, Y., et al.: Inversion-based style transfer with diffusion models. In: CVPR, pp. 10146–10156 (2023)

Mask-ControlNet: Higher-Quality Image Generation with an Additional Mask Prompt

Zhiqi Huang[1], Huixin Xiong[2], Haoyu Wang[1], Longguang Wang[3], and Zhiheng Li[1(✉)]

[1] Tsinghua University, Beijing, China
zhhli@mail.tsinghua.edu.cn
[2] Megvii, Beijing, China
[3] Sun Yat-sen University, Shenzhen, China

Abstract. Text-to-image generation has witnessed great progress, especially with the recent advancements in diffusion models. Since texts cannot provide detailed conditions like object appearance, reference images are usually leveraged for the control of objects in the generated images. However, existing methods still suffer limited accuracy when the relationship between the foreground and background is complicated. To address this issue, we developed a framework termed Mask-ControlNet by introducing an additional mask prompt. Specifically, we first employ large vision models to obtain masks to segment the objects of interest in the reference image. Then, the object images are employed as additional prompts to facilitate the diffusion model to better understand the relationship between foreground and background regions during image generation. Experiments show that the mask prompts enhance the controllability of the diffusion model to maintain higher fidelity to the reference image while achieving better image quality. Comparison with previous text-to-image generation methods demonstrates our method's superior quantitative and qualitative performance on the benchmark datasets.

Keywords: Image Generation · Diffusion Model · Object Reconstruction

1 Introduction

Recently, text-to-image generation has achieved remarkable progress with numerous models being developed, ranging from GAN [9], VQGAN [7] and DALL-E [31] to current diffusion models [29,35]. Despite their promising results, text-to-image models still suffer limited capability in fine-grained control of the synthetic images since text prompts cannot provide details like the object's appearance and spatial layout. To remedy this, extensive studies have been conducted to control text-to-image models using additional prompts like spatial masks [1,13], image editing instructions [3,17,22], and other formats [8,25,46].

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 83–97, 2025.
https://doi.org/10.1007/978-3-031-78172-8_6

In numerous real-world applications, mimicking the appearance of objects in a reference image while changing the composition and contexts is under great demand. To tame text-to-image models to meet this requirement, a reference image is provided as an additional prompt [33]. Despite the synthetic images can fit the provided prompts, they still suffer from three major limitations. **(i) Object distortion.** The object in the reference image may not be faithfully transferred to the synthetic image and usually suffers detail losses (the first row of Fig. 1). **(ii) Background overfitting.** The concurrence of the foreground object and its background in the training set (e.g., the table in the second row of Fig. 1) may be overfitted. As a result, the background may also be synthesized in the generated images. **(iii) Foreground-background inharmony.** Despite high object fidelity, the generated image may suffer inharmonious foreground and background (the third row of Fig. 1). Overall, these limitations are attributed to the complicated relationship between the foreground and background, which is not well modeled by the generative model.

Fig. 1. Limitations of existing image generation methods. The synthetic images suffer from object distortion (the first row), background overfitting (the second row) and foreground-background inharmony (the last row).

Intuitively, the foreground and background in the prompt image play different roles during image synthesis. The foreground provides the appearance details of the object while the background helps the network to understand the context of the object and produce harmonious ones in the synthetic image. To better model the relationship between foreground and background, we propose to decouple these two components using an additional mask prompt. With the

great advancements of recent large vision models like SAM [20], the foreground object mask can be easily obtained. Specifically, the reference image is first fed to SAM to produce a mask to segment the object region. Then, the resultant image is concatenated with the reference image as the conditional information for image synthesis. The additional mask prompt facilitates the network to better maintain the object details and model the foreground-background relationship, resulting in higher-quality synthetic images.

The main contributions are summarized as below:

1) We propose a framework termed Mask-ControlNet to achieve higher-quality image generation by introducing an additional mask prompt. With the help of this mask prompt, the foreground and background in the reference image can be decoupled and well-modeled to improve the quality of the synthetic image.
2) We conduct extensive experiments to study our framework. Quantitative and qualitative results show that our framework is able to generate high-quality images with fewer artifacts.

2 Related Work

In this section, we briefly review recent advances in text-to-image generative models and controllable generative models.

Text-to-Image Generative Models. Text-to-image generative models have been studied for decades, with diverse networks being developed, including VAE [18], PixelCNN [28,38], Glow [19], and GAN [9]. Among these methods, GAN stands out as one of the most popular models. Over the past five years, GANs have witnessed substantial progress and are capable of generating high-resolution images up to 1024×1024 pixels or higher [2,15,16]. Despite the theoretical equilibrium, GAN's training is still faced with challenges like instability and mode collapse.

The advent of diffusion models [29,35] offers a stable, high-quality, and controllable solution. DDPM [10] surpassed GAN using a U-net denoising autoencoder. DDIM [36] further improves image synthesis with implicit probabilistic models and continuous-time dynamics. Recently, stable diffusion model further outperforms DDIM by introducing a latent diffusion model [32] to replace pixel-level diffusion models and using an autoregressive flow model to realize the prior distribution of the latent space.

Diffusion models usually introduce interfaces when conducting fine-grained image editing such as color and texture details [17,21,22]. To remedy this, Textual Inversion [8] personalizes the generated content using a small set of images under the same overarching theme. DreamBooth [33] associates target themes with unique identifiers, enabling these identifiers to be embedded in the model's output domain, which synthesizes realistic images of specific themes in differentiated scenarios.

Controllable Generative Models. Despite high quality, diffusion models commonly suffer inferior fidelity to the conditional image. To improve the controllability of diffusion models, more complicated texts are employed to provide detailed descriptions of the image. Liu et al. [25] proposed a semantic diffusion-guided framework that utilizes CLIP [30] or other image-matching models to compute the similarity between the text prompt and the generated images to guide the diffusion model in generating images that better match the semantic requirements. Avrahami et al. [1] introduced a text-driven image editing method that allows the modification of specific shards in an image according to user-provided text.

As the text cannot provide fine-grained control (e.g., object appearance) to the generated images, the results may not meet the requirements of users. To address this issue, ControlNet [43] extends text-to-image diffusion models with conditional control [12,14,24,27,39,45], allowing the generation of images that match diverse types of user-specified prompts. Inpainting is proposed to preserve the essential components in a reference image while extending or enhancing the background. Subsequent methods [4,6,26,41] use a pre-trained unconditional diffusion model as a generative prior, and then achieve conditional generation by sampling randomly shaped unmasked regions during the reverse diffusion process. While these methods perform well on simple objects, they cannot well handle complicated scenes and encounter challenges including object distortion, background overfitting, and foreground-background inharmony.

3 Methodology

Given a reference image of a specific object, our objective is to generate images that maintain high detail fidelity of the object while synthesizing diverse contexts and compositions conditioned on the text prompts.

3.1 Training-Time Framework

As shown in Fig. 2, our framework is built on top of a diffusion model and is trained in a self-supervised manner. First, the input image is fed to the VAE encoder to obtain feature maps F and then the noise is progressively added, resulting in F_t. Here, t represents the number of times noise is added. Afterward, the noisy feature maps F_t are passed to the diffusion model to predict the noise and reconstruct the input image.

In parallel to the main path, our framework has an image branch and a text branch to provide additional conditions for the diffusion model. In the image branch, the input image is first fed to SAM [20] to produce the object mask. Then, the resultant mask is used to segment the object in the image. Next, the concatenation of the object image and the image is passed to an adapter layer. Afterward, a VAE encoder and ControlNet are employed to control the diffusion model to reconstruct the input image. In the text branch, BLIP [23] is adopted to extract textual descriptions of the input image. Then, the extracted text prompt

is fed to CLIP to provide additional control to the diffusion model. Following [43], the features extracted from the text and image prompts are connected to the diffusion model with zero convolution layers.

During optimization, only the adapter layer and the ControlNet are trainable while the diffusion model is frozen. The loss function used is defined as:

$$L = E_{z_0,t,c_t,c_f,\epsilon\sim N(0,1)}||\epsilon - \epsilon_\theta(z_t, t, c_t, c_f)||_2^2, \tag{1}$$

where z_0 represents the data in the latent space, c_t and c_f are the text condition and the latent condition, respectively.

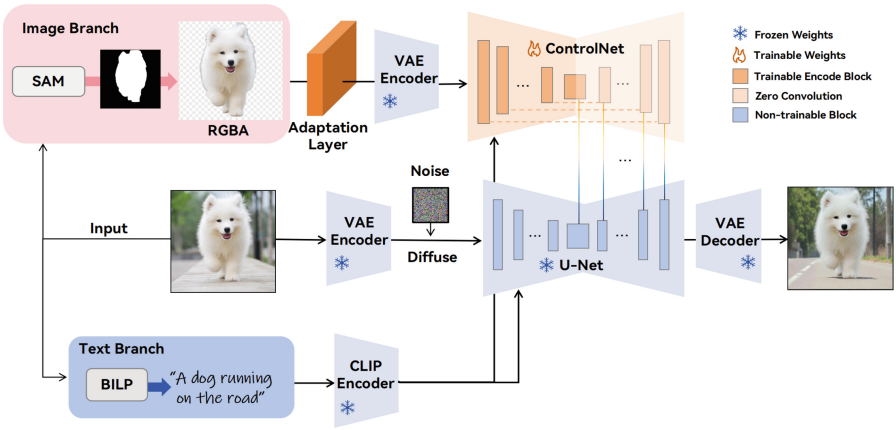


Fig. 2. An illustration of our framework during the training phase.

3.2 Inference-Time Framework

Our framework during inference time is illustrated in Fig. 3. First, the reference image is fed to SAM to produce a mask to segment the object. Then, the concatenation of the object image and the reference image is passed to the VAE encoder. Meanwhile, the text that describes the context of the generated image is fed to CLIP. Next, the features extracted from the image and the text prompt are passed through ControlNet and used as conditions for the diffusion model to synthesize an image from a noise image.

4 Experiments

In this section, we first introduce the experimental setup. Then, we conduct experiments to compare our method with previous approaches. Next, we conduct several ablation studies to demonstrate the effectiveness and robustness of the mask.

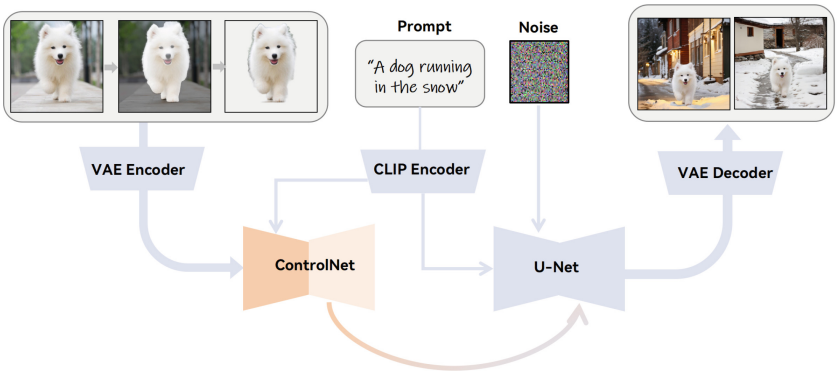


Fig. 3. An illustration of our framework during the inference phase.

4.1 Experimental Setup

During the training phase, we collected 100,000 images from numerous websites using keywords such as people, food, animals, furniture, cosmetics, automobiles, and clothing. In addition, we selected 100,000 images from the SA1B and COCO datasets. After data cleaning and annotation, a total of 130,000 valid images and approximately 300,000 valid masks are obtained as the training set.

During the test phase, the DreamBooth dataset is included for evaluation, which comprises 30 categories like backpacks, toys, dogs, cats, and sunglasses. Each category has about 5–8 images. FID [34], PSNR [37], SSIM [40] and LPIPS [44] are employed to evaluate the generated images. In addition, to measure the object grounding accuracy from the reference image and the text to the generated image, we adopt the CLIP score [30] and DINO score [5].

4.2 Performance Evaluation

(1) Quantitative Results

We compare our method with three representative methods, including DreamBooth [33], ControlNet [43]+LoRA [11], and Inpainting-Anything [42]. These methods are capable of generating images with different backgrounds for specified objects. DINO and CLIP-I are used to evaluate the object fidelity while CLIP-T is employed to calculate prompt fidelity. As shown in Table 1, our method is more faithful to the text prompts with higher CLIP-T scores. Additionally, our method outperforms other methods except ControlNet+LoRA in terms of both CLIP-I and DINO. It should be noted that ControlNet+LoRA requires partial input images for fine-tuning. Although our method does not conduct this fine-tuning process, competitive results are produced, which demonstrates the effectiveness of our method. As compared to DreamBooth, our method achieves much better performance in terms of all metrics.

Then, we compare the quality of images synthesized by different methods in Table 2. As we can see, our Mask-ControlNet surpasses ControlNet+LoRA by

Table 1. Prompt fidelity comparison of images produced by different methods.

Method	CLIP-T↑	CLIP-I↑	DINO↑
DreamBooth [33]	0.171	0.828	0.527
ControlNet [43]+LoRA [11]	0.173	**0.873**	**0.607**
Inpainting-Anything [42]	0.166	0.849	0.423
Mask-ControlNet	**0.175**	0.858	0.593
Real Images	0.176	0.885	0.774

Table 2. Quantitative results achieved by ControlNet+LoRA and Mask-ControlNet.

Method	FID↓	PSNR↑	SSIM↑	LPIPS↓
ControlNet [43]+LoRA [11]	6.161	28.06	0.957	0.031
Mask-ControlNet	**5.172**	**30.67**	**0.958**	**0.022**
Real Images	0.000	INF	1.0	0.00

notable margins on all metrics. Specifically, our Mask-ControlNet demonstrates superior performance in terms of FID and PSNR (5.172 and 30.67, respectively) as compared to ControlNet+LoRA (6.161 and 28.06). This indicates that our method generates images that closely match the distribution of real images and better match the ground truths in terms of pixel-level fidelity.

(2) User Study

We further invited users to evaluate the aesthetics, grounding accuracy, and the realness of the images generated by different methods. Average Human Ranking (AHR) is employed to rank each result on a scale of 1 to 5 (the higher the better). We asked 524 users to answer questionnaires of 20 comparative questions. They were rated for a set of images of the same item generated by different methods. Total 9,212 answers were collected and 37.5% of users had no understanding of image generation models. The results are presented in Table 3.

Table 3. Average User Ranking (AUR) of image quality in terms of aesthetics, accuracy and realness. 1 to 5 indicates worst to best.

Method	Aesthetics↑	Accuracy↑	Realness(%)↑
DreamBooth [33]	3.874	3.217	13.65
ControlNet [43]+LoRA [11]	3.467	3.868	35.53
Inpainting-Anything [42]	3.584	3.508	20.48
Mask-ControlNet	**4.252**	**4.320**	**61.85**

As we can see, our method outperforms other methods by notable margins in terms of all metrics. Particularly, the results synthesized by our method

achieve significantly higher accuracy (4.330 vs. 3.888). This demonstrates that our method can better maintain the object details in the reference images with higher fidelity. In addition, 61.04% of the users cannot distinguish between our results and real images. This further validates the effectiveness of our method to produce images high in realness.

(3) Qualitative Analysis

We further compare the visual quality of the images produced by different methods. We focus on the three challenges as presented in Sect. 1.

(i) Object Distortions

In Fig. 4, it is evident that the results produced by other methods suffer object distortions. For example, in the second row, DreamBooth and ControlNet+LoRA generate incorrect text on the can, while Outpainting produces a distorted bottom. In contrast, our method maintains higher fidelity of the objects and better preserves their details in the generated images.

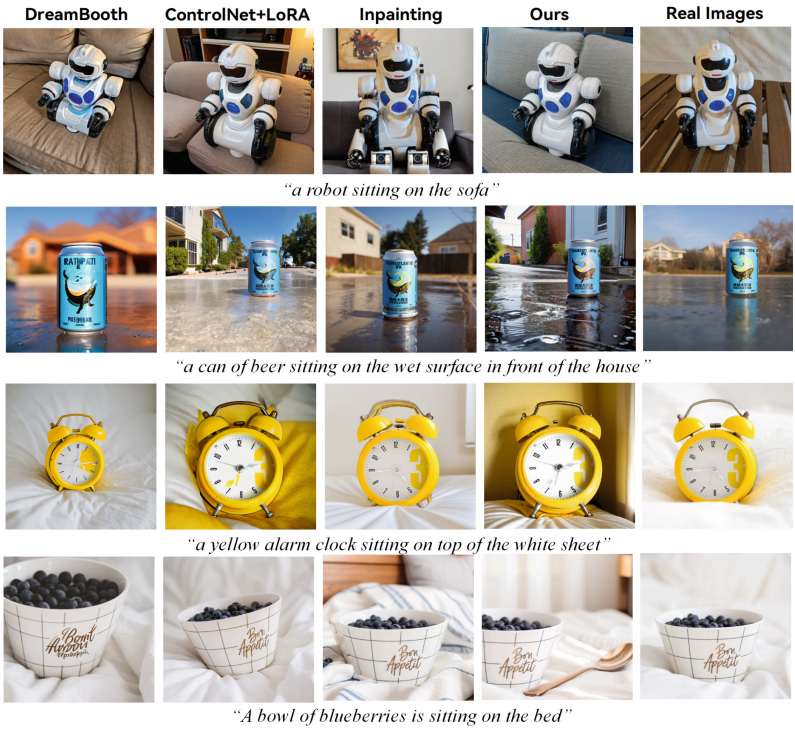


Fig. 4. Synthetic images produced by different methods under the same prompt. From the perspectives of object edges, texture, text, color, etc., our method generates images that are more lossless and closer to reality.

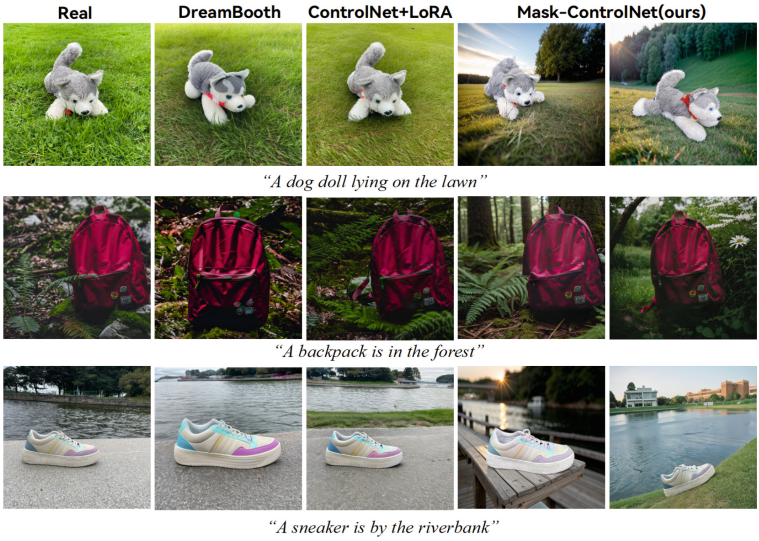


Fig. 5. Background contrast generated under the same prompt. From the figure, it can be seen that our method can generate more diverse backgrounds.

(ii) Background Overfitting

In Fig. 5, we can see that other methods are prone to introducing the background of the object in the reference image to the synthetic images, such as the riverbank in the third row. In contrast, our method can better understand the relationship between foreground and background, thereby generating images with diverse contexts.

(iii) Foreground-background Inharmony

In Fig. 6, we can observe that the foreground and background are more harmonious in our results as compared to other methods. For example, in the first scene (a cat is playing basketball), the inpainting method cannot well understand the text and generates a ball behind the cat. In contrast, our method can better understand the text and synthesize a ball under the foot. For other challenging conditions like occlusions, reflections, and shadows, the relationship between the foreground and the background can also be well handled. These results validate the superiority of the proposed method.

4.3 Model Analyses

(1) Effectiveness of Mask Prompts. To demonstrate the effectiveness of masks, ablation experiments are conducted. Specifically, we develop a variant that directly uses the reference image as the prompt without leveraging the mask to segment the objects. As shown in Table 4, the mask prompt facilitates our method to better maintain the object details in the reference image to achieve higher fidelity scores in terms of both CLIP-I and DINO. Qualitatively, as shown

Fig. 6. Comparison of foreground-background harmony in different scenarios. The images from upper left to lower right present six different interaction scenarios, including contact, refraction, shadow, obstruction, reflection, and dynamic effects.

in Fig. 7, the model without mask exhibits relatively low quality. In contrast, by incorporating a mask as an additional prompt, our method is able to generate images that are more faithful to the reference image (e.g., the T-shirt of the girl in the first row) with higher perceptual quality.

Table 4. Ablation results on mask prompts.

Method	CLIP-I↑	DINO↑
w/ mask	**0.858**	**0.593**
w/o mask	0.812	0.386

(2) Flexibility of Mask Prompts

We further test the flexibility of the mask prompts in our framework. When an image containing multiple objects is employed as the reference image, our framework can flexibly transfer objects of interest to the generated images using different masks, as shown in Fig. 8. It can be observed that our method produces visually promising images using the same reference image with different masks as conditions. Without our mask prompt, the same reference image is used as a condition. As a result, the diffusion model may be confused and cannot well understand the text prompts.

(3) Potential Applications

Mask-ControlNet can be flexibly applied to a variety of scenarios, including but not limited to the following ones:

(1) Data augmentation. As shown in Fig. 9, we can employ our Mask-ControlNet to synthesize additional images for data augmentation. These images are of high quality and realness, which can better improve the data diversity as compared with simple cropping or mixup techniques.

Fig. 7. Images synthesized by our method with and without masks.

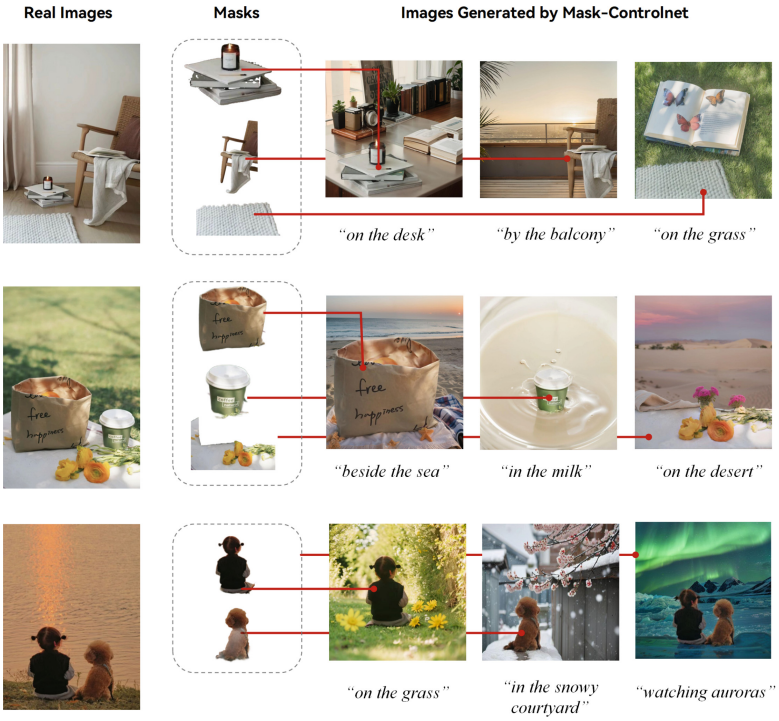


Fig. 8. Images generated using the same reference but different masks.

(2) Virtual try-on. As shown in Fig. 10, our Mask-ControlNet can be adopted for virtual try-on by changing the clothes, hairstyle, and facial accessories in the image while maintaining face fidelity using masks.

Fig. 9. Augmented images synthesized from samples in the COCO dataset.

(3) Image editing. As shown in Fig. 11, our Mask-ControlNet can be used as a powerful interactive editing tool. With our method, fine-grained image editing can be achieved by using the mask prompt to maintain the fidelity of corresponding regions.

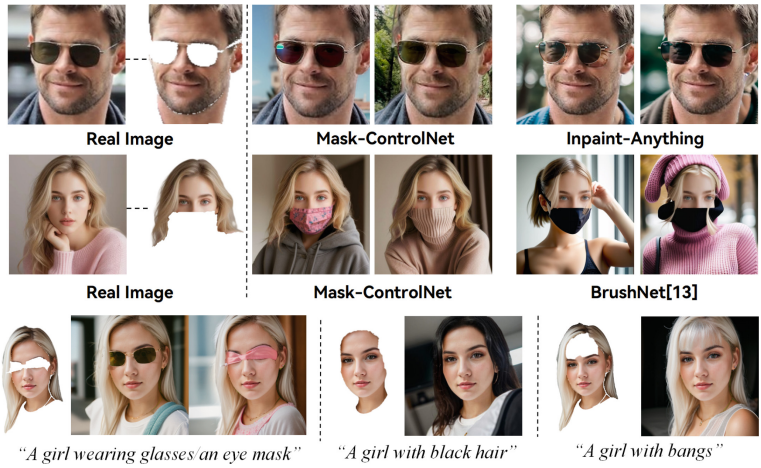


Fig. 10. Virtual try-on results.

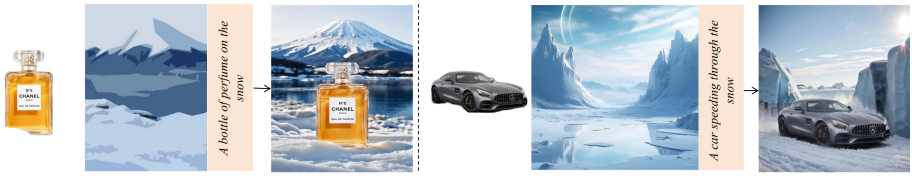


Fig. 11. Image editing results.

5 Conclusion

In this paper, we present a simple yet effective framework to synthesize high-quality images with an additional mask prompt. To better model the relationship between foreground and background, we use SAM to obtain the object mask to provide additional cues. With this addition of conditional information, the network can well capture the foreground-background correlation and generate visually more pleasing results. Extensive experiments demonstrate the effectiveness of our method and the superiority of our generated images.

References

1. Avrahami, O., Fried, O., Lischinski, D.: Blended latent diffusion. ACM Trans. Graph. (TOG) **42**(4), 1–11 (2023)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
3. Brooks, T., Holynski, A., Efros, A.A.: InstructPix2Pix: learning to follow image editing instructions (2023)
4. Cai, S., et al.: DiffDreamer: towards consistent unsupervised single-view scene extrapolation with conditional diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 2139–2150, October 2023
5. Caron, M.,et al.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9650–9660 (2021)
6. Corneanu, C., Gadde, R., Martinez, A.M.: LatentPaint: image inpainting in latent space with diffusion models. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 4334–4343 (2024)
7. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12873–12883 (2021)
8. Gal, R., et al.: An image is worth one word: personalizing text-to-image generation using textual inversion (2022)
9. Goodfellow, I., et al.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
10. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Adv. Neural. Inf. Process. Syst. **33**, 6840–6851 (2020)
11. Hu, E.J., et al.: LoRa: low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)

12. Huang, L., Chen, D., Liu, Y., Shen, Y., Zhao, D., Zhou, J.: Composer: creative and controllable image synthesis with composable conditions. arXiv preprint arXiv:2302.09778 (2023)
13. Ju, X., Liu, X., Wang, X., Bian, Y., Shan, Y., Xu, Q.: BrushNet: a plug-and-play image inpainting model with decomposed dual-branch diffusion (2024). https://arxiv.org/abs/2403.06976
14. Ju, X., Zeng, A., Zhao, C., Wang, J., Zhang, L., Xu, Q.: HumanSD: a native skeleton-guided diffusion model for human image generation. arXiv preprint arXiv:2304.04269 (2023)
15. Kang, M., et al.: Scaling up GANs for text-to-image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10124–10134 (2023)
16. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
17. Kawar, B., et al.: Imagic: text-based real image editing with diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6007–6017 (2023)
18. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
19. Kingma, D.P., Dhariwal, P.: Glow: generative flow with invertible 1x1 convolutions. Adv. Neural Inf. Process. Syst. **31** (2018)
20. Kirillov, A., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)
21. Kumari, N., Zhang, B., Wang, S.Y., Shechtman, E., Zhang, R., Zhu, J.Y.: Ablating concepts in text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 22691–22702 (2023)
22. Kumari, N., Zhang, B., Zhang, R., Shechtman, E., Zhu, J.Y.: Multi-concept customization of text-to-image diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1931–1941 (2023)
23. Li, J., Li, D., Xiong, C., Hoi, S.: BLIP: bootstrapping language-image pre-training for unified vision-language understanding and generation. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 12888–12900. PMLR, 17–23 July 2022. https://proceedings.mlr.press/v162/li22n.html
24. Li, Y., et al.: GLIGEN: open-set grounded text-to-image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22511–22521 (2023)
25. Liu, X., et al.: More control for free! Image synthesis with semantic diffusion guidance. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 289–299 (2023)
26. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Repaint: inpainting using denoising diffusion probabilistic models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11461–11471 (2022)
27. Mou, C., et al.: T2I-adapter: learning adapters to dig out more controllable ability for text-to-image diffusion models. arXiv preprint arXiv:2302.08453 (2023)
28. Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelCNN decoders. Adv. Neural Inf. Process. Syst. **29** (2016)

29. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4195–4205 (2023)
30. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763. PMLR (2021)
31. Reddy, M.D.M., Basha, M.S.M., Hari, M.M.C., Penchalaiah, M.N.: DALL-E: creating images from text. UGC Care Group I J. **8**(14), 71–75 (2021)
32. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022)
33. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: DreamBooth: fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22500–22510 (2023)
34. Seitzer, M.: PyTorch-fid: fid score for pytorch (2020)
35. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning, pp. 2256–2265. PMLR (2015)
36. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
37. Tanchenko, A.: Visual-PSNR measure of image quality. J. Vis. Commun. Image Represent. **25**(5), 874–878 (2014)
38. Van Den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: International Conference on Machine Learning, pp. 1747–1756. PMLR (2016)
39. Voynov, A., Aberman, K., Cohen-Or, D.: Sketch-guided text-to-image diffusion models. In: ACM SIGGRAPH 2023 Conference Proceedings, pp. 1–11 (2023)
40. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
41. Xie, S., Zhang, Z., Lin, Z., Hinz, T., Zhang, K.: SmartBrush: text and shape guided object inpainting with diffusion model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22428–22437 (2023)
42. Yu, T., et al.: Inpaint anything: segment anything meets image inpainting (2023). https://arxiv.org/abs/2304.06790
43. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3836–3847 (2023)
44. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–595 (2018)
45. Zheng, G., Zhou, X., Li, X., Qi, Z., Shan, Y., Li, X.: LayoutDiffusion: controllable diffusion model for layout-to-image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22490–22499 (2023)
46. Zhou, Y., Liu, B., Zhu, Y., Yang, X., Chen, C., Xu, J.: Shifted diffusion for text-to-image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10157–10166 (2023)

Freestyle 3D-Aware Portrait Synthesis Based on Compositional Generative Priors

Tianxiang Ma[1,2], Kang Zhao[3], Jianxin Sun[1,2], Yingya Zhang[3], and Jing Dong[2(✉)]

[1] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[2] CRIPAC; MAIS; Institute of Automation, Chinese Academy of Sciences, Beijing, China
jdong@nlpr.ia.ac.cn
[3] Alibaba Group, Hangzhou, China

Abstract. Efficiently generating a freestyle 3D portrait with high quality and 3D-consistency is a promising yet challenging task. The portrait styles generated by most existing methods are usually restricted by their 3D generators, which are learned in specific facial datasets, such as FFHQ. To get the diverse 3D portraits, one can build a large-scale multi-style database to retrain a 3D-aware generator, or use a off-the-shelf tool to do the style translation. However, the former is time-consuming due to data collection and training process, the latter may destroy the multi-view consistency. To tackle this problem, we propose a novel text-driven 3D-aware portrait synthesis framework that can generate out-of-distribution portrait styles. Specifically, for a given portrait style prompt, we first composite two generative priors, a 3D-aware GAN generator and a text-guided image editor, to quickly construct a few-shot stylized portrait set. Then we map the special style domain of this set to our proposed 3D latent feature generator and obtain a 3D representation containing the given style information. Finally we use a pre-trained 3D renderer to generate view-consistent stylized portraits from the 3D representation. Extensive experimental results show that our method is capable of synthesizing high-quality 3D portraits with specified styles in a few minutes, outperforming the state-of-the-art.

Keywords: 3D Portrait Synthesis · Text-to-3D Generation

1 Introduction

Portrait synthesis [6,13,20,21] is a promising yet challenging research topic for its wide range of application potential, e.g. game character production, Metaverse avatars and digital human. With the rapid development of generative models such as generative adversarial models [11], 2D portrait synthesis has achieved remarkable success. After that, many methods [19–21] are proposed to improve the generation quality to photo-realistic level.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 98–113, 2025.
https://doi.org/10.1007/978-3-031-78172-8_7

Fig. 1. Our freestyle 3D-aware portrait synthesis results. The first row shows man with different styles, the second row shows various characters, and the last two rows show diverse identities with different styles from varied viewpoints. Zoom in for better viewing.

Recently, 3D portrait synthesis has attracted more and more attention, especially with the emergence of Neural Radiance Field (NeRF) [29]. As the representatives among them, 3D-aware GAN methods [6,13,33] combine NeRF with StyleGANs [21] to ensure 3D consistency synthesis. By mapping an image to the 3D GAN latent space, 3D GAN inversion approaches [23,24,45] can generate or edit a specific 3D portrait. However, both of them fail to create a freestyle 3D portrait, e.g., a style defined by user's text prompt, since their generators are usually trained on a dataset that follows a particular style distribution, such as the realism style in FFHQ [20], which raises a question: how to generate a freestyle 3D portrait at a low cost? One may collect a large number of portrait images with different styles to retrain their 3D generative models, but the data preparing and training process are usually time-consuming. Another potential solution is that synthesizing a dataset-based 3D portrait first, then transferring each portrait view to other styles with a off-the-shelf 2D style transfer tool. Unfortunately, the 3D consistency will be difficult to be maintained.

To this end, we propose an efficient framework to achieve freestyle 3D-aware portrait synthesis in this paper. At first, we composite the knowledge of two pre-trained generative priors, EG3D [6] and Instruct-pix2pix (Ip2p) [3] we used in this paper, to construct a few-shot stylized portrait dataset with a given style prompt, avoiding dirty data collection and cleaning. The former generates a multi-view 3D portrait, and the latter performs text-guided style editing in each viewpoint. We empirically find the editing results of Ip2p vary significantly along viewpoints for some given style, resulting in the issue of multi-view misalignment. To alleviate this problem, some optimization strategies are introduced into the inference stage of Ip2p. Secondly, we propose a 3D latent feature generator model, which can map the style information from stylized portrait dataset to 3D latent representation. This generator is first pre-trained with large amount

of generative data from EG3D, so that it can learn 3D latent representation well from multi-view portraits. Our 3D latent feature generator enables fast fine-tuning with few-shot stylized portrait dataset, and generates out-of-distribution stylized 3D representation. Finally, we utilize the pre-trained EG3D neural renderer and super-res generator as our 3D Renderer, and we find that this model does not need to be fine-tuned to generate 3D-consistent portrait with a specified style when we have learned a well-stylized 3D implicit representation. Our high-quality stylized 3D-aware portrait synthesis results are shown in Fig. 1, each style is specified by a text prompt, and each 3D portrait model can be fine-tuned in 3 min. Our main contributions can be summarized as follows:

- We propose a 3D-aware portrait synthesis framework based on compositional generative priors that can be text-driven to generate freestyle 3D avatar.
- We further propose a 3D latent feature generator, which allows for quick fine-tuning to map out-of-distribution style to 3D representation.
- Compared with stylized 3D portrait synthesis baselines, our approach has clear advantages in performance and efficiency.

2 Related Work

2.1 3D-Aware GAN

With the development of neural implicit representation (NIR) represented by neural radiance fields (NeRF) [29], more and more methods [1,5,9,10,12,17,28,31] are focusing on learning 3D scenes and 3D object representation using neural networks. NeRF represents the 3D scene as a series of neural radiance and density fields, and uses volume rendering [18] technique for 3D reconstruction. Similarly, some methods [32,39] learn neural implicit representation using multi-view 2D images without 3D data supervision. However, even multi-view data is usually expensive to construct in some scenes, such as portraits, so many approaches gradually migrate to learn 3D-aware GAN using unstructured data, i.e., single-view portraits, based on the idea of adversarial training. PiGAN [7] proposes a siren-based neural radiance field and uses global latent code to control the generation of shapes and textures. GIRAFFE [30] proposes a two-stage rendering process, which first generates the low-resolution features with a volume renderer, and then learns to upsample the features with a 2D CNN network. Some methods introduce StyleGAN structures into the 3D-aware GAN. StyleNeRF [13] integrates NeRF into a style-based generator to improve rendering efficiency and 3D-consistency of high-resolution image generation. StyleSDF [33] merges a Signed Distance Fields representation with a style-based 2D generator. EG3D [6] proposes a triplane 3D representation method to improve rendering computational efficiency and generation quality. Some approaches have also started to focus on the control and editing of 3D-aware GANs. FENeRF [42] and Sem2nerf [8] introduce semantic segmentation into the generative network, and learn a whole neural radiance field with semantic information. CNeRF [26] proposes a compositional neural radiance field to split the portrait into multiple semantic

regions, and learns semantic synthesis separately with a local neural radiance field, and finally fuses them into a complete 3D representation of the portrait. Next3D [41] learns a face-driven method based on EG3D, which can generate 3D-consistent facial avatars. Along with the development of 3D-aware GAN, 3D GAN Inversion methods [23,24,45] have appeared. They learn to map real images to the latent space of 3DGAN for image inversion and editing. However, such methods face a problem that they cannot jump out of the pre-trained 3DGAN prior and cannot synthesize out-of-distribution portraits. In this paper, we propose a new framework that can synthesize stylized 3D portraits freely, which is not restricted by the 3D generative prior and can generate 3D portraits of specific styles based on text prompts.

2.2 Text-Guided Image Editing

There are numerous image editing methods, and the performance of text-guided image editing methods [2,3,14,15,22,25] has been qualitatively improved thanks to the advancement of pre-trained image generation large models [36–38] based on the Diffusion model. Ip2p [3] is a SOTA text-guided image editing method, which uses two generative priors, GPT-3 [4] and Stable Diffusion [38], to synthesize a large number of paired images and then train a conditional diffusion model on them. This model allows the users to provide a relatively free text instruction to edit a given image, including stylistic transfer. Some Ip2p-based methods [14,43] have also verified its application potential in image stylization tasks. Therefore, Ip2p is well suitable as a text-guided image editing prior for this paper to perform text-guided style transfer on portraits from different viewpoints. However, the model also has problems, such as poor generation with some simple text prompts and generating portraits with large stylistic variations for different views of the same portrait. We introduce some optimization strategies to alleviate these problems in this paper.

3 Methodology

In this section, we detail our freestyle 3D-aware portrait synthesis framework, as shown in Fig. 2. We first briefly introduce two generative priors, EG3D and Ip2p, and composite them to build a few-shot dataset with a given style. To describe styles more freely, we optimize the inference of Ip2p to make it more stable for stylizing portraits from different perspectives (Sect. 3.1). We then use the few-shot dataset to quickly fine-tune our proposed 3D latent feature generator, which is equipped with a pre-trained 3D renderer to generate 3D-consistent stylized portraits (Sect. 3.2).

3.1 Few-Shot Stylized Portrait Dataset Construction

3D-Aware GAN Prior. As a state-of-the-art 3D-aware GAN method, EG3D [6] can be expressed as $G(\theta, \mathbf{w}, \mathbf{v})$, where θ is the model parameters, $\mathbf{w}$ is a

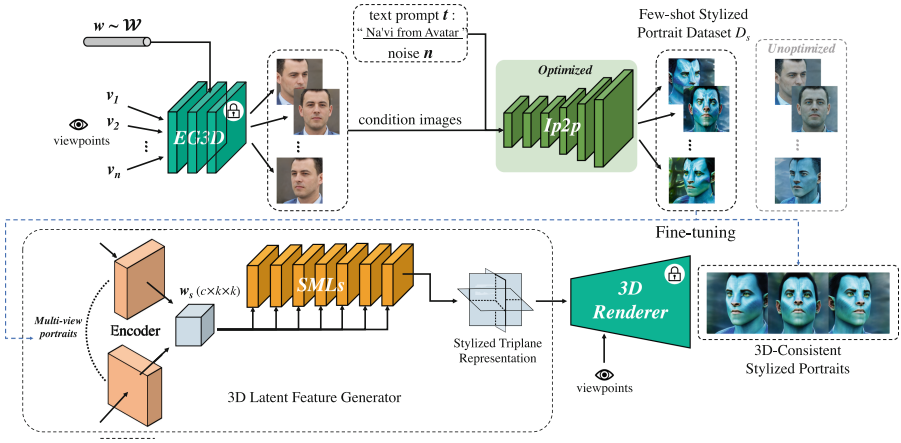


Fig. 2. The framework of our method. We composite two generative priors EG3D and Ip2p, a 3D-aware GAN generator and a text-guided image editor, to quickly build a few-shot stylized portrait dataset $\mathcal{D}_s$, whose style is specified by the text prompt. Ip2p is optimized to produce more stable and consistent text-guided stylization results. With the constructed few-shot dataset, we quickly fine-tune a 3D latent feature generator to map the style information from the dataset into 3D representation, and then utilize a pre-trained 3D renderer to generate the stylized 3D-consistent portraits.

sampling vector in the $\mathcal{W}$ latent space, and $\mathbf{v}$ is the view direction to be rendered. We randomly sample a $\mathbf{w}$ vector, and set $\mathbf{v}$ as follows: assuming pitch and yaw angles of front view portrait are zero, $\mathbf{v}$ is uniformly sampled i times within both pitch and yaw range of $-30°$ to $30°$. We denote the candidate set of $\mathbf{v}$ as (P, Y), which contains i^2 sampling results (i *pitch* $\times$ i *yaw*). Then G can output i^2 portrait images along each $\mathbf{v}$. Note that these portraits keep the same identity since $\mathbf{w}$ is only sampled once.

Text-Guided Image Editing Prior. Next, we employ Instruct-pix2pix (Ip2p) [3] to perform the style editing on the portraits generated above. Ip2p implements conditional image editing based on Stable Diffusion [38], which can be denoted as $T(\phi, I, \mathbf{n}, \mathbf{t})$, where ϕ is the model parameters, I is the input portrait, $\mathbf{n}$ is the Gaussian noise used in the denoising process, and $\mathbf{t}$ is the text prompt that guides the editing direction. Considering different input noise will generate different results, we fix $\mathbf{n}$ to keep the identity unchanged. Then we let I be the multi-view portraits that are produced by EG3D, and use T to generate stylized portraits.

Optimizing Instruct-pix2pix Inference. As mentioned above, some $\mathbf{t}$ will cause Ip2p to generate unsatisfactory stylized portraits. For example, the style "Na'vi from Avatar" will make the portrait style vary greatly from one viewpoint to another. More samples are listed in Sect. 4.4. Therefore, we want to optimize Ip2p to make it generate stable results for different $\mathbf{t}$. Inspired by SDEdit [27],

we replace the original Gaussian noise $\mathbf{n}$ with a new noise $\mathbf{n}^*$ during the inference stage of Ip2p:

$$\mathbf{n}^* = Add(\mathcal{E}(I), \mathbf{n}, \tau), \tag{1}$$

where $\mathcal{E}(I)$ denotes the latent features obtained from the Stable Diffusion encoder of I, τ is the degree of noise addition, and Add represents the standard DDPM [16] noise addition operation. In addition, we design an enhanced prompt to further improve the quality of synthesized portraits: $\mathbf{t}^* = \{\mathbf{t}, \mathbf{t_d}, \mathbf{t_n}\}$, where $\mathbf{t_d}$ and $\mathbf{t_n}$ mean decorative and negative prompts, respectively. Consequently, our stylized portrait generation can be rewritten as:

$$I_s = T(\phi, G(\theta, \mathbf{w}, \mathbf{v}), \mathbf{n}^*, \mathbf{t}^*), \tag{2}$$

where I_s is one stylized portrait. We construct a few-shot stylized portrait dataset $\mathcal{D}_s$ using different view $\mathbf{v}$ from (P, Y), so $\mathcal{D}_s$ contains i^2 stylized portraits. The construction pipeline is summarized in Algorithm 1.

3.2 3D Latent Feature Generator

Considering the ability in generating high quality and 3D consistency image, we also utilize EG3D baseline to synthesize stylized 3D portrait. In the experiment, we find that if we have learned a well-stylized 3D implicit representation, the EG3D neural renderer and super-res generator do not need to be fine-tuned to generate 3D-consistent portrait with an out-of-distribution style. Therefore, we utilize the pre-trained EG3D neural renderer and super-res generator as our 3D renderer, and just need to learn a 3D latent feature generator to map the style information from $\mathcal{D}_s$ to the 3D implicit representation.

There are two other solutions that need to be discussed: 1) Directly fine-tuning the whole EG3D on $\mathcal{D}_s$, which is very inefficient compared to our method and generates somewhat blurred images. 2) Inverting the images from $\mathcal{D}_s$ to the $\mathcal{W}$ latent space of EG3D with GAN inversion methods, which will limit the portrait style within the scope of training set that EG3D is pre-trained, since the triplane feature generator is not changed.

We design the 3D latent feature generator as shown in the Fig. 2. An Encoder is used to learn the modulation latent feature $\mathbf{w}_s$ from different viewpoints, we expand the latent code $\mathbf{w}_s$ along spatial dimension (from c to $c \times k \times k$) to enrich it with style and structural information. Then we propose the *SMLs* consisting of multiple Stylized modulation layers, which can learn to generate stylized triplane implicit representation. For a particular modulation layer in *SMLs*, we have the following formula:

$$F^i_{c,h,w} = \gamma^i_{h,w}(\mathbf{w}_s) \times \frac{F^i_{c,h,w} - \mu^i_{h,w}}{\sigma^i_{h,w}} + \beta^i_{h,w}(\mathbf{w}_s), \tag{3}$$

where $\mu^i_{h,w}$ and $\sigma^i_{h,w}$ represent the calculated mean and standard deviation across channel dimension, respectively. $\gamma^i_{h,w}$ and $\beta^i_{h,w}$ are learnable weight networks. The last layer of *SMLs* outputs the stylized triplane representation.

Algorithm 1. Few-shot dataset construction

Input: $\mathbf{w} \sim \mathcal{W}$, $\mathbf{n} \sim \mathcal{N}(0,1)$, $\mathbf{t}$, $\mathcal{D}_s = \emptyset$
for v in (P, Y) **do**
  $I = G(\theta, \mathbf{w}, \mathbf{v})$
  $\mathbf{n}^* = Add(\mathcal{E}(I), \mathbf{n}, \tau)$
  $I_s = T(\phi, I, \mathbf{n}^*, \mathbf{t}^*)$
  $\mathcal{D}_s = \mathcal{D}_s \cup I_s$
end for

Algorithm 2. The 3D latent feature generator fine-tuning

Input: $\mathcal{D}_s$, Pre-trained 3D latent feature generator
Init 3D generator net using the pre-trained model
repeat
  Randomly select I_v^1, I_v^2 from $\mathcal{D}_s$
  Fine-tuning 3D latent feature generator using loss:
    $\mathcal{L}_{\text{total}} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{dr}}\mathcal{L}_{\text{dr}}$
until end of iterations

Although the 3D latent feature generator can be trained on $\mathcal{D}_s$, it still suffers from two challenges: First, $\mathcal{D}_s$ is a few-shot dataset, it has no more than 100 images for a style in practice. Second, the portraits in $\mathcal{D}_s$ have more or less differences in details, resulting in 3D inconsistency. We solve this problem by pre-training the 3D latent feature generator on the large number of portraits generated by EG3D prior, which can help to build accurate 3D implicit representation from multi-view portraits. In particular, in each iteration of the pre-training, we randomly sample to generate an arbitrary viewpoint portrait $I_v \in \mathbb{R}^{512\times512\times3}$ from EG3D, and record its triplane representation as $\mathbf{p} \in \mathbb{R}^{256\times256\times96}$, Then the pre-training process is constrained by the following loss function:

$$\mathcal{L}_{\text{pre}} = \mathbb{E}_{I_v,\mathbf{p}}[\|H(I_v) - \mathbf{p}\|_1], \tag{4}$$

where H represents our 3D latent feature generator.

Training. After being pre-trained, the 3D latent feature generator can be quickly fine-tuned on the few-shot stylized portrait dataset $\mathcal{D}_s$. We use a multi-view cross-construction strategy for learning a more robust 3D implicit representation. For two arbitrary viewpoint portraits I_v^1 and I_v^2 in $\mathcal{D}_s$, we use the following reconstruction loss:

$$\begin{aligned}\mathcal{L}_{\text{rec}} = \mathbb{E}_{I_v^1, I_v^2 \in \mathcal{D}_s}[\big\| G^*(I_v^1, \mathbf{v_2}) - I_v^2 \big\|_1 \\ + lpips(G^*(I_v^1, \mathbf{v_2}), I_v^2)],\end{aligned} \tag{5}$$

where $\|\cdot\|_1$ is the L_1 reconstruction loss, G^* represents the whole model with 3D latent feature generator and 3D renderer, and the parameters are input image and the rendering viewpoint, respectively. $lpips(\cdot,\cdot)$ is the Learned Perceptual Image Patch Similarity [46] loss, which calculates the distance of the latent features extracted from the VGG network. In addition to the reconstruction loss, we add the density regularization, which encourages smoothness of the density field rendered by triplane and prevents sharp or hollow portrait shapes during fine-tuning. The density regularization loss is shown as follows,

$$\mathcal{L}_{\text{dr}} = \mathbb{E}_{\mathbf{x},\delta}[\|\sigma(\mathbf{x}) - \sigma(\mathbf{x} + \delta \cdot \mathbf{x})\|_1], \tag{6}$$

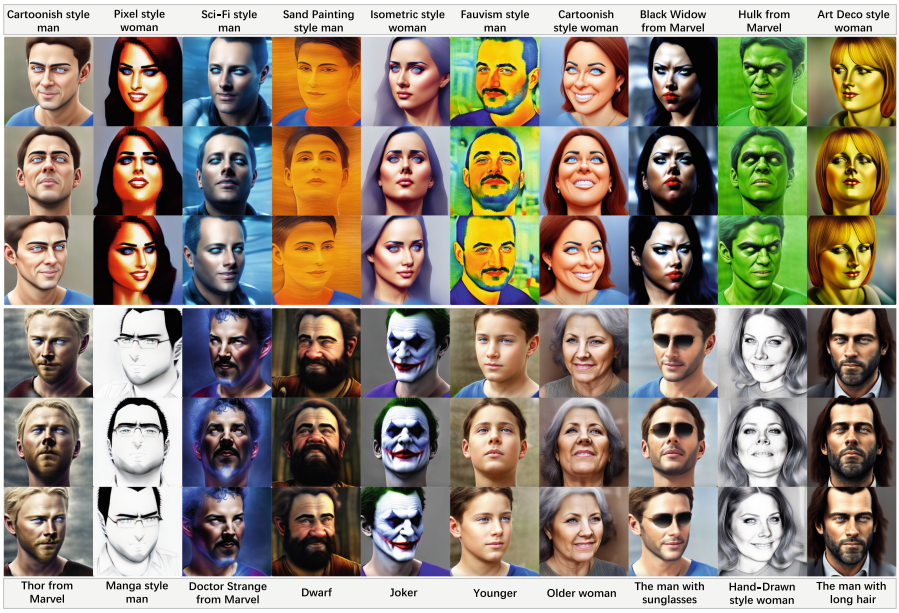


Fig. 3. Multi-style and Multi-identity 3D portrait synthesis results. Zoom in for better viewing.

where $\mathbf{x}$ are the random sampling points in the volume rendering, δ is a small Gaussian noise, and $\sigma(\mathbf{x})$ denotes the density rendering process. Thus the final loss function is:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{dr}}\mathcal{L}_{\text{dr}}, \tag{7}$$

where λ_{rec} and λ_{dr} are loss weights. The fine-tuning process is listed in Algorithm 2.

4 Experiments

4.1 Implementation Detail

Our method is implemented in PyTorch using an NVIDIA A100. We use Adam optimizer with learning rate of 0.002 and $\beta_1 = 0, \beta_2 = 0.99$. The number of samples i in few-shot dataset construction is 10. Other parameters, such as camera focal length, use the EG3D default settings. In Ip2p inference, we set the number of time step $\mathbf{s}$ in DDIM [40] to 20. The degree of noise addition τ is 0.9, and the image guide and text guide weight parameters of Ip2p are set to 1.5 and 20.0. The decorative prompt $\mathbf{t_d}$ is "realistic, detail, 8k, photorealistic", then the positive prompt input for Ip2p is "turn the head into $\mathbf{t}$, $\mathbf{t_d}$". The negative prompt $\mathbf{t_n}$ is "unclear facial features, non-face objects, ugly, bad". Note that we fix $\mathbf{t_d}$ and $\mathbf{t_n}$, and only change $\mathbf{t}$ in all following experiments. Fine-tuning $\mathbf{t_d}$

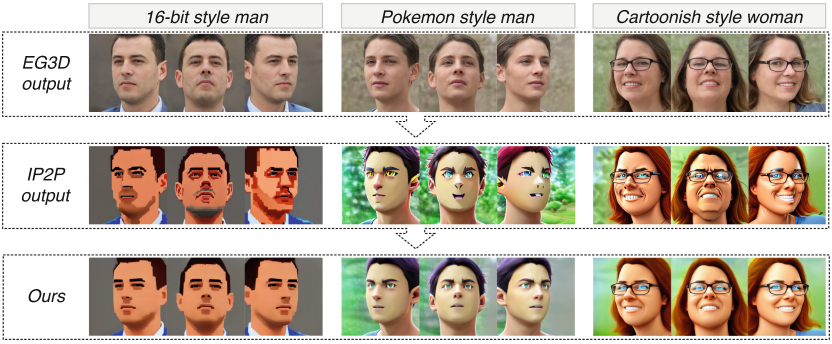


Fig. 4. The outputs of different stages in our method.

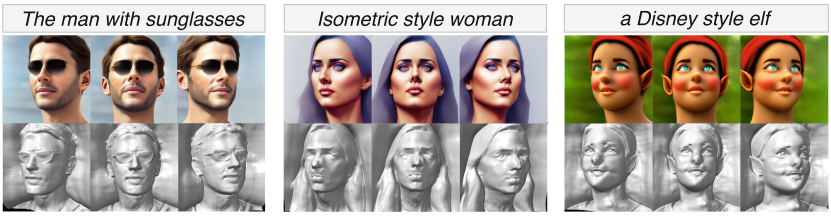


Fig. 5. Visualization of 3D geometry from different stylized generation results.

or $\mathbf{t_n}$ will polish the generated results, but it's not the scope of this paper. For each text prompt $\mathbf{t}$ we randomly sample a $\mathbf{w}$ and construct the few-shot dataset according to Algorithm 1. In 3D latent feature generator, The spatial dimension k of $\mathbf{w}_s$ is 32. *SMLs* consists of 7 modulation layers. The 3D generator is pre-trained on EG3D randomly sampled data for 100k iterations. When fine-tuning the model on the few-shot dataset $\mathcal{D}_s$, the loss weights are set as $\lambda_{\text{rec}} = 10.0$ and $\lambda_{\text{dr}} = 0.2$.

4.2 Freestyle 3D Portrait Synthesis

In this section, we show the 3D portrait synthesis results of our approach. As shown in Fig. 3, our method can generate diverse style 3D portrait, and the synthesized portraits are high quality and 3D consistent, proving our ability of freestyle generation. Then, we display the outputs of different stage (i.e., EG3D, IP2P and final output) in Fig. 4, it can be seen EG3D outputs the original 3D portrait, IP2P will add the style into the portrait images (the facial details and consistency can not be guaranteed), our method will generate the final high quality and 3D-consistency results. In addition, we generate the 3D geometry results of different style in Fig. 5, which can accurately reflect the given style. Last but not least, Fig. 6 shows the smooth style mixing results of our method. We interpolate the encoded features of two different style in the $\mathbf{w}_s$ latent space,

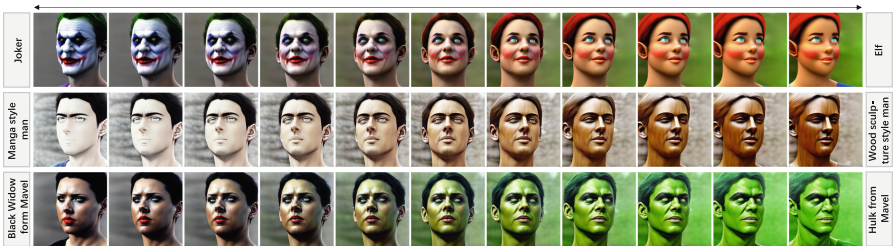


Fig. 6. The 3D portrait results synthesized by mixing different styles in the $\mathbf{w}_s$ latent space.

Table 1. User Study, quantitative evaluation and running time of different methods.

Method	TIS	IQ	3DC	CLIP Score	Average Time
DreamFusion [34]	2.82	2.24	3.18	0.304	∼40 min
3DGAN-Inv [23]	2.04	1.90	2.59	0.229	∼4 min
HFGI3D [44]	3.40	3.54	3.67	0.310	∼10 min
Ours	**4.05**	**4.27**	**4.34**	**0.332**	∼3 min

and use the new triplane implicit representation to synthesize 3D portraits. The results reveal the controllability and potential style editing ability of our model.

4.3 Comparison Experiments

Baselines. We divide the baselines into two categories. 1) Text-to-3D. DreamFusion[1] [34] is a representative method that generates 3D images based on the text prompts. 2) Image style transfer + 3D GAN Inversion. 3DGAN-Inv [23] and HFGI3D [44] are two SOTA methods of 3D GAN inversion. We use them with Instruct-pix2pix as the baselines.

The results of our method compared with baselines are shown in Fig. 7. DreamFusion, the representative of the Text-to-3D method, is able to optimize the model to synthesize 3D portraits based on text prompts, such as "A Disney style Elf", but the generated results have low quality, while for more specific portrait styles, such as "sand painting style", no results can be synthesized. What's more, the portraits with large stylistic variations cannot be inverted well using the 3DGAN-Inv, because the 3D portraits synthesized by this method cannot escape from the domain of the pre-trained 3D-aware GAN model. The results of HFGI3D are better, but the 3D shape is destroyed after inversion, as shown in the Elf's face. At the same time, for some more difficult samples, this method cannot produce effective results, such as the last two lines. In contrast, our method is able to synthesize high quality 3D portraits that satisfy both 3D consistency and stylization.

[1] https://github.com/ashawkey/stable-dreamfusion.

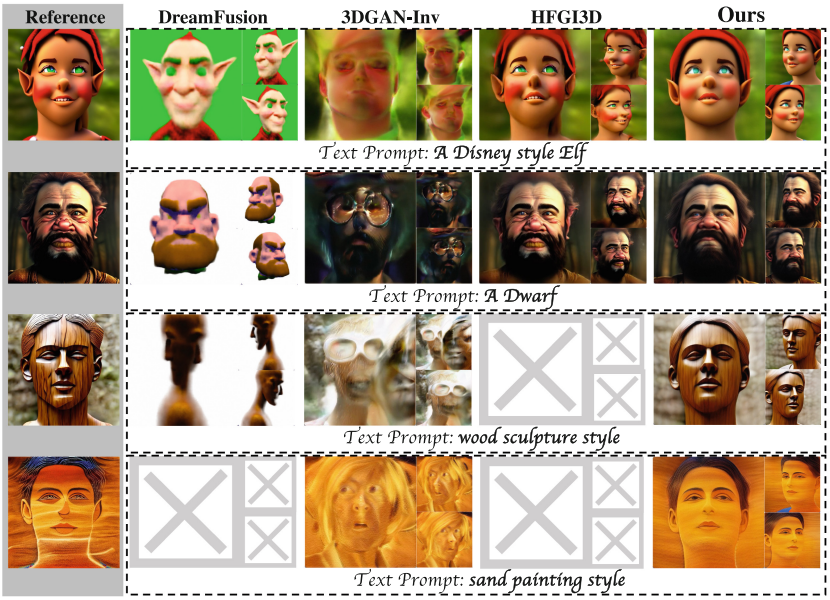


Fig. 7. Qualitative comparison results between our method and baselines. Reference image represents the result after Ip2p style transformation, which is used as the input of the 3D GAN Inversion methods. The part with no result means that the method is unable to generate a image.

For quantitative evaluation, we conduct a user study. We invite 30 volunteers to evaluate each method from three perspectives, namely the Text-image Similarity (TIS), Image Quality (IQ), and 3D Consistency of different views (3DC). Each item is scored on a 5-point scale, and the average is calculated as the final result. As shown in Table 1, our approach achieves the best scores under each dimension. We also calculate the CLIP Score for each method shown in Table 1, which uses the CLIP [35] model to extract features and calculate cosine similarity between input text and generated image. Our method also achieves optimal result. In addition, our method also has a significant advantage in running time, as shown in Table 1. Our method is able to fine-tune model on the given text prompt in about 3 min, while other approaches require more time. Although 3DGAN-Inv costs comparable running time to ours, its portrait generation quality is poor and fails to generate freestyle portraits.

4.4 Ablation Study

We conduct the ablation studies of our method from three aspects, which are shown in Fig. 8. First, when we do not optimize the Ip2p inference, some character prompts generate poor results, such as the example in (a) of Fig. 8). Second, we perform the ablation of the 3D latent feature generator in (b) of Fig. 8. 1)

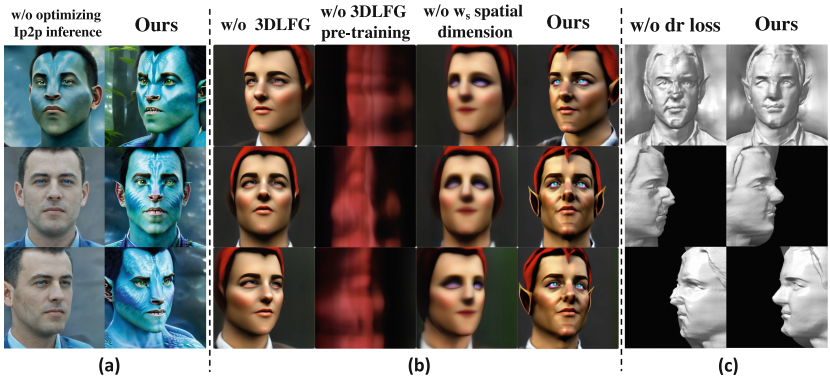


Fig. 8. The ablation studies of our approach. (a) Ip2p generated results w/ or w/o inference optimization. (b) Different ablation experiments on the 3D Latent Feature Generator (3DLFG). (c) 3D shape outputs w/ or w/o dr loss.

Table 2. Quantitative ablation results on 3D latent feature generator.

Method	CLIP Score
w/o 3D generator	0.251
w/o pre-training	0.113
w/o $\mathbf{w}_s$ spatial dimension	0.282
Ours	**0.332**

When we remove the 3D latent feature generator and directly fine-tuning the whole EG3D, we can achieve some stylization effect. However, the generation quality is somewhat poor, and this fine-tuning is inefficient, taking dozens of minutes. 2) When using our 3D latent feature generator without pre-training, training from scratch on the few-shot dataset will not be able to learn the correct 3D implicit representation. Because it is difficult to establish the mapping from the image to the triplane representation only using 3D inconsistent data. 3) When the $\mathbf{w}_s$ code of our 3D latent feature generator is a vector without spatial dimension, the learned 3D portrait is blurred and lacks details. Furthermore, we calculate the CLIP Score of different ablation models in Table 2, and our method achieves the best result. In addition, we ablate the density regularization (dr) loss used in model training, shown in (c) of Fig. 8. When the density regularization is not used, the synthesized portrait shape will have rough surfaces.

5 Limitations

Our approach is based on two powerful pre-trained generative priors, which are the basis of our method's ability to synthesize high-quality stylized 3D portraits. At the same time, our method is limited by both priors, especially Ip2p,

which is unable to achieve perfect 3D-consistent portrait stylization in different viewpoints, so the final synthesized 3D portrait of our method differs slightly from the style generated by Ip2p. Meanwhile, some stylistic changes that differ significantly from the human portrait shape, such as "Iron Man" and "Stormtrooper"' in Fig. 9, result in poorer quality 3D portraits compared to human portraits. Because when Ip2p generates this type of style, the stylization effect will be more different under different views.

Fig. 9. Some bad cases.

6 Conclusions

This paper proposes a novel freestyle 3D-aware portrait synthesis framework based on compositional generative priors. We combine two pre-trained generative priors, EG3D and Ip2p, to quickly construct the out-of-distribution stylized multi-view portraits. We optimize Ip2p inference in order to stylize portraits at different views more freely and stable. We propose a 3D latent feature generator, which learns to map the style information from few-shot stylized portrait dataset to the 3D implicit representation. Equipping the generator with a pre-trained 3D renderer, we can generate 3D-consistent stylized portraits. A large number of high-quality 3D portrait synthesis results and comparison experiments with baselines show the superiority of our method.

Acknowledgments. This work is supported by the National Key Research and Development Program of China under Grant No. 2021YFC3320103.

References

1. Atzmon, M., Lipman, Y.: SAL: Sign agnostic learning of shapes from raw data. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2565–2574 (2020)
2. Avrahami, O., Lischinski, D., Fried, O.: Blended diffusion for text-driven editing of natural images. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 18208–18218 (2022)
3. Brooks, T., Holynski, A., Efros, A.A.: InstructPix2Pix: learning to follow image editing instructions. arXiv (2022)
4. Brown, T., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, pp. 1877–1901 (2020)
5. Chabra, R., et al.: Deep local shapes: learning local SDF priors for detailed 3D reconstruction. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12374, pp. 608–625. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58526-6_36
6. Chan, E.R., et al.: Efficient geometry-aware 3D generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 16123–16133 (2022)

7. Chan, E.R., Monteiro, M., Kellnhofer, P., Wu, J., Wetzstein, G.: pi-GAN: periodic implicit generative adversarial networks for 3D-aware image synthesis. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 5799–5809 (2021)
8. Chen, Y., Wu, Q., Zheng, C., Cham, T.J., Cai, J.: Sem2NeRF: converting single-view semantic masks to neural radiance fields. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13674, pp. 8518–8528. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19781-9_42
9. Chibane, J., Alldieck, T., Pons-Moll, G.: Implicit functions in feature space for 3D shape reconstruction and completion. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 6970–6981 (2020)
10. Chibane, J., Pons-Moll, G., et al.: Neural unsigned distance fields for implicit function learning. In: Advances in Neural Information Processing Systems, pp. 21638–21652 (2020)
11. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems (2014)
12. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. In: International Conference on Machine Learning, pp. 3789–3799 (2020)
13. Gu, J., Liu, L., Wang, P., Theobalt, C.: StyleNeRF: a style-based 3d-aware generator for high-resolution image synthesis. In: International Conference on Learning Representations (2022)
14. Haque, A., Tancik, M., Efros, A.A., Holynski, A., Kanazawa, A.: Instruct-NeRF2NeRF: editing 3D scenes with instructions. arXiv (2023)
15. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D.: Prompt-to-prompt image editing with cross attention control. arXiv (2022)
16. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, pp. 6840–6851 (2020)
17. Jiang, C., et al.: Local implicit grid representations for 3D scenes. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 6001–6010 (2020)
18. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. In: ACM SIGGRAPH Computer Graphics, pp. 165–174 (1984)
19. Karras, T., et al.: Alias-free generative adversarial networks. In: Advances in Neural Information Processing Systems, pp. 852–863 (2021)
20. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 4401–4410 (2019)
21. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of styleGAN. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 8110–8119 (2020)
22. Kawar, B., et al.: Imagic: text-based real image editing with diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 6007–6017 (2023)
23. Ko, J., Cho, K., Choi, D., Ryoo, K., Kim, S.: 3D GAN inversion with pose optimization. In: IEEE Winter Conference on Applications of Computer Vision, pp. 2967–2976 (2023)
24. Lin, C.Z., Lindell, D.B., Chan, E.R., Wetzstein, G.: 3D GAN inversion for controllable portrait image animation. arXiv (2022)
25. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: zero-shot one image to 3D object. arXiv (2023)

26. Ma, T., Li, B., He, Q., Dong, J., Tan, T.: Semantic 3D-aware portrait synthesis and manipulation based on compositional neural radiance field. In: The Association for the Advancement of Artificial Intelligence (2023)
27. Meng, C., et al.: SDEdit: guided image synthesis and editing with stochastic differential equations. In: International Conference on Learning Representations (2021)
28. Michalkiewicz, M., Pontes, J.K., Jack, D., Baktashmotlagh, M., Eriksson, A.: Implicit surface representations as layers in neural networks. In: International Conference on Computer Vision, pp. 4743–4752 (2019)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 405–421. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_24
30. Niemeyer, M., Geiger, A.: Giraffe: representing scenes as compositional generative neural feature fields. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 11453–11464 (2021)
31. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy Flow: 4D reconstruction by learning particle dynamics. In: International Conference on Computer Vision, pp. 5379–5389 (2019)
32. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: learning implicit 3d representations without 3D supervision. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 3504–3515 (2020)
33. Or-El, R., Luo, X., Shan, M., Shechtman, E., Park, J.J., Kemelmacher-Shlizerman, I.: StyleSDF: high-resolution 3d-consistent image and geometry generation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 13503–13513 (2022)
34. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: DreamFusion: Text-to-3D using 2D diffusion. arXiv (2022)
35. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763 (2021)
36. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv (2022)
37. Ramesh, A., et al.: Zero-shot text-to-image generation. In: International Conference on Machine Learning, pp. 8821–8831 (2021)
38. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022)
39. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3D-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
40. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021)
41. Sun, J., et al.: Next3D: generative neural texture rasterization for 3D-aware head avatars. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 20991–21002 (2023)
42. Sun, J., et al.: FENeRF: Face editing in neural radiance fields. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 7672–7682 (2022)
43. Sun, Y., He, R., Tan, W., Yan, B.: Instruct-NeuralTalker: editing audio-driven talking radiance fields with instructions. arXiv preprint arXiv:2306.10813 (2023)

44. Xie, J., Hao, O., Jingtan, P., Chenyang, L., Qifeng, C.: High-fidelity 3D GAN inversion by pseudo-multi-view optimization. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 321–331 (2023)
45. Yin, F., et al.: 3D GAN inversion with facial symmetry prior. arXiv (2022)
46. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–595 (2018)

FUGAN: A GAN Based Facial Reconstructor for Accurate Unveiling of Hidden Faces

Mrinmoy Sadhukhan[1(✉)], Indrajit Bhattacharya[2], and Paramartha Dutta[1]

[1] Department of Computer and System Sciences, Siksha Bhavana, Visva Bharati, Bolpur, India
mrinmoy.sadhukhan1996@gmail.com, paramartha.dutta@visva-bharati.ac.in

[2] Department of Computer Applications, Kalyani Government Engineering College, Kalyani, India

Abstract. Image-to-image translation has become a prominent trend in the field of computer vision. This innovative technique is widely employed for generating concealed facial features from given noise. It proves particularly useful in reconstructing obscured facial regions covered by masks. The underlying technology at the core of this method is the Generative Adversarial Network (GAN). The primary mechanism of GAN-based neural networks relies on an encoder-decoder structure combined with a discriminator network. The discriminator plays a crucial role in training the network to distinguish between authentic and generated images. FUGAN, a novel approach in this domain, incorporates a self-attention block and has demonstrated superior performance to other models. With an impressive SSIM and BRISQUE score of 0.928 and 27.990, respectively, FUGAN is at the forefront of achieving state-of-the-art results. The architecture of the FUGAN neural network is built upon the UNET framework, with modifications to enhance its effectiveness. These modifications involve increasing the network depth and incorporating batch instance normalization as a regularizer to achieve the desired outcomes. The resulting model attains a smaller size and exhibits high accuracy in image regeneration. This streamlined and efficient FUGAN model is well-suited for deploying low-resource Internet of Things (IoT) devices. Its capability to regenerate facial features from masked regions makes it instrumental in surveillance systems, detecting individuals wearing masks and reconstructing their facial details. Github link (FUGAN: https://github.com/mrinmoy-sadhukhan/FUGAN-model) of our work is provided.

Keywords: Facial Reconstructor · GAN · Self Attention · CNN · encoder-decoder · discriminator · Masked Faces

Visva Bharati University.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 114–129, 2025.
https://doi.org/10.1007/978-3-031-78172-8_8

1 Introduction

In the field of computer science, experts have made significant contributions to enhancing security systems for detecting intruders or monitoring suspicious activities, particularly in scenarios where external oversight is limited. They have introduced a variety of tools and techniques, ranging from object detection to segmentation and the sophisticated method of regeneration in computer vision, which enables the generation of images from a single reference image with added noise. While face masks serve as a defense against viruses, some individuals misuse them for criminal activities, concealing their identities in places like malls or hotels to evade surveillance cameras. Researchers have taken on this challenge by employing image inpainting methods to reveal the facial areas obscured by masks. Techniques such as pix2pix, cycle generative adversarial networks (CGAN), image-to-image translation, conditional GAN (C-GAN), encoder-decoder architectures, unpaired image-to-image translation, autoencoders and UNET architectures have been used for this purpose. Despite the substantial research efforts in face identification, the issue of occluded face images, including those obscured by masks, remains unresolved due to factors such as the lack of comprehensive masked face datasets, the intricate nature and variability of masks, and the variability in facial features. Therefore, the recognition and authentication of individuals wearing masks continue to be a longstanding research area focusing on visual improvements of facial images and necessitating more efficient methods for realtime face recognition. In this study, we aim to address the challenge of removing face masks from facial images using Generative Adversarial Networks (GANs). This problem can be framed as image-to-image translation, involving the conversion of images from one domain to another. The contribution of this work is to propose a GAN model that automatically removes mask objects from the face and reconstructs the affected region with delicate details irrespective of color and lighting conditions. Moreover, it can be deployed in IOT devices for real-time testing and performance measurement in a natural environment.

The rest of this paper is organized as follows. In sect 2, the motivation of the work is given. Section 3 presents related studies. Novelty of the work is given in sect 4. The contribution of our work is presented in sect 5. The proposed model with methodology is detailed in sect 6. Section 7 describes the experimental setup and results. Future work and conclusion are described in Sect 8 and sect 9, respectively.

2 Motivation

Intruder detection has become increasingly crucial in malls, offices, hospitals, and other public and private places following and after COVID-19, as some individuals use face masks to conceal their identities and engage in malicious activities. Ensuring the security of individuals in public spaces like roads, tourist attractions, and religious places has also become crucial, as some people hide their

faces with masks or cloths to harm them, making identification challenging. This situation underscores the need for a face-unmasking system that can segment masks from faces, regenerate the occluded parts guided by this segmented binary mask, and subsequently use these images to identify culprits. In response to this need, we have proposed a face-unmasking system called "FUGAN," which can accurately regenerate realistic facial parts from still images.

3 Literature Survey

The key concept of occlusion removal from faces relies on two things: 1) Detection of occlusion (face mask) from faces with an accurate segmentation map and 2) Generation of facial parts from a segmented map starting from noise. The regeneration of facial part can be possible using different supervised and unsupervised methods in field of Machine learning and deep learning. Below a detailed analysis of different SOTA model is given.

Conventional methods such as interpolation, patch-based approaches, and diffusion-based techniques have historically been extensively employed for automated image inpainting. In the interpolation-based technique, nearest neighbor interpolation, bilinear interpolation, and bicubic interpolation are used to estimate missing and corrupted pixels. Alsalamah et al. [1] proposed a radial basis function (RBF) based interpolation technique for image inpainting, demonstrating high accuracy in restoring damaged images. Patch-based methods [16] involve segmenting the image into small patches and using similar patches from known areas to deduce the missing parts. Lin Liang et al. [10] introduced a patch-sampling-based texture synthesis algorithm that generates high-quality textures from an input sample. Bertalmio et al. [3] developed an algorithm inspired by professional restorers that can simultaneously fill multiple regions with different structures without constraints on the inpainting areas. Biradar et al. [4] applied the median filter as a nonlinear filter for inpainting, diffusing the median pixel from the exterior to the interior. They used the PSNR metric to compare their method with other image inpainting techniques.

In the realm of deep learning, there exist cutting-edge models proficient in effectively reconstructing obscured facial features through GAN-based image inpainting techniques, facilitated by precise occlusion segmentation maps generated from segmentation models. Delving deeper into this domain reveals a plethora of GAN models fashioned on UNET architecture, aimed at attaining heightened accuracy in producing lifelike images. Several notable models are discussed herein: NIZAM UD DIN et al. [17] introduced a pioneering GAN model by adapting the UNET architecture, incorporating Leaky relu activation functions and instance normalization layers after each convolutional block. This model, trained with two discriminator networks and a custom loss function, demonstrates proficiency in generating authentic images and is further applied for face mask segmentation, yielding an SSIM score of 0.864 and a commendable PSNR value of 26.19 dB. Divyam Gupta et al. [8] present an innovative approach, UNET++, which modifies the UNET architecture by integrating dense

blocks between different encoder and decoder blocks, within the CoGAN-based Pix2Pix GAN framework. This adaptation achieves an SSIM value of 0.85752 and a PSNR of 24.998 dB. Farnaz Farahanipad et al. [5] employ the CycleGAN model, based on ResNET blocks, trained with a 70×70 patch GAN-based discriminator to compute cycle loss for enhanced training refinement, resulting in an SSIM value of 0.89. Yefan Jiang et al. [7] propose a novel mask removal inpainting network, employing face attributes such as nose, chubby, and beard in a two-staged GAN model. They utilize PRNet for face pose estimation and alignment to generate binary masks, achieving accuracy metrics of 0.9180 for SSIM and 3.7065 for FID. Akhil Kumar et al. [9] introduce the SAM C-GAN method for face reconstruction, employing a conditional generative adversarial network with a spatial attention module to aid in discerning between real and synthetic images of masked faces. Utilizing binary cross-entropy loss for the GAN method and Poisson loss and mean absolute error for the discriminator module, they achieve an SSIM of 0.91231 and a PSNR of 30.9879. Hitoshi Yoshihashi et al. [19] utilize the pix2pix model as a foundation, enhancing it with pixel error and feature point comparison alongside a custom loss function for training on face-masked image datasets, achieving an SSIM value of 0.926 and a PSNR of 28.789. K. Nazeri et al. [14] proposed EdgeConnect method to regenerate the missing area of an image. It is based on two steps: one is an edge generator, and the other is an image completion network. Edge generator predicts the missing edges of an image, then completion networks fill the missing regions guided by regenerated edges. J. Yu et al. [20] proposed a generative image inpainting method with a novel contextual attention mechanism, which effectively borrows and utilizes contextual information from the known regions of an image to infer the missing parts. This method leverages the power of both deep learning and traditional techniques to improve the quality and coherence of the inpainted images.

4 Novelty

In the previous section, it is clear that substantial advancements have been made in the face unmasking method from 2020 to the present, achieving a 90% similarity between original and fake images. Researchers have primarily employed a modified UNET architecture, along with a custom discriminator and complex loss function, to improve the accuracy of GAN models. Synthetic datasets have been used for training as well as testing. However, current literature lacks a comprehensive approach to address the challenge of regenerating facial images with different mask types during the training and testing phases and varying face tilting positions. Furthermore, the literature review reveals that various proposed SOTA models have not achieved high SSIM and PSNR values. This suggests that the images lack realism regarding natural color compression and structural similarity that can not deceive face recognition systems. To fill this gap, a novel GAN based neural network called “FUGAN” has been developed based on a UNET architecture. The modified UNET architecture with one discriminator

and simple loss function can achieve nearly 93% SSIM value in training and testing. The deployment of the model in an IOT device (Jetson Xavier NX) with a minimum number of resources for testing is another novel contribution of the proposed work and can be readily implementable in any real-world situation.

5 Contribution

The FUGAN model enhances the UNET architecture, created by researchers, for segmentation purposes. We customized this architecture to meet our requirements. Firstly, a modified UNET model is used as a segmentation network to extract face masks from faces. Then, it is utilized as a GAN network to regenerate facial parts, guided by the segmentation map. Later, the models are deployed in an IOT device for testing. The main contributions of this work include:

- Development of a modified UNET architecture, FUGAN, with a discriminator to regenerate the facial region.
- Integration of attention blocks within the deeper layers of the model to handle various scenarios and effectively regenerate faces even for unknown types of face masks.
- Creation of a custom loss function tailored to train the face generator efficiently.
- Utilization of the SegUnet model to generate binary face masks for the facial region.
- Develop an improved UNET model to learn facial features in a few iterations. So that it can be further fine-tuned in an unknown face mask dataset.
- Deployment feasibility on edge devices due to the model's compact size, enabling for testing the facial regeneration with minimal resources.

6 Methodology

This section outlines the comprehensive methodology of the proposed model. Figure 1 illustrates the proposed architecture, comprising two main stages: the Segmentation and Regeneration modules. The Segmentation module employs the SegUnet model to precisely segment occlusions from the face. Following this, the Regeneration module utilizes the GAN-based FUGAN model to reconstruct the facial parts obscured by these occlusions. Subsequent subsections provide an in-depth description of the model, including its architecture and training strategy.

6.1 Dataset Creation

A GAN-based model necessitates an extensive dataset for effective training. However, since there is a lack of publicly available datasets suitable for training, a solution to this challenge involved generating a synthetic dataset. This synthetic

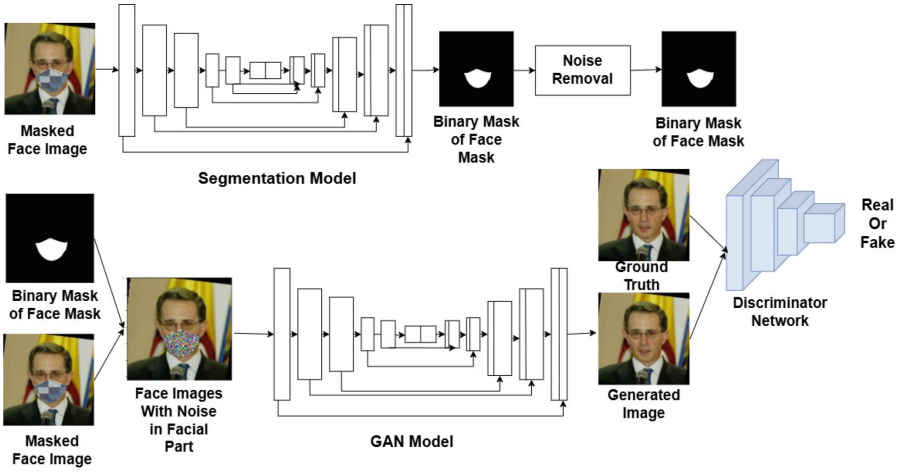


Fig. 1. Proposed architecture of our model

dataset was constructed using the CelebFaces [11] Attributes dataset, employing various types of masks placed over the mouth area. Face landmark detection facilitated the precise positioning of these masks, determining both their key positions and tilt angles, which were then overlaid onto the faces. Approximately 20,000 face images were gathered from the CelebA dataset for this purpose. In the creation of the synthetic dataset, a binary map representing occlusions was generated. The entire dataset was then divided into 70% for training and 30% for testing and validation purposes. This division enabled effective training on one portion while assessing performance on the other.

6.2 Architecture of Proposed GAN Model

The GAN model comprises of two main components: the generative model and the discriminator model. In the generative model, a modified U-Net architecture is used, drawing inspiration from the base U-Net model [15]. The ability of the suggested architecture to integrate multi-scale contextual information through its downsampling and upsampling paths enables a thorough understanding of the input image. It combines low-level spatial information from the encoder with high-level features from the decoder, allowing for precise localization and detailed output generation. Additionally, the skip connections in U-Net facilitate the flow of gradients during backpropagation, aiding in the training of deeper networks. In our scenario, since the mask detail is crucial and diverse complex information related to the face mask needs to be captured for precise facial part reconstruction, we have customized the encoder part. It comprises five blocks of convolution layers, with two blocks allocated for the bottleneck, while in the decoder, we also utilize five blocks. We have selected five blocks for both the encoder and decoder because, in the future, we plan to use the model

for real-time face unmasking. Faces from CCTV footage or live video are usually not high resolution, and the face images from each frame similarly lack high resolution. To make the model effective, we avoided excessive depth, starting with image dimensions of 128×128 and gradually decreasing to 4×4 to capture core pixel information. In the contracting path, each encoder block typically includes two convolutional layers followed by downsampling operations like max pooling to reduce feature map dimensions. Spectral normalization, Batch-Instance Normalization, and Leaky ReLU are applied to each convolutional block. The last two encoder blocks have slight modifications, incorporating self-attention-based convolutional layers [2] instead of the first convolutional layer. Self-attention blocks allow the network to accurately infer missing parts by utilizing information from all regions, ensuring consistency and coherence in the synthesized textures and patterns by effectively capturing long-range dependencies. This makes them especially useful for tasks such as image inpainting. In the expanding path, the decoder architecture mirrors the encoder blocks, except for the last two encoder blocks, where convolutional layers are replaced by upsampling operations. The bottleneck block serves as a bridge between the encoder and decoder paths, capturing essential image features. It typically consists of two convolutional layers, each paired with spectral normalization, Batch-Instance Normalization, and Leaky ReLU. The output layer employs one convolutional layer with a tanh activation function and three color channels for facial part regeneration. The segmentation architecture mirrors the generative model, except for the last layer, which uses a sigmoid activation function with one color channel for generating segmentation maps. One discriminator is employed, following the architecture in pix2pix, which penalizes dissimilar structures at a patch scale of 70×70. The discriminator's role is to compel the editing generator to produce visually plausible and semantically consistent images. Training of the discriminator and face generator modules occurs simultaneously. In Table 1, a detailed architecture of the proposed GAN model is described.

6.3 Training Strategy

In order to effectively train the regenerative module, a variety of loss functions are employed, and each output is combined with a multiplicative factor to aid in the convergence of the neural network, resulting in optimal performance across diverse real-world scenarios. When training the GAN model with face images containing masks positioned according to corresponding segmented masks, these images are fused and manipulated to introduce noise in the masked regions. The resulting images are then fed into the GAN network as input, generating synthesized images. In Fig. 2 some example of fuse images are given. To compel the editing generator to produce realistic missing content, a reconstruction loss is utilized, which is a combination of L1 loss with a fixed value of $\lambda_{rc} = 100$ [17], alongside structural similarity loss (SSIM). The equation of $\mathcal{L}_{total}$ combination of l1 and SSIM loss is given below.

$$\mathcal{L}_{total} = \lambda_{rc}\mathcal{L}_{l_1} + \mathcal{L}_{SSIM} \tag{1}$$

Table 1. Different Blocks Of FUGAN

Encoder Block	Name of Components	Input	Channel (C)
EN1,	Convolutional layer with spectral normalization, Batch Instance Normalization, Leaky relu and maxpooling layer	$128^2 \times 3$	32
EN2,		$64^2 \times 32$	64
EN3		$32^2 \times 64$	128
EN4,	self attention block, Convolutional layer with, spectral normalization, Batch Instance Normalization Leaky relu and maxpooling layer	$16^2 \times 128$	256
EN5		$8^2 \times 256$	512
Decoder Block	Name of Components	Input	Channel (C)
DEC5,	upsampling layer, concatenation layer, Convolutional layer with spectral normalization, Batch Instance Normalization, Leaky relu	$4^2 \times 512$	256
DEC4,		$8^2 \times 256$	128
DEC3,		$16^2 \times 128$	64
DEC2,		$32^2 \times 64$	32
DEC1		$64^2 \times 32$	32
Bottleneck Block	Name of Components	Input	Channel (C)
BN1,	Convolutional layer with spectral normalization, Batch-Instance Normalization, Leaky relu	$4^2 \times 512$	512
BN2		$4^2 \times 512$	512
Output Block	Name of Components	Input	Channel (C)
Output	Convolutional layer with 3 color channel with tanh activation	$128^2 \times 32$	3

Here L_{l_1} loss is defined as pixel difference between original (I_{gt}) and generated ($I_{generate}$) images. In the below equation, the calculation of the l_1 loss is presented.

$$\mathcal{L}_{l_1} = |I_{generate} - I_{gt}| \tag{2}$$

SSIM loss can be obtained by difference between 1 and SSIM value between original and generated fake image. More minimal SSIM loss more the image is accurate to near. This can be written a in below equation.

$$\mathcal{L}_{SSIM} = 1 - SSIM(I_{generate}, I_{gt}) \tag{3}$$

To train the discriminator D_{whole_region}, we use loss function $\mathcal{L}_D^{whole\_region}$. This assists in ensuring that the generator generates counterfeit images that are struc-

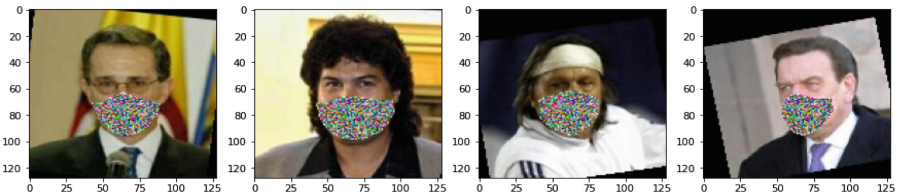


Fig. 2. Fused Images

turally similar to the original image. This loss function can be expressed as below:

$$\begin{aligned}\mathcal{L}_D^{whole\_region} = & -\mathbb{E}_{I_{gt}\epsilon O} \log D_{whole\_region}(I_{generate}, I_{gt}) \\ & + \mathbb{E}_{I_{generate}\epsilon S} \log(1 - D_{whole\_region}(I_{generate}))\end{aligned} \tag{4}$$

Here O and S denote real and synthesized image sets, respectively. In order to train the GAN setup the generator tries to fool the discriminator by minimizing the $\mathcal{L}_{total}$ and $\mathcal{L}_D^{whole\_region}$ loss function. However, the $\mathcal{L}_{total}$ loss function alone does not effectively produce realistic content in the deep pixels of the missing region; instead, it primarily contributes to generating structural similarities. To address this limitation, a modified loss function, denoted as $\mathcal{L}_{total}^{modified}$, is introduced. This modified loss function incorporates perceptual and adversarial loss into the final formula. The GAN model is then fine-tuned using this updated loss function, allowing it to generate pixel information without starting from scratch. To ensure that the generator's output closely resembles the features of the ground truth, a pre-trained VGG-19 network is employed as a fixed perceptual network. This network penalizes outputs that lack perceptual reasonability by calculating a feature-level distance measure between the intermediate feature maps of the original image and those generated by the model, based on the pre-trained network. To ensure the modified total loss function ($\mathcal{L}_{total}^{modified}$) effectively trains the neural network, we have assigned equal weights to the L1 loss, SSIM loss, and perceptual loss. This approach allows the model to reduce pixel error (L1 loss), enhance perceptual similarity by preserving structural information (SSIM loss), and maintain high-level feature consistency for better semantic similarity (perceptual loss) in a balanced way during the training phase, so that, each loss function can contribute their features equally, to generate images that are both visually and semantically coherent. Perceptual loss can be defined as:

$$\mathcal{L}_{perc} = \sum_j ||V_j(\hat{y}) - V_j(y)|| \tag{5}$$

Here $V_j(\hat{y})$ represents the activation's of the j^{th} layer of the VGG network when processing the image Y generated from Generative module. And $V_j(y)$ represents the activation's of the j^{th} layer of the VGG network when processing the ground truth or original image Y. We calculate another loss value from $\mathcal{L}_{adv}^{whole\_region}$ defined as binary crossentropy loss between image structure and generated image. It can be mathematically expressed as:

$$\mathcal{L}_{adv}^{whole\_region} = -\mathbb{E}_{I_{generate}\epsilon S} \log D_{whole\_region}(I_{generate}) \tag{6}$$

Final loss function $\mathcal{L}_{total}^{modified}$ can be expressed as follows:

$$\mathcal{L}_{total}^{modified} = \lambda_{rc}((\mathcal{L}_{l_1} + \mathcal{L}_{SSIM}) + \mathcal{L}_{perc}) + \mathcal{L}_{adv}^{whole\_region} \tag{7}$$

We have implemented the model in TensorFlow and taken 128×128 image dimension for training and testing of FUGAN model. For training the model effectively kaggle notebook is used with T4 GPU.

7 Result and Discussion

In this section, we have analyzed results, generated by our FUGAN model and compare with other state-of-the art image editing methods by quantitatively and qualitatively. For comparison we used several key metrices such as SSIM (Structural similarity index measure) [18], PSNR (Peak signal to noise ration) [6], FID (Frechet Inception Distance) [12] and BRISQUE (Blind or Referenceless Image spatial Quality Evaluator) [13]. In the below subsection a detail analysis of different result is given with some experimental image result.

7.1 Quantitative Analysis

Various metrics are utilized for quantitative assessment, including PSNR, SSIM, FID, and BRISQUE, to gauge the quality of generated images. PSNR, a full-reference metric, measures pixel-level differences for image reconstruction, expressed noise or distortion in decibels as a ratio of the pixel value of the generated and original image. SSIM evaluates the perceptual and structural similarity between two images, considering texture, edges, and contrast. FID assesses the similarity between generated and real images using a pre-trained InceptionV3 network, with lower scores indicating higher quality. BRISQUE, a non-reference metric, evaluates spatial image quality on a scale of 0 to 100, with lower scores indicating higher quality. These metrics quantitatively assess how faithfully unmasked images resemble their originals, aiding in model evaluation. Table 2 presents scores of different metrics comparing our model with state-of-the-art counterparts. Noteworthy models for comparison include those proposed by NIZAM UD DIN et al. [17], Akhil Kumar et al. [9], Farnaz Farahanipad et al. [5], Hitoshi Yoshihashi et al. [19], and Divyam Gupta et al. [8]. The results demonstrate that the self-attention-based block facilitates the generation of realistic images with minimal iterations, surpassing other models in SSIM, PSNR, and BRISQUE metrics. From the Table 2, it is observed that the model proposed by Hitoshi Yoshihashi et al. [19] has an SSIM score of 0.926, which is nearly equal to ours, but the BRISQUE score of that model far behind from our's model score 27.990. In the PSNR metric section, our model achieved 31.748, which is the best among all. However, our model falls short in FID, as FID emphasizes image quality measured by InceptionV3, capturing detailed pixel accuracy, and our model does not generate detailed in depth pixel information but focuses on overall realism in facial images. Nonetheless, besides FID, our model outperforms others overall. In the qualitative analysis section below, some regenerated images are presented alongside those from other models for further examination.

7.2 Qualitative Analysis

To qualitatively analyze the FUGAN model, our proposed model was trained for up to 300 iterations to generate realistic images efficiently. In the figures

Table 2. Quantitative Analysis

Proposed Model	SSIM	PSNR(dB)	FID	BRISQUE
NIZAM UD DIN et al. [17]	0.864	26.19	3.548	37.85
Akhil Kumar et al. [9]	0.91231	30.9879	6.8	35.644
Farnaz Farahanipad et al. [5]	0.89	28.65	13.63	36.52
Hitoshi Yoshihashi et al. [19]	0.926	28.789	9.4	29.547
Divyam Gupta et al. [8]	0.85752	24.998	5.64	39.77
FUGAN	**0.928**	**31.748**	4.4	**27.990**

denoted as Fig. 3, we've displayed the training result of the generator and discriminator loss function with $\mathcal{L}_{total}$ generator loss (sub-figure a and b) and with modified generator loss function $\mathcal{L}_{total}^{modified}$ (sub-figure c and d). The visual data illustrates a notable trend: as the iteration count increases, the generator loss decreases, indicating an enhancement in the realism of the generated facial features. Concurrently, when the regenerated image closely resembles the ground truth image, there's a gradual decline in the discriminator loss over time. These observations are clearly depicted in Fig. 3. Additionally, as depicted in sub-figure c, it is evident that the generator loss experiences a significant decrease up to epoch 138, followed by a sharp increase. Despite subsequent attempts to minimize the loss, they prove unsuccessful. Therefore, the model state closest to epoch 138, which exhibits the best performance, is selected as the final model for further testing. In Fig. 4, we have presented some images generated in different training iterations to visualize how the facial parts are reconstructed in a step-by-step manner by learning from ground truth images. Images generated by FUGAN model trained with $\mathcal{L}_{total}$ and $\mathcal{L}_{total}^{modified}$ loss function is presented in Fig. 5. From the figure it has been observed that Figs. 5(a) and 5(a) generated by FUGAN model trained with $\mathcal{L}_{total}$ loss function and Figs. 5(b) and 5(b) generated by FUGAN model fine tuned using modified loss function $\mathcal{L}_{total}^{modified}$. From these figures it can be observed that modified loss function helps the model to generate the deep pixel information means skin color texture related to original face image color. A comparison of image generation is given in Fig. 6. The sequence in the figure displays masked face images, ground truth, images regenerated by the FUGAN model, and images generated by various state-of-the-art models such as those proposed by NIZAM UD DIN et al. [17], Divyam Gupta et al. [8], and Hitoshi Yoshihashi et al. [19]. Here, we introduced only three SOTA models for qualitative analysis as other models lag from our result in quantitative section. Another set of images is presented in Fig. 7, comprising unknown images sent to the FUGAN model for testing purposes. Comparing the results in Fig. 6, it is evident that our model outperforms others in regeneration even with a shorter training duration. This regeneration capability of the FUGAN model is

attributed to its self-attention block, which reconstructs structural information swiftly. However, it's noted that images produced by FUGAN may lack some pixel-level information, resulting in a higher FID score as detailed in Sect. 7.1. Some distortion may be observed in pixel-level accuracy, but overall, the images appear realistic, making the model suitable for face unmasking tasks. Examining the generated images reveals that our model consistently maintains the original color composition during training phases across different lighting conditions, particularly regarding facial regeneration. Additionally, our model demonstrates the capability to accurately regenerate the bread of facial features for training images. For testing purposes, we have used the Jetson Xavier Nx module. This helps us identify our proposed model's capability in small factors without quantization and serialization techniques. Figure 7 shows that the FUGAN model can recover faces with hidden facial expressions. However, sometimes, there might be a mismatch in facial color in the regenerated part only, which is not critical for unmasking purposes. The proper reconstruction of facial structure, regardless of face color, aids in identifying individuals, which is valuable for our objectives. While the FUGAN model generally outperforms other models, it occasionally struggles to reconstruct facial features accurately in certain scenarios within the testing dataset, which is given in Fig. 8. From the experiment, we have found that instances arise where the model fails to adequately regenerate facial components when the facial mask size surpasses that of the actual facial area artificially applied to faces. Furthermore, discrepancies occur when individuals possess larger-than-average facial structures, causing the FUGAN

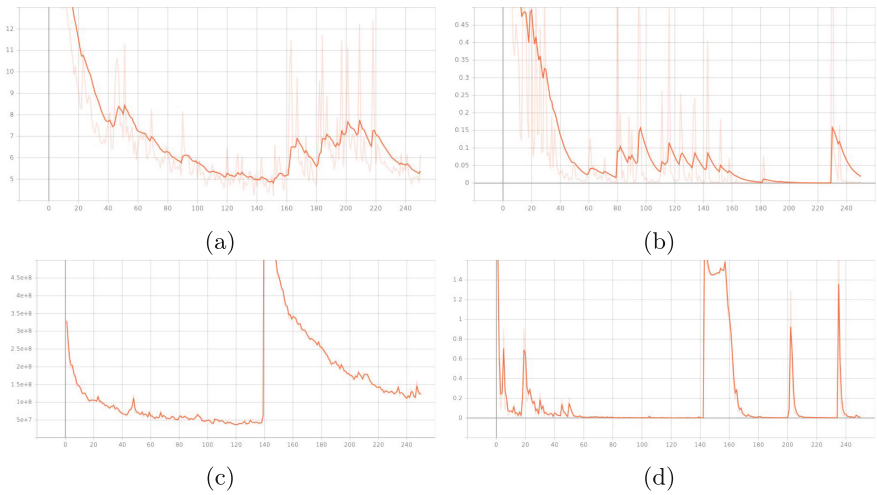


Fig. 3. Figure a and b describe FUGAN model's training generator loss and discriminator loss with respect of number of epoch. On contrary figure c and d describe FUGAN model's training generator loss and discriminator loss with respect of number of epoch with modified generator loss function.

Fig. 4. Image produced by our FUGAN model during training iteration.

Fig. 5. Image produced by our FUGAN model trained with different loss functions.

model to inadequately align regenerated features like the nose and mouth, albeit maintaining coherence with the overall facial structure. Challenges emerge when images contain more than two faces with facial masks, resulting in the model's inability to regenerate all faces. These identified shortcomings or limitations surfaced during the rigorous testing phase of our model. Based on this analysis, it can be concluded that our model surpasses other state-of-the-art models both quantitatively and qualitatively.

Fig. 6. Comparison of generated face images. In the figure from left to right, images are masked face image, Ground Truth, Generated Face Image(ours), NIZAM UD DIN et al. [17], Divyam Gupta et al. [8], Hitoshi Yoshihashi et al. [19].

Fig. 7. More example of generated Images produced by our FUGAN model.

Fig. 8. Shortcomings of regenerate of facial part.

8 Future Work

Looking ahead, there are promising avenues for further research and advancements in face mask removal. This includes enhancing our model's effectiveness in handling complex scenarios, such as significant occlusion or intricate mask designs. In the future, we aim to expand this work by refining the facial color and texture using the generated facial components. We also explore the possibility of inferring facial expressions from facial structure and eye expressions. In the future, we will try to enhance our model more effectively, which can handle live images captured by webcam or CCTV footage in the natural environment and unmask the faces covered with masks.

9 Conclusion

In this study, we have introduced an innovative technique for removing large objects, specifically masks, from facial images without the need for user interaction and to fill the removed portions of the image a GAN-based image-inpainting method FUGAN is utilized for generating realistic face images. Our proposed training approach, utilized one discriminator and simple loss function, allowing for the gradual learning of global coherence and deep missing regions, resulting in realistic and structurally consistent outputs. Through both qualitative and quantitative evaluations, we have demonstrated that our model can produce facial images with high perceptual quality, particularly when addressing large missing areas, outperforming other cutting-edge image editing techniques.

References

1. Alsalamah, M., Amin, S.: Medical image inpainting with RBF interpolation technique. Int. J. Adv. Comput. Sci. Appl. **7** (2016). https://api.semanticscholar.org/CorpusID:44198224
2. Bello, I., Zoph, B., Vaswani, A., Shlens, J., Le, Q.V.: Attention augmented convolutional networks (2020). https://arxiv.org/abs/1904.09925
3. Bertalmío, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (2000). https://api.semanticscholar.org/CorpusID:308278
4. Biradar, R.L., Kohir, V.V.: A novel image inpainting technique based on median diffusion. Sadhana **38**(4), 621–644 (2013). https://doi.org/10.1007/s12046-013-0152-2
5. Farahanipad, F., Rezaei, M., Nasr, M., Kamangar, F., Athitsos, V.: GAN-based face reconstruction for masked-face. In: Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments, pp. 583–587. ACM (2022). https://doi.org/10.1145/3529190.3534774. https://dl.acm.org/doi/10.1145/3529190.3534774
6. Horé, A., Ziou, D.: Image quality metrics: PSNR vs. SSIM. In: 2010 20th International Conference on Pattern Recognition, pp. 2366–2369 (2010). https://doi.org/10.1109/ICPR.2010.579

7. Jiang, Y., Yang, F., Bian, Z., Lu, C., Xia, S.: Mask removal: face inpainting via attributes. Multimedia Tools Appl. (4), 1–13 (2022). https://doi.org/10.1007/s11042-022-12912-1
8. Kumar, A., Gupta, D., Kaushal, M., Sharma, A.: A UNET++ and CoGAN-based method to remove face masks from the masked faces. Research square (2023). https://doi.org/10.21203/rs.3.rs-3351025/v1
9. Kumar, A., Kaushal, M., Sharma, A.: SAM c-GAN: a method for removal of face masks from masked faces. SIViP **17**(7), 3749–3757 (2023). https://doi.org/10.1007/s11760-023-02602-2
10. Liu, J., Musialski, P., Wonka, P., Ye, J.: Tensor completion for estimating missing values in visual data. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 208–220 (2013). https://doi.org/10.1109/TPAMI.2012.39
11. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV), December 2015
12. Mathiasen, A., Hvilshøj, F.: Backpropagating through Fréchet inception distance (2021)
13. Mittal, A., Moorthy, A.K., Bovik, A.C.: No-reference image quality assessment in the spatial domain. IEEE Trans. Image Process. **21**(12), 4695–4708 (2012). https://doi.org/10.1109/TIP.2012.2214050
14. Nazeri, K., Ng, E., Joseph, T., Qureshi, F.Z., Ebrahimi, M.: EdgeConnect: generative image inpainting with adversarial edge learning. CoRR **abs/1901.00212** (2019). http://arxiv.org/abs/1901.00212
15. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. CoRR **abs/1505.04597** (2015). http://arxiv.org/abs/1505.04597
16. Simakov, D., Caspi, Y., Shechtman, E., Irani, M.: Summarizing visual data using bidirectional similarity. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008). https://doi.org/10.1109/CVPR.2008.4587842
17. Ud Din, N., Javed, K., Bae, S., Yi, J.: A novel GAN-based network for unmasking of masked face. IEEE Access **PP**, 44276–44287 (2020). https://doi.org/10.1109/ACCESS.2020.2977386
18. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004). https://doi.org/10.1109/TIP.2003.819861
19. Yoshihashi, H., Ienaga, N., Sugimoto, M.: GAN-based face mask removal using facial landmarks and pixel errors in masked region. In: Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, pp. 125–133. SCITEPRESS - Science and Technology Publications (2022). https://doi.org/10.5220/0010827500003124
20. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. CoRR **abs/1801.07892** (2018). http://arxiv.org/abs/1801.07892

Text2Street: Controllable Text-to-Image Generation for Street Views

Songen Gu[1,2], Jinming Su[3], Yiting Duan[3], Xingyue Chen[3], Junfeng Luo[3], and Hao Zhao[4(✉)]

[1] Institute of Software Chinese Academy of Sciences, Beijing, China
[2] School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
gusongen22@mails.ucas.ac.cn
[3] Meituan, Beijing, China
{duanyiting,chenxingyue02,luojunfeng}@meituan.com
[4] AIR, Tsinghua University, Beijing, China
zhaohao@air.tsinghua.edu.cn

Abstract. Text-to-image generation has made remarkable progress with the emergence of diffusion models. However, it is still a difficult task to generate images for street views based on text, mainly because the road topology of street scenes is complex, the traffic status is diverse and the weather condition is various, which makes conventional text-to-image models difficult to deal with. To address these challenges, we propose a novel controllable text-to-image framework, named **Text2Street**. In the framework, we first introduce the lane-aware road topology generator, which achieves text-to-map generation with the accurate road structure and lane lines armed with the counting adapter, realizing the controllable road topology generation. Then, the position-based object layout generator is proposed to obtain text-to-layout generation through an object-level bounding box diffusion strategy, realizing the controllable traffic object layout generation. Finally, the multiple control image generator is designed to integrate the road topology, object layout and weather description to realize controllable street-view image generation. Extensive experiments show that the proposed approach achieves controllable street-view text-to-image generation and validates the effectiveness of the Text2Street framework for street views.

Keywords: Text-to-image generation · Diffusion model · Multi-modal

1 Introduction

Text-to-image generation [10,18,27], as an essential task of computer vision that aims to coherent images solely based on textual descriptions. In recent years, great efforts [25,26] have been dedicated to text-to-image generation for common

S. Gu and J. Su—Equal contribution.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 130–145, 2025.
https://doi.org/10.1007/978-3-031-78172-8_9

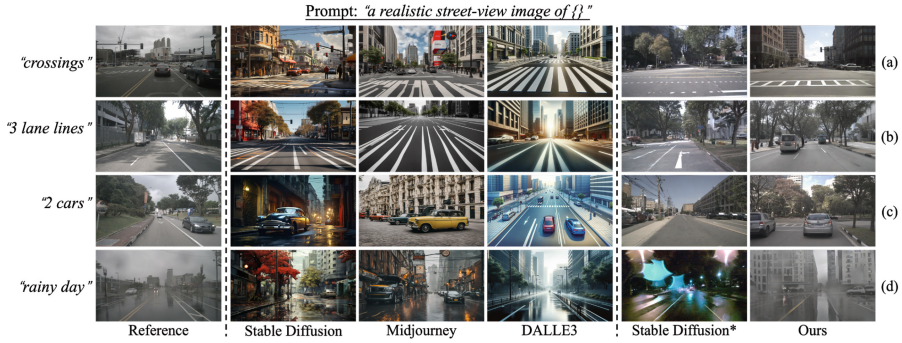


Fig. 1. Challenges of text-to-image generation for street views. There are three primary challenges: (1) complex road topology, including road structure in the first row and topological marks in the second row, (2) diverse traffic status, *e.g.*, varying traffic objects in the third row, and (3) various weather conditions like the rainy day in the last row. Note that Reference are original images from nuScenes [3], Stable Diffusion [28]/Midjourney [20]/DALLE3 [2] are tested on their official APIs, and Stable Diffusion* and Ours are finetuned on nuScenes.

scenarios, such as people and objects. Remarkable progress has been achieved, especially with the advent of diffusion models [14,28]. However, it is equally valuable to generate images in specialized domains, including autonomous driving [19], medical image analysis [4], robot perception [31], among others. Text-to-image generation for street views holds particular importance for data generation in the context of autonomous driving perception and map construction, yet it remains relatively unexplored.

Street-view text-to-image generation, as an underdeveloped task, faces several serious challenges, which can be categorized into three main aspects. Firstly, generating road topologies that adhere to traffic regulations presents a challenge. On one hand, as depicted in Fig. 1 (a), learning the road structure from text-image pairs is hindered by incomplete road structure information in the image, arising from limited imaging angles and frequent occlusions. This complexity makes it challenging for Stable Diffusion* [28], fine-tuned on nuScenes dataset [3], to generate expected images. On the other hand, as illustrated in Fig. 1 (b), generating lane lines that both comply with traffic regulations and match the count specified in the text poses a formidable challenge. Secondly, the representation of traffic status, a crucial element in street-view images, is often achieved through the number of traffic objects present. However, generating a specified number of traffic objects while adhering to motion rules using current models frequently falls short of expectations. As demonstrated in Fig. 1 (c), existing methods tend to lack sensitivity to precise numerical requirements. For instance, while our goal is to generate a road scene with two cars, the actual output from Stable Diffusion* often includes a significantly higher number of cars. Lastly, weather conditions are typically contingent upon the scene content, and direct image generation based on these conditions often yields vague or sub-

optimal outcomes, as depicted in Fig. 1 (d). Due to the presence of these three challenges, street-view text-to-image generation is a demanding task in computer vision.

To address previously mentioned challenges, we propose a novel controllable text-to-image framework for street views termed as **Text2Street**, illustrated in Fig. 2. Within this framework, we first introduce the lane-aware road topology generator, which utilizes text descriptions to create a local semantic map representing the intricate road topology. This generator also produces lane lines within the semantic map that conform to specified quantities and traffic regulations through a counting adapter. Subsequently, we introduce the position-based object layout generator to capture the diverse traffic status. By employing an object-level bounding box diffusion strategy, it generates the traffic object layout based on textual descriptions that adhere to specified quantities and traffic rules. Finally, the road topology and object layout are projected into the camera's imaging perspective through pose sampling. The projected road topology, object layout, and textual weather description are then integrated using the multiple control image generator to produce the final street-view image. Experimental validation confirms the effectiveness of our proposed method in generating street-view images from textual inputs.

The main contributions of this paper are as follows: 1) We propose a novel controllable text-to-image framework for street views, enabling the controls of road topology, traffic status, and weather conditions based solely on text descriptions. 2) We introduce the lane-aware road topology generator that generates specific road structures as well as lane topologies. 3) We propose the position-based object layout generator, capable of generating a specific number of traffic objects that comply with traffic rules. 4) We propose the multiple control image generator that can integrate road topology, traffic status, and weather conditions to achieve multi-condition image generation.

2 Related Work

In this section, we review related works in two aspects.

2.1 Text-to-Image Generation

In recent years, many methods [2,7,14,18,25–28,30,39] have been dedicated to dealing with the task of general text-to-image generation. For example, AlignDRAW [18] iteratively draws patches on a canvas, while attending to the relevant words in the description. GAWWN [27] synthesizes images given instructions describing what content to draw in which location based on generative adversarial networks [9]. DALLE [26] describes a simple approach for this text-to-image task based on a transformer that autoregressively models the text and image tokens as a single stream of data. DALLE2 [25] proposes a two-stage model: a prior that generates a CLIP [24] image embedding given a text caption, and a decoder that generates an image conditioned on the image embedding. DDPM [14] presents high-quality image synthesis results using diffusion

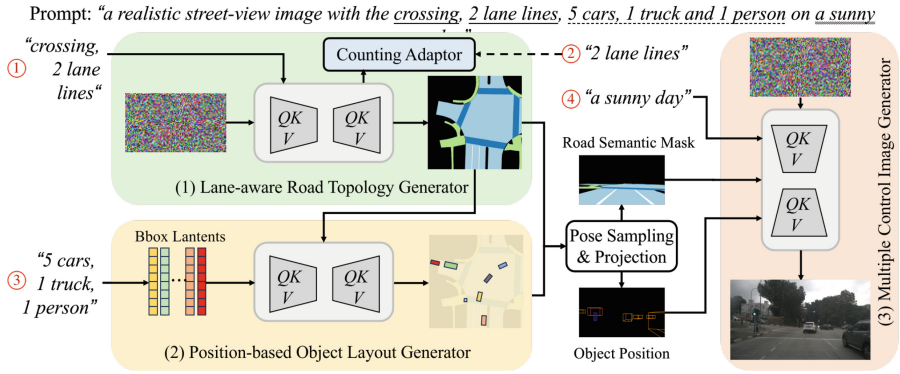


Fig. 2. Framework of **Text2Street**. We begin by introducing the lane-aware road topology generator, which utilizes textual input to create a local semantic map representing the intricate road topology with lane information. Next, we present the position-based object layout generator, which captures the diversity of traffic status and generates the traffic object layout. Subsequently, the road topology and object layout are projected into the camera's perspective through pose sampling. Finally, the projected road topology, object layout, and textual weather description are integrated through the multiple control image generator to produce the ultimate street-view image.

models [33], a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Stable Diffusion [28] applies diffusion models training in the latent space of pretrained autoencoders, and turns diffusion models into powerful and flexible generators for general conditioning inputs by introducing cross-attention layers into the model architecture. GLIDE [22] explores diffusion models for text-conditional image synthesis, comparing CLIP guidance and classifier-free guidance, and finds the latter preferable for photorealism and caption similarity. Imagen [30] employs a pretrained language model for encoding text, similar to GLIDE [22]. These methods have garnered remarkable results in general text-to-image generation. However, their effects in street-view text-to-image tasks are not as commendable.

Some works try to introduce fine-grained control to text-to-image generation models for better custom results. ControlNet [37] demonstrates the integration of spatial conditioning controls into pretrained text-to-image diffusion models leveraging zero convolutions. T2I-Adapter [21] proposes using lightweight adapters to align the internal knowledge of large-scale text-to-image models with external control signals. Uni-ControlNet [38] proposes a unified framework allowing for the use of multiple local and global controls in text-to-image diffusion models. However, using these methods relies on the condition image as input, which is hard to obtain during large-scale data generation. On the other hand, they still face the challenge of losing control when dealing with sophisticated condition cases (human skeletion [16] or complex bounding boxes [32]). In our work, we extend these control methods by incorporating condition generation

and introducing fine-grained controls, such as multi-control and an object position encoder.

2.2 Street-View Image Generation

There has been a recent surge in the study of methods for street-view image generation. For example, SDM [35] processes semantic layout and noisy image differently. It feeds noisy image to the encoder of the U-Net [29] structure while the semantic layout to the decoder by multi-layer spatially-adaptive normalization operators. BEVGen [34] synthesizes a set of realistic and spatially consistent surrounding images that match the birds-eye view (BEV) layout of a traffic scenario. BEVGen incorporates a novel cross-view transformation with spatial attention design which learns the relationship between cameras and map views to ensure their consistency. GeoDiffusion [6] translates various geometric conditions into text prompts and empower pre-trained text-to-image diffusion models for high-quality detection data generation and is able to encode not only the bounding boxes but also extra geometric conditions such as camera views in self-driving scenes. BEVControl [36] proposes a two-stage generative method that can generate accurate foreground and background contents. These methods typically require the input of BEV maps, object bounding boxes, or semantic masks to generate images. However, there is almost no research on generating street-view images relying solely on text. In this paper, we primarily focus on resolving the issue of street-view text-to-image generation.

3 The Proposed Approach

To address these challenges (*i.e.*, complex road topology, diverse traffic status, various weather conditions) in street-view text-to-image generation, we introduce **Text2Street**, a novel controllable framework illustrated in Fig. 2. In this section, details of the approach are described as follows.

3.1 Overview

Text2Street takes a street-view description prompt (*e.g.*, "*a street-view image with the crossing, 3 lanes, 4 cars and 1 truck driving on a sunny day*") as input and generates a corresponding street-view image. Prior to the main process, the input prompt is parsed by a large language model (*e.g.*, GPT-4 [23]) to extract descriptions of road topology, traffic status, and weather conditions, which are then fed into three main components. The first component is the lane-aware road topology generator, which takes the road topology description ("*crossing, 3 lanes*") as input and produces a local semantic map. The second component is the position-based object layout generator, which takes the traffic object description from the traffic status ("*4 cars and 1 truck*") as input and generates traffic object layout. The third component is the multiple control image generator, which takes road topology, object layout, and weather condition descriptions ("*a sunny day*") as input, and outputs an image that matches the original street-view description prompt.

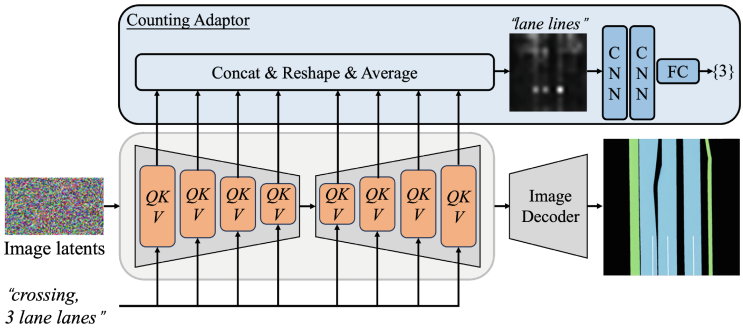


Fig. 3. Architecture of the lane-aware road topology generator. The generation part is similar to Stable Diffusion [28]. When generating, a noised latent is gradually denoised and sent to the image decoder to be decoded into a BEV image. In addition to text embedding as a condition, we utilize a counting adaptor, which is a shallow CNN that learns to regress the line count from an attention map.

3.2 Lane-Aware Road Topology Generator

For Stable Diffusion [28], directly generating images that comply with road topology, including road structure and lane topologies, is difficult. To address this, we introduce the lane-aware road topology generator (LRTG), as shown in Fig. 3. This generator does not directly produce road images; instead, it first creates a local semantic map describing the road structure, representing a complete regional-level road structure, including drivable areas, intersections, sidewalks, zebra crossings, etc. Simultaneously, to ensure the generated lane lines adhere to traffic regulations (*i.e.*, equidistant and parallel lanes), we characterize and generate lane lines on the semantic map, which is easier and more controllable than generating lane lines directly on perspective-view images. Furthermore, to ensure the number of lane lines aligns with the provided text, we incorporate a counting adapter for the precise generation of a specified number of lane lines. In LRTG, we only generate the semantic map, which serves as a crucial intermediary for street-view images, as further detailed in Sect. 3.4.

When generating local semantic maps, we utilize Stable Diffusion to encode road topology descriptions based on the CLIP [24] text encoder. Subsequently, the encoded input is then fed into cross-attention layers of U-Net [29] to denoise image latents, ultimately outputting the corresponding semantic map. Consistent with Stable Diffusion, the learning objective is as follows:

$$\mathcal{L}_{SD} = \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau(y))\|_2^2\right], \tag{1}$$

where $x \in \mathbb{R}^{H\times W\times 3}$ is the given images cropped from labeled semantic maps in RGB space, $\mathcal{E}(\cdot)$ refers to the encoder of pretrained autoencoders [8] and $z = \mathcal{E}(x)$ represents encoded image latents, z_t is from the forward diffusion process at the timestep t, y is the text prompt and $\tau(\cdot)$ represents the pretrained CLIP text encoder, the term ϵ denotes the target noise, and $\epsilon_\theta(\cdot)$ signifies the

time-conditional U-Net used for predicting the noise. This manner ensures the reasonable generation of road structures and lane line shapes within the semantic map.

For achieving precise control over the number of lane lines, the counting adapter f_{CA} gathers attention maps from all cross-attention layers of the U-Net. These scores are subsequently reshaped to match the same resolution and then averaged to yield attention features for all tokens. From these attention features, the ones corresponding to tokens "*lane lines*" are selected, which highlight where visual tokens correspond to lane lines. These selected features $\mathcal{F}_l$, undergo further processing through a shallow CNN with two convolutional layers with the kernel of 3×3, followed by one fully connected layer, which serves to regress the number of lane lines N_l. The learning objective for achieving precise control over the number of lane lines is as follows:

$$\mathcal{L}_{CA} = \|N_l - f_{CA}(\mathcal{F}_l)\|_2^2 . \tag{2}$$

Based on Eq. 1 and 2, LRTG can be jointly optimized to generate the local semantic map, encompassing both road structure and lane lines as required.

3.3 Position-Based Object Layout Generator

To ensure that generated images can depict diverse traffic conditions, we utilize the large language model to convert traffic status into the number of traffic objects (*e.g.*, car, truck, pedestrian, etc.). The full prompt used in large language model appears below:

```
You are a keyword extractor, your job is to extract object
information (eg. <N> cars, <N> trucks, etc) and return them
in JSON style. now the input is <fill in>
```

Then, the position-based object layout generator (POLG) is proposed to create an object layout based on the text description of object quantity, as demonstrated in Fig. 4. To guarantee a specified number of objects are generated, we incorporate an object-level bounding box diffusion strategy to generate positions of object bounding boxes. Simultaneously, to ensure the generated traffic objects comply with traffic rules, we incorporate the local semantic map from the LRTG into the box diffusion process. With POLG, we generate layout information for traffic

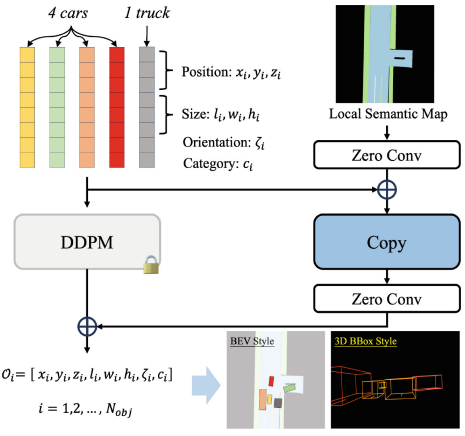


Fig. 4. Architecture of the position-based object layout generator. Note $\oplus$ means element-wise addition.

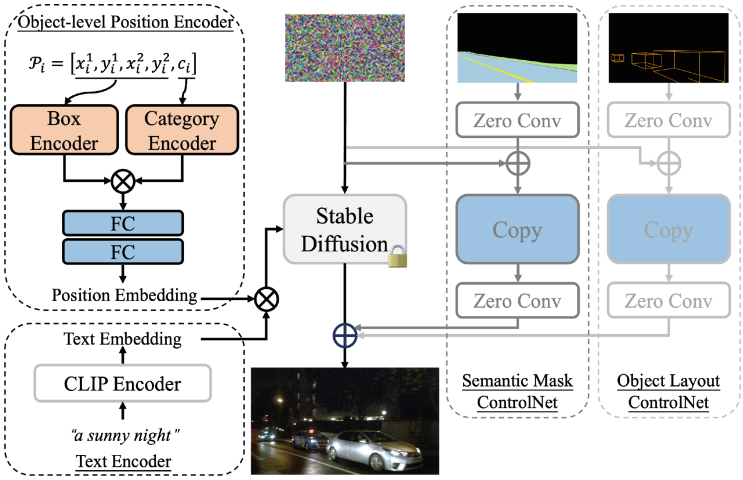


Fig. 5. Architecture of the multiple control image generator. Note $\otimes$, $\oplus$ means the concatenation and element-wise addition.

objects, which also serves as an intermediary for generating the final street-view images, as introduced in Sect. 3.4.

In the bounding box diffusion strategy, we first represent traffic objects as position vectors $\mathcal{O}_i = [x_i, y_i, z_i, l_i, w_i, h_i, \zeta_i, c_i]$ $(i = 1, 2, ..., N_o$, N_o is the number of objects), where x_i, y_i, z_i denote the coordinate of object position, l_i, w_i, h_i represent the object's size with length/width/height, ζ_i signifies the object's yaw angle, and c_i indicates the object's category. Subsequently, the position vector is diffused based on diffusion models DDPM [14]. Furthermore, to ensure objects adhere to traffic regulations (*e.g.*, cars must be driven on the road and not against traffic), we use ControlNet [37], incorporating the local semantic map from LRTG as a control into the POLG. Ultimately, the learning objective is as follows:

$$\mathcal{L}_{POLG} = \mathbb{E}_{o,m,\epsilon,t}\left[\|\epsilon - \epsilon_\theta(o_t, t, \mathcal{C}(m))\|_2^2\right], \tag{3}$$

where o represents the position vectors of the objects, o_t is from the forward diffusion process at the timestep t, m denotes the local semantic map, and $\mathcal{C}(\cdot)$ signifies the ControlNet. And other symbols are consistent with those in Eq. 1.

Based on Eq. 3, the layout information of traffic objects that meet the traffic status can be optimized and generated through POLG based on textual descriptions.

3.4 Multiple Control Image Generator

To produce images with realistic weather that align with road topology and traffic status, we introduce the multiple control image generator (MCIG), as depicted in Fig. 5.

To effectively utilize the previously generated local semantic map and traffic object layout, camera pose sampling and image projection are conducted before these two pieces of information enter MCIG. This results in a 2D road semantic mask $\mathcal{M}_r$ and traffic object layout map $\mathcal{M}_o$ from a perspective view, as shown in Fig. 2. The 2D traffic object layout maps are also represented as 2D traffic object position vectors $\mathcal{P} = \{\mathcal{P}_i\}_{i=1}^{N_o} = \{[x_i^1, y_i^1, x_i^2, y_i^2, c_i]\}_{i=1}^{N_o}$. The projection uses a conventional method based on intrinsic and extrinsic transformation, where the intrinsic parameters use fixed camera parameters, and the extrinsic parameters are sampled near the prior camera height.

As depicted in Fig. 5, MCIG comprises five modules: object-level position encoder, text encoder, semantic mask ControlNet, object layout ControlNet, and naive Stable Diffusion. The first four modules control image generation based on four different types of information, *i.e.*, 2D traffic position vectors, text describing the weather, 2D road semantic masks, and 2D traffic object layout maps.

The object-level position encoder encodes the 2D traffic object position vectors, including 2D bounding boxes and object categories, represented as:

$$\mathcal{PE}(\mathcal{P}) = f_{\mathcal{PE}}(\mathcal{BE} \otimes \mathcal{CE}). \tag{4}$$

The box encoder maps object bounding boxes to a higher-dimensional space, ensuring that the network can learn higher-frequency mapping functions and focus on the positions of each object. Specifically, the box encoder is an encoding function based on sine and cosine. The mathematical form of the encoding function is as follows:

$$\mathcal{BE}(p) = \left[\cdots, \sin(2^l \pi p), \cos(2^l \pi p), \cdots\right]_{l=0}^{L-1}, \tag{5}$$

where $\mathcal{BE}(\cdot)$ is applied to each component of the box (*i.e.*, $x_i^1, y_i^1, x_i^2, y_i^2$) of each object $\mathcal{P}_i$, and L is empirically set to 10. Simultaneously, the category encoder $\mathcal{CE}$ employs the CLIP text encoder to encode the object category (*e.g.*, "car"). Subsequently, the box encoding and category encoding are concatenated at the feature embedding dimension of each object. The concatenated features are then mapped to features with the same dimension as the original text encoder's embedding through a two-layer fully connected network $f_{\mathcal{PE}}(\cdot)$, serving as position embeddings. The text encoder, based on the CLIP text encoder, encodes the weather description text $\mathcal{T}$, resulting in text embeddings.

The object position encodings and weather text embeddings, upon concatenation at the token dimension, are fed into the cross-attention layer of Stable Diffusion, individually controlling the object position and weather during the image generation. Simultaneously, the semantic mask ControlNet and object layout ControlNet employ two similar ControlNets, utilizing images (*i.e.*, semantic masks and layout maps) as inputs to control the road topology and object layout during the street-view image generation. The learning objective function of MCIG is as follows:

$$\mathcal{L}_{MCIG} = \mathbb{E}_{\mathbb{P}}\left[\|\epsilon - \epsilon_\theta(z_t, t, \mathcal{PE}(\mathcal{P}), \tau(\mathcal{T}), \mathcal{C}(\mathcal{M}_r), \mathcal{C}(\mathcal{M}_o))\|_2^2\right], \tag{6}$$

Table 1. Comparisons with state-of-the-art methods on nuScenes validation dataset. The best result is in **bold** fonts.

	$S_{\text{FID}} \downarrow$	$S_{\text{CLIP}} \uparrow$	$S_{\text{road}} \uparrow$	$S_{\text{lane}} \uparrow$	$S_{\text{obj}} \uparrow$	$S_{\text{wea}} \uparrow$
Reference	0	18.00	89.67	48.30	52.33	97.00
Stable Diffusion [28]	67.91	16.38	83.50	21.33	29.83	98.50
Stable Diffusion 2.1 [1]	63.83	16.40	83.57	29.81	31.42	98.55
Attend-and-Excite [5]	69.63	16.40	83.00	30.67	31.00	98.83
Text2Street (Ours)	**53.92**	**17.81**	**84.17**	**35.17**	**46.33**	**99.67**

where $\mathbb{P}$ is the set of $\{\mathcal{E}(x), \mathcal{P}, \mathcal{T}, \mathcal{M}_r, \mathcal{M}_o, \epsilon, t\}$ for convenience of presentation.

Through the optimization of MCIG using Eq. 6, we obtain street-view images that conform to the initial prompt about road topology, traffic status, and weather conditions.

4 Experiments and Results

4.1 Experimental Setup

Datasets. To validate the performance of the proposed approach, we conduct all experiments on the public autonomous driving dataset nuScenes [3]. nuScenes dataset contains 1,000 street-view scenes (700/150/150 for training/validation/testing, respectively). Each scene comprises approximately 40 frames, with each frame encompassing six RGB images captured by six cameras mounted for panoramic view on the ego vehicle. Additionally, each frame is with a labeled semantic map with 32 semantic categories. For the sake of simplicity and clarity, we solely use images captured by the `FRONT` camera in all experiments.

Evaluation Metrics. To comprehensively evaluate the text-to-image generation for street views, we assess the generation results from the image level and attribute level.

For image-level evaluation, we use Fréchet Inception Distance (FID) S_{FID} [13] to measure image fidelity, and CLIP score S_{CLIP} [12] to image-text alignment. Please refer to related works [12,13] for computation details.

In the attribute-level evaluation, we primarily measure the accuracy of text-to-image street-view generation in four aspects: road structure, lane line counting, traffic object counting, and weather conditions. For these four metrics, we train four neural networks on nuScenes dataset to evaluate scores of generated images. Specifically, a two-class classifier based on ResNet-50 [11] is trained for road structure accuracy S_{road} to distinguish whether the road structure in street-view RGB images is an "intersection" or "non-intersection". For the accuracy of lane line counting S_{lane}, a six-class classifier is similarly trained on ResNet-50 to distinguish whether the number of lane lines in street-view RGB images is equal to 0, 1, 2, 3, 4, or ≥ 5. For the accuracy of traffic object counting S_{obj}, an

object detector based on YOLOv5 [15] is trained to evaluate the number of traffic objects in street-view RGB images. For the accuracy of weather conditions S_{wea}, a four-class classifier is also trained on ResNet-50 to distinguish whether the weather conditions in street-view RGB images are sunny day, sunny night, rainy day, or rainy night. All models are trained on nuScenes training dataset and used as evaluation metrics for attribute-level evaluation of street-view image generation.

Training and Inference. During the training phase, we separately train three generators, *i.e.*, lane-aware road topology generator (LRTG), position-based object layout generator (POLG) and multiple control image generator (MCIG). LRTG and MCIG are initialized with Stable Diffusion[1] [28], POLG is random initialized based on DDPM[2] [14] modified with ControlNet [37], and CLIP [24] text encoder are fixed with pretrained weights. For these three generators, we train them with AdamW [17] optimizer for 10 epochs with a learning rate $1e^{-4}$, a batch size of 32. In addition, semantic maps in LRTG are resized to the resolution of 512×512, and RGB images in MCIG are resized to the resolution of 895×512. In the inference phase, the three generators perform inference sequentially, with the denoising iterations all set to 30 times.

4.2 Comparisons with State-of-the-Art Methods

We compare our approach with several state-of-the-art algorithms in text-to-image generation, including Stable Diffusion [28], Stable Diffusion 2.1 [1] and Attend-and-Excite [5] on nuScenes validation dataset, as listed in Table 1. These methods are all finetuned on nuScenes training dataset. Note that we have also listed the performance on the nuScenes validation dataset as the "Reference".

Comparing the proposed method with state-of-the-art methods, we can see that our method consistently outperforms other methods across almost all metrics from Table 1. Significantly, our method outperforms all others on attribute-level metrics (*i.e.*, S_{road}, S_{lane}, S_{obj} and S_{wea}), demonstrating its superior controllability for fine-grained text-to-image street-view image generation. Specifically, our method shows a obviously 4.50%, 14.91% improvement on metric S_{lane} and S_{obj} compared to the second best performance. Additionally, our method also performs better on image-level metrics (*i.e.*, S_{FID} and S_{CLIP}), reflecting its superior overall generation quality and image-text consistency. Overall, these observations validate the effectiveness of our proposed method for controllable image generation for street views.

Visual examples generated by our method are illustrated in Fig. 6. From Fig. 6, it is evident that our method yields superior results in dealing with varying road structures (1st and 4th rows), different numbers of lane lines (1st and 3rd rows), diverse numbers of traffic objects (1st and 2nd rows), and various weather conditions (2nd and 3rd rows) compared to other methods. This indicates that our method can effectively generate street-view images only based on

[1] https://huggingface.co/runwayml/stable-diffusion-v1-5.
[2] https://huggingface.co/docs/diffusers/api/pipelines/ddpm.

Fig. 6. Qualitative comparisons of Stable Diffusion [28] and our approach **Text2Street**. These two methods are finetuned on nuScenes [3] dataset. Note that in nuScenes, the double yellow/white lane line counts as one lane line, not two lane lines.

text, and also implies its controllability and superiority in street-view text-to-image generation.

4.3 Ablation Analysis

To assess the effectiveness of individual components, we carry out ablation experiments on nuScenes validation dataset, comparing the performance variations within the proposed approach.

Firstly, to validate the effectiveness of the lane-aware road topology generator (LRTG), we introduced three models for ablation comparison. The first model, termed as "Baseline", is a naive multiple control image generator (MCIG) with only the text encoder, which actually is a Stable Diffusion model. The second model, named "A_1", is based on the "Baseline" with the addition of LRTG excluding lane line control. The third model, "A_2", adds LRTG with lane line control to the first model. The comparison of these three models is presented in the first three rows of Table 2. It can be observed that the introduction of road

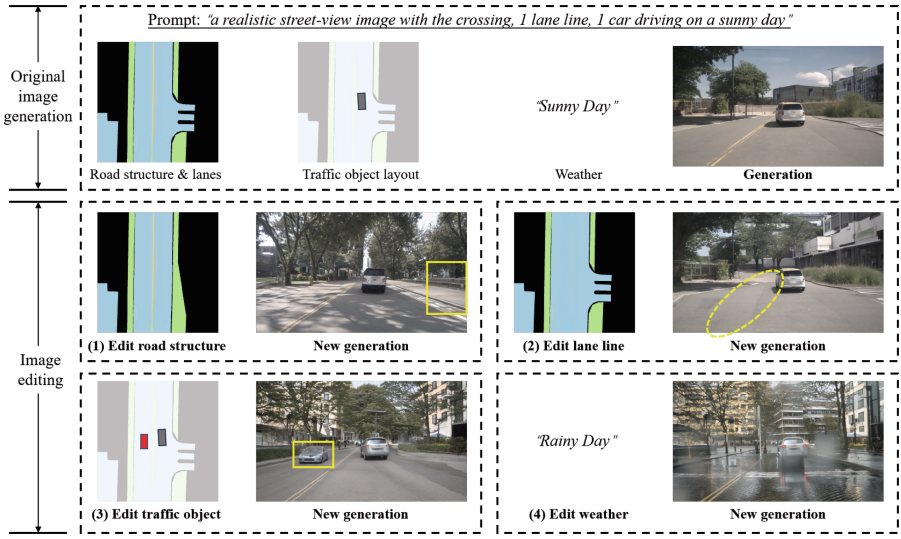


Fig. 7. Examples of different image editing operations by our approach.

structure control ("A_1") significantly improves the S_{road} metric, and the addition of both road structure and lane lines ("A_2") controls further enhances both S_{road} and S_{lane} metrics. This confirms the effectiveness of LRTG in controlling road topology.

Secondly, to validate the effectiveness of the position-based object layout generator (POLG), we add POLG to "Baseline", termed as "B". Comparing the first and fourth rows of Table 2, it is evident that the inclusion of POLG significantly improves the metric S_{obj}, demonstrating the control capability of POLG in traffic object generation.

Thirdly, to verify the compatibility of different modules, we also list the model "C" (*i.e.*, Text2Street), which combines all three modules. As can be seen from the last row of Table 2, "C" achieves the best performance across all metrics, confirming the compatibility among different modules.

Table 2. Performance of different settings of the proposed method on nuScenes validation dataset.

	LRTG	POLG	MCIG	$S_{\text{FID}} \downarrow$	$S_{\text{CLIP}} \uparrow$	$S_{\text{road}} \uparrow$	$S_{\text{lane}} \uparrow$	$S_{\text{obj}} \uparrow$	$S_{\text{wea}} \uparrow$
Baseline			✓	67.91	16.38	83.50	21.33	29.83	98.50
A_1	✓		✓	60.45	**17.82**	**85.17**	33.16	30.00	98.50
A_2	✓		✓	59.85	**17.82**	84.67	34.01	31.50	98.53
B		✓	✓	59.60	17.81	84.40	29.33	41.50	98.67
C (**Text2Street**)	✓	✓	✓	**53.92**	17.81	84.17	**35.17**	**46.33**	**99.67**

Table 3. Performance of YOLOv5 without/with the data augmentation of our method on nuScenes validation dataset.

	mAP ↑	Precision ↑	Recall ↑
YOLOv5	46.30	81.77	70.91
YOLOv5 with Text2Street	**47.83**	**84.45**	**72.37**

4.4 Text-to-Image Generation for Object Detection

To demonstrate the utility of street-view text-to-image generation for downstream tasks, we select object detection as a representative task. We use the proposed Text2Street to generate 30,000 images based on random prompts as a supplement to the original training data to train YOLOv5 on the nuScenes dataset, as listed in Table 3. The results indicate that the images generated by our method are beneficial for downstream street-view tasks, highlighting the potential of the street-view text-to-image generation.

4.5 Image Editing

In addition to street-view text-to-image generation, our approach also allows for modifications to local semantic maps, object layouts, or text, enabling the editing of road structures, lane lines, object layouts, and weather conditions in the originally generated RGB images, as depicted in Fig. 7.

5 Conclusion

In this paper, we propose a novel controllable text-to-image generation framework for street views. In this framework, we design the lane-aware road topology generator to exert control over the road topology in a text-to-map manner. Additionally, the position-based object layout generator is proposed to control the layout of traffic objects through a text-to-layout manner. Moreover, the multiple control image generator is built to integrate multiple controls to generate street-view images. Empirical results substantiate the effectiveness of our proposed approach.

References

1. Stability AI: Stable diffusion 2.1. https://huggingface.co/stabilityai/stable-diffusion-2-1
2. Betker, J., et al.: Improving image generation with better captions (2023)
3. Caesar, H., et al.: nuScenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11621–11631 (2020)

4. Chan, H.-P., Samala, R.K., Hadjiiski, L.M., Zhou, C.: Deep learning in medical image analysis. In: Lee, G., Fujita, H. (eds.) Deep Learning in Medical Image Analysis. AEMB, vol. 1213, pp. 3–21. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-33128-3_1
5. Chefer, H., Alaluf, Y., Vinker, Y., Wolf, L., Cohen-Or, D.: Attend-and-excite: attention-based semantic guidance for text-to-image diffusion models. ACM Trans. Graph. (TOG) **42**(4), 1–10 (2023)
6. Chen, K., Xie, E., Chen, Z., Hong, L., Li, Z., Yeung, D.Y.: Integrating geometric control into text-to-image diffusion models for high-quality detection data generation via text prompt. arXiv: 2306.04607 (2023)
7. Ding, M., et al.: CogView: mastering text-to-image generation via transformers. arXiv preprint arXiv:2105.13290 (2021)
8. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12873–12883 (2021)
9. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
10. Gregor, K., Danihelka, I., Graves, A., Rezende, D., Wierstra, D.: DRAW: a recurrent neural network for image generation. In: International Conference on Machine Learning, pp. 1462–1471. PMLR (2015)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
12. Hessel, J., Holtzman, A., Forbes, M., Bras, R.L., Choi, Y.: CLIPScore: a reference-free evaluation metric for image captioning. arXiv preprint arXiv:2104.08718 (2021)
13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol. 33, pp. 6840–6851 (2020)
15. Jocher, G.: YOLOv5 by Ultralytics, May 2020. https://doi.org/10.5281/zenodo.3908559. https://github.com/ultralytics/yolov5
16. Ju, X., Zeng, A., Zhao, C., Wang, J., Zhang, L., Xu, Q.: HumanSD: a native skeleton-guided diffusion model for human image generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 15988–15998 (2023)
17. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
18. Mansimov, E., Parisotto, E., Ba, J.L., Salakhutdinov, R.: Generating images from captions with attention. arXiv preprint arXiv:1511.02793 (2015)
19. Marti, E., De Miguel, M.A., Garcia, F., Perez, J.: A review of sensor technologies for perception in automated driving. IEEE Intell. Transp. Syst. Mag. **11**(4), 94–108 (2019)
20. Midjourney: Midjourney. https://www.midjourney.com
21. Mou, C., et al.: T2I-adapter: learning adapters to dig out more controllable ability for text-to-image diffusion models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 4296–4304 (2024)
22. Nichol, A., et al.: GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
23. OpenAI: GPT-4 technical report (2023)

24. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763. PMLR (2021)
25. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with CLIP latents. arXiv preprint arXiv:2204.06125 (2022)
26. Ramesh, A., et al.: Zero-shot text-to-image generation. In: International Conference on Machine Learning, pp. 8821–8831. PMLR (2021)
27. Reed, S.E., Akata, Z., Mohan, S., Tenka, S., Schiele, B., Lee, H.: Learning what and where to draw. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
28. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022)
29. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
30. Saharia, C., et al.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems, vol. 35, pp. 36479–36494 (2022)
31. Shi, Q., Li, C., Wang, C., Luo, H., Huang, Q., Fukuda, T.: Design and implementation of an omnidirectional vision system for robot perception. Mechatronics **41**, 58–66 (2017)
32. Shirakawa, T., Uchida, S.: NoiseCollage: a layout-aware text-to-image diffusion model based on noise cropping and merging. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8921–8930 (2024)
33. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning, pp. 2256–2265. PMLR (2015)
34. Swerdlow, A., Xu, R., Zhou, B.: Street-view image generation from a bird's-eye view layout. arXiv preprint arXiv:2301.04634 (2023)
35. Wang, W., et al.: Semantic image synthesis via diffusion models. arXiv preprint arXiv:2207.00050 (2022)
36. Yang, K., Ma, E., Peng, J., Guo, Q., Lin, D., Yu, K.: BEVControl: accurately controlling street-view elements with multi-perspective consistency via BEV sketch layout. arXiv preprint arXiv:2308.01661 (2023)
37. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3836–3847 (2023)
38. Zhao, S., et al.: Uni-ControlNet: all-in-one control to text-to-image diffusion models. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
39. Zhou, Y., et al.: LAFITE: towards language-free training for text-to-image generation. arXiv preprint arXiv:2111.13792 (2021)

Make an Image Move: Few-Shot Based Video Generation Guided by CLIP

Yonglong Huang[1], Cheng Yu[2], Nannan Li[1(✉)], Fuqin Deng[3], Ruiquan Ge[4], and Changmiao Wang[5]

[1] School of Computer Science and Engineering, Macau University of Science and Technology, Macau, China
nnli@must.edu.mo
[2] Chongqing University of Technology, Chongqing, China
[3] Wuyi University, Jiangmen, China
[4] Hangzhou Dianzi University, Hangzhou, China
gespring@hdu.edu.cn
[5] Shenzhen Research Institute of Big Data, Shenzhen, China

Abstract. The recent surge in interest surrounding text-to-video diffusion models highlights their capability to generate videos with both consistency and diversity. Current methods focus on leveraging large-scale datasets to align with training videos, while certain approaches explore the potential of zero-shot generation. Few-shot generative models adapt text-to-image model with temporal layers. They can capture the temporal motion and appearance with acceptable computational resources. However, the prevalent few-shot based approaches, which employ a singular prompt for training videos, typically result in inadequate control over complex backgrounds and multiple objects. To overcome this limitation, we introduce a novel component: the dual cross-attention layer. This component leverages CLIP for image feature extraction. The image feature and text feature will be processed independently in dual cross-attention layer, aiming to achieve image reference and enrich the training videos with additional information. This dual cross-attention mechanism empowers the diffusion model to learn both image and text information effectively, facilitating the generation of higher quality videos. Furthermore, we propose an innovative sampling method to enhance temporal consistency and stability of generative videos. Compared to other text-to-video generative models, our framework demonstrates superior efficiency in generating high quality videos with diverse styles. The code is available at https://github.com/FatLong666/MAIM.

Keywords: Diffusion Model · Text-to-video generation · Few-shot generation

Y. Huang and C. Yu—Co-first author.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 146–161, 2025.
https://doi.org/10.1007/978-3-031-78172-8_10

1 Introduction

In recent years, generative *text-to-image* (T2I) diffusion models, such as Stable Diffusion [3] have demonstrated remarkable capability in generating a wide range of diverse images from textual prompts. Moreover, methods like ControlNet [1], IP-adaptor [2] have further enhanced these models by introducing additional conditioning during inference, which called *text-to-image synthesis*. Recent works have increasingly focused on *text-to-video*(T2V) generation [4–6], which utilize textual prompts to generate videos that exhibit both content diversity and temporal consistency.

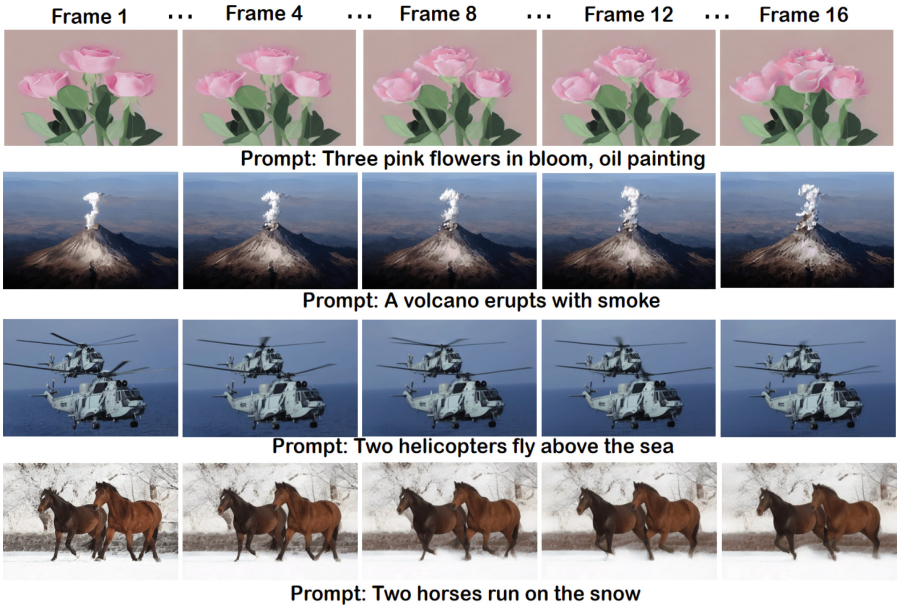


Fig. 1. Our text-to-video generation model results. **MAIM** is trained on few videos and generate videos on different style images. We have trained the model on four distinct types of motion videos, producing a unique video for each specific motion.

Current advancements in T2V generation have been marked by significant breakthroughs, notably by studies such as CogVideo [7] and Gen-1 [8], which have leveraged large-scale multimodal datasets for training. Although these models are capable of producing diverse, high quality videos from textual prompts, creating videos with specific custom motions and styles continues to be challenging. Several studies [9,10,27], extending the idea in zero-shot manner to balance the computational resources and generative quality. Nevertheless, transferring knowledge from the T2I domain to the T2V domain poses significant challenges in video consistency and stability. Few-shot generation models can overcome those weakness. Our work takes the study of few-shot generation further.

Tune-A-Video [11] employs a single video as training video and modifies the attention layers in the Latent Diffusion Model (LDM) [3], facilitating the generation of content and motion that closely mimics the original video. It's an

amazing work in video editing and generation. However, its efficacy is markedly dependent on the original video for reference, with the generated content being predominantly influenced by the referenced video. LAMP [12] takes one step forward in few-shot method, using a limited number of videos for training. It learns motion patterns from these videos based on a textual prompt and leverages the initial condition image to generate diverse array of videos.

LAMP significantly narrows the gap between the constraints of training resources and the capability to produce varied video content. Nevertheless, the fine-tuning process in LAMP is limited by its reliance on a simple textual prompt to guide the LDM in learning motion patterns and object information from the videos. This approach may exhibit limited control over the prompt and video content during inference, especially when the background of the image is complex, encompassing diverse styles and containing multiple objects. In [2], it proposes to utilize image features other than text features as the image reference. Drawing inspiration from this, we redesign it in LAMP and employ CLIP [13] to improve performance in generating consistency and diversity videos. It is significant to maintain the latent diffusion model's ability to learn motion patterns from the input video datasets and to ensure the temporal consistency of generative videos. There are two issues need to be solved. **Firstly**, the fine-tuning process which relies solely on a simple prompt to describe every frame of the training videos, may lead to over-fitting in the T2V diffusion model. During the inference process, T2V model may lack control over the prompt conditions and the object will be blurred or missing in subsequent generative video frames. **Secondly**, existing few-shot methods [11,12], constrained by their incomplete capture of image features, face challenges in generating videos that are with complex backgrounds or exhibit diverse styles.

Hence, We introduce **MAIM (Make An Image Move)**, a few-shot based text-to-video generation model, which is guided by CLIP and utilizes the novel *dual cross-attention layer* to enhance the interaction between text features and image features. To improve efficiency and reduce computational overhead, we extract image and text features from CLIP image encoder and its text encoder, process them independently and subsequently integrate them in a feed forward network, representing the innovation of our method compared to previous methods. Furthermore, our method includes a unique sampling technique, *latent-time-shift sampling*, which aims to construct the original noise from the latent space of the first frame with a shared noise. Figure 1 showcases the results of our T2V diffusion model, illustrating enhancements in both the quality and stability of the generative videos with two novel components. In summary, our key contributions are as follows:

- We introduce **MAIM**, a novel few-shot based text-to-video generation framework. Our method is capable of generating videos with complex backgrounds and multiple objects, demonstrating greater generalization capabilities compared to other approaches.
- We propose the *dual cross-attention layer* to decouple image and text features. The textual prompt is no longer the sole reference. Complementally,

we incorporate image features extracted by the CLIP image encoder to enrich the reference information.
– During the inference stage, We propose a *latent-time-shift sampling* method to guarantee both the flexibility and stability of the generated videos.

2 Related Work

2.1 Text-to-Image Diffusion Models

Lately, the domain of T2I generation has garnered considerable attention. The diffusion model [14] outperforms alternatives like GANs [13,15,16], VAEs [17,18]. Diffusion models and it's variant DDIM [19] are the foundation of the T2I generation task. Several works have been proposed recently, GLIDE [20] leverages classifier-free guidance to significantly advance image quality within diffusion framework. Simultaneously, DALL-E2 [21] employs the feature space of CLIP model to achieve substantial improvements in text-to-image alignment. It facilities a more nuanced and coherent generation of images from textual descriptions. Furthermore, Imagen [22] introduces a novel solution by employing cascaded diffusion models to create high-definition images which shows the versatility of diffusion models in handling complex T2I generation task. VQ-diffusion [23], LDM [3] explores the operating within the latent spaces of autoencoders. The latent space, being of a lower dimension, offers a more efficient approach for noise addition and denoising compared to working directly with pixel images. Our work is based on the LDM to decouple the attention layer and inject the image features.

2.2 Text-to-Video Diffusion Models

Text-to-video diffusion models have been proposed recently. T2V generative diffusion models can be mainly divided into three categories: large-scale models, zero-shot generative models, and few-shot generative models.

Large-Scale Models. Training large-scale models on extensive text-video pairs datasets, necessitates substantial computational resources. Such models, capable of generating diverse videos, exhibit impressive generalization capabilities. Imagen Video [24] and Make-A-Video [25] propose hierarchical structures, including separate key frames, interpolation and super-resolution models for sampling high fidelity video generation. Magic-video [26] advances the field by training a novel auto-encoder designed to address pixel jitter in generative videos. Similarly, AnimateDiff [5] also employs a cascading model approach, which is the same paradigm. Show-1 [6] combines pixel-based and latent-based video diffusion models for T2V generation. The implementation of these methodologies are supported by extensive video datasets like webVid-10M [27] and HD-VILA-100M [29], which present a substantial challenge for most researchers.

Zero-Shot Generative Models. Zero-shot text-to-video generation refers to a "training-free" approach, requiring no optimization or fine-tuning of the T2I pretrained model. T2V-Zero [9] syntheses a video with a simple textual prompt. It utilizes motion flow to modify the original latent and the frame cross-attention layer to ensure the temporal consistency. Furthermore, Free-bloom [10] and DirectT2V [27] attempt to leverage the capabilities of Large Language Model to generate a series of continues descriptions to guide the diffusion model to generate consistent video frames. Although zero-shot generation can relieve the computational cost, the motion pattern and detail of object still hard to maintain the temporal consistency.

Few-Shot Generative Models. Recent advancements in few-shot generation task, such as Tune-A-Video [11] and LAMP [12], training T2I diffusion model and incorporating novel temporal layers. These layers are designed to capture both appearance and motion information from the training data. Tune-A-Video [11] utilizes a reference video and employs diverse prompts to generate various videos, while maintaining the original motion patterns. However, Tune-A-Video is based on the template video and edits it with different content prompts. These template based methods [11,29,30] will restrict the freedom of generative video. LAMP proposes a motion learning model to capture the motion pattern from the training data and utilizes about 8∼16 videos to tune the pretrained T2I model. Unlike existing few-shot based methods, our framework aims to achieve image reference by incorporating image features extracted by CLIP image encoder.

3 Method

In this section, particularly in Subsect. 3.1, we provide fundamental knowledge about diffusion models. This is followed by Subsects. 3.2 and 3.3, which detail our methodology, describing the *dual cross-attention layer* and the *latent-time-shift sampling* module, respectively. An overview of our approach (**MAIM**) is shown in Fig. 2. Our work emphasizes cost-effective computational consumption. We aims to leverage the CLIP to extract image features and combine them with text prompts to enhance the consistency and diversity of the generative videos.

3.1 Preliminaries

In this section, we present to introduce the foundational concepts related to the diffusion model. The Denoising Diffusion Probabilistic Model (DDPM) [14] functions as a probabilistic model, while its variant, the Denoising Diffusion Implicit Model (DDIM) [19], which employs a deterministic sampling process to expedite sampling. DDIM follows a Markov chain structure and diffusion forward process incrementally introduces Gaussian noise into the initial data x_0:

$$q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right), \quad t = 1, \ldots, T. \tag{1}$$

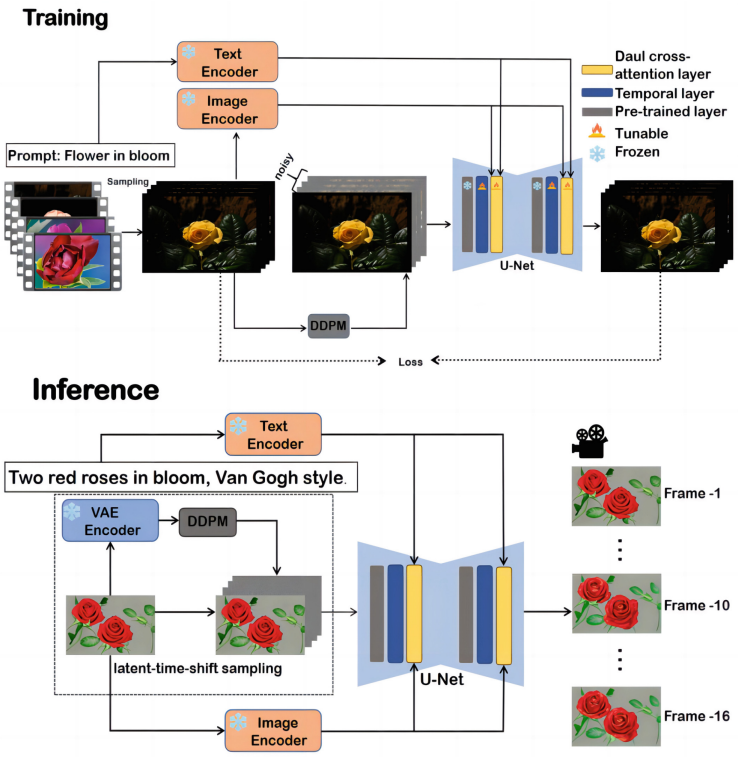


Fig. 2. The framework of **MAIM**. For the training stage, MAIM learns motion pattern and object information from the textual prompt and image features extracted by CLIP respectively. For the Inference stage, MAIM advances an image for the first frame and the framework will predict the subsequent frames.

$q\{x_t|x_{t-1}\}$ is the Markov transition, it can be conceptualized as a Gaussian distribution with $\beta_t \in (0,1)$. β_t is a hyperparameter. $t = 1, ..., T$ is the total time step of forward process. Forward process will destroy the initial data x_0 into $x_t \in (0, I)$. Diffusion model's goal is to learn the Gaussian transition in backward process:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \quad t = T, \ldots, 1. \tag{2}$$

For $t = T, ..., 1$ represents a backward process, which denoises the standard Gaussian noise x_t to x_0 that is a valid signal. The learnable parameters, denoted by θ, are optimized to ensure that the generative reverse process closely approximates the forward process.

3.2 Dual Cross-Attention Layer

Existing few-shot generation methods, such as LAMP and Tune-A-Video, utilize a sample prompt to characterize the training video. However, this approach

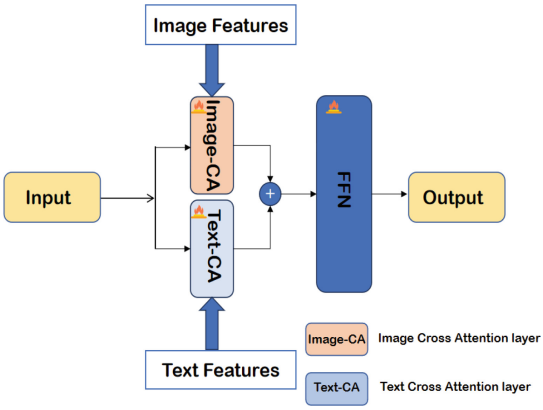


Fig. 3. Dual Cross-Attention Layer, Image features are extracted by CLIP image encoders and text features are encoded by CLIP text encoder. We train part of the parameters of the Text-CA layer while training the Image-CA layer.

inevitably omits some visual concepts presenting in the video frames. Consequently, we incorporate image features extracted by the CLIP to enhance image referencing capabilities, thereby improving generalization and semantic alignment between text and image. As illustrated in Fig. 3, we have extended the cross-attention layer to more effectively accommodate image and text features. The dual cross-attention layer is structured into two branches: one for processing image information and the other for processing text information. Given the query features Z and the text features c_t, the output of the text cross-attention Z_t can be defined as:

$$Z_t = Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V, \tag{3}$$

where $Q = ZW_q, K = c_t W_k, V = c_t W_v$ are the query, key and values matrices of the attention operation respectively, and W_q, W_k, W_v are the weight matrices of the trainable linear projection layers.

The image branch incorporates image features with the objective of mitigating the over-fitting issue in the fine-tuned model. Consequently, the image features are adjusted to match the dimensions of the text features, allowing them to be integrated into the image cross-attention layer. Given the query features Z and the image features c_i, the output of the image cross-attention Z_i can be computed as follows:

$$Z_i = Attention(Q, K', V') = Softmax\left(\frac{QK'^T}{\sqrt{d}}\right) \cdot V', \tag{4}$$

where $Q = ZW_q, K' = c_t W'_k, V' = c_t W'_v$ are the query, key and values matrices associated with image features. W'_k, W'_v are the weight matrices, and W_q share the same weight with text cross-attention layer.

Finally, the output of the image branch and the text branch will be merged and processed through the Feed Forward Network (FFN). The final output of dual cross-attention mechanism can be obtained as follows:

$$Z_{final} = FFN(Z_t + Z_i). \tag{5}$$

3.3 Latent-Time-Shift Sampling

The inference stage is an essential part of generating videos. LAMP proposes a first frame conditioned pipeline, which divides video generation into two key parts: the first frame condition and shared noise sampling. Although the first frame conditioned pipeline can determine the main content, subsequent frames are prone to missing multiple objects or producing blurred content. In addition, shared noise sampling struggles to keep the consistency of video frames.

Inspired by LAMP, we propose a *latent-time-shift sampling* strategy, in which the first frame image is encoded into the latent space Z_f and the DDPM process will add the noise to disrupt Z_f. Specifically, we will sample a standard Gaussian latent noise $\epsilon^s \sim \mathcal{N}(0, I)$ as x_{base} and a latent noise sequence $x_{rest} \in (\epsilon^2, ..., \epsilon^l)$ with the same distribution as x_{base}. Basically, x_{base} will be added to the latent noise sequence x_{rest}, which decreases with the length of the video frames, thereby allowing for greater freedom in the video. At the same time, Z_f will be added into each latent noise to maintain the temporal consistency and image style. The final formulation can be written as:

$$z = \alpha \cdot x_{base} + (1 - \alpha) \cdot x_{rest} + \beta \cdot Z_f, \tag{6}$$

where α and β are coefficients. We set $\beta = 0.1$ in our experiment. α is used to control the degree of sharing and decreases with the length of the video f as:

$$\alpha = 0.1 + 0.1 * exp(-0.1 * f). \tag{7}$$

4 Experiments

We utilize stable diffusion v-1.5 with its pretrained weight as the backbone to implement the framework. In our experiment, we generate $f = 16$ frames with $320 * 512$ resolution. In the inference stage, we first employ stable diffusion to generate an image with prompts or select an image as the first frame condition, then predict the subsequent frames. For the training stage, we utilize about 8∼16 videos and sample 16 frames randomly per video. We employ the CLIP image encoder to extract image features. All the sample frames will be resized in a resolution of $320 * 512$. The dual attention layers and the temporal layers will be trained with a learning rate of $3.0 * 10^{-6}$. All the experiments are implemented on a single RTX 3090 GPU.

4.1 Comparison Baselines

We compare our method with three baselines: 1) Show-1 [6], a T2V model trained on WebVid-10M [27] dataset, which can generate videos in various prompts. 2) T2V-Zero [9] introduces a zero-shot video generation approach that operates without any training. 3) LAMP [12] represents a few-shot generation model that is tuned with a few videos to learn the motion pattern from video frames. In our work, we trained **MAIM** on four types of dynamic scenarios, including the blooming of flower, volcano eruptions, horse running, and helicopter flying. We collect training video datasets from the internet. During the inference, we apply the same prompts across all comparison methods, aiming to facilitate both objective and subjective evaluations in our experiments.

4.2 Qualitative Results

We showcase several generation results from different methods in Fig. 4. We note that while LAMP can learn motion patterns from videos, its inability to integrate the reference image features complicates maintaining consistency between the content and textual prompts. Show-1 is capable of generating frames consistent with the given prompt, but it fails to produce the desired style given by prompts and user description. T2V-Zero encounters difficulties in generating detailed and coherent motion patterns, potentially attributable to an insufficiency in learning motion information from the training videos. In contrast, our method utilizes image features from the reference images and combines them with text features. Leading to generate videos with complex background and multiple objects. As shown in Fig. 4, MAIM can accurately maintain multiple objects in complex backgrounds, e.g. the four helicopters, and satisfactorily handle the video generation with different image styles, e.g. the oil painting. In the following, we further utilize **objective metrics** and **user study** to evaluate the performance comprehensively. More visual results are demonstrated in **Appendix**.

Objective Metrics. Following the protocol utilized in [11,12]. We apply Alignment, Consistency, and Diversity to get the quantitative comparison. Forty videos exhibiting four different motion patterns have been selected for comparison. To calculate alignment, we compute the average of the CLIP image embeddings across all frames within a video, and subsequently calculate the cosine similarity between the mean image embedding and the CLIP text embedding extracted from the provided text prompt. Consistency is assessed by computing the average cosine distance across all video frames. For diversity, we refer to LAMP and represent each video with the average CLIP image embedding, and compute the cosine distance between the average embedding and each video frame embedding. Table 1 shows that our method exhibits superior performance compared to other methods.

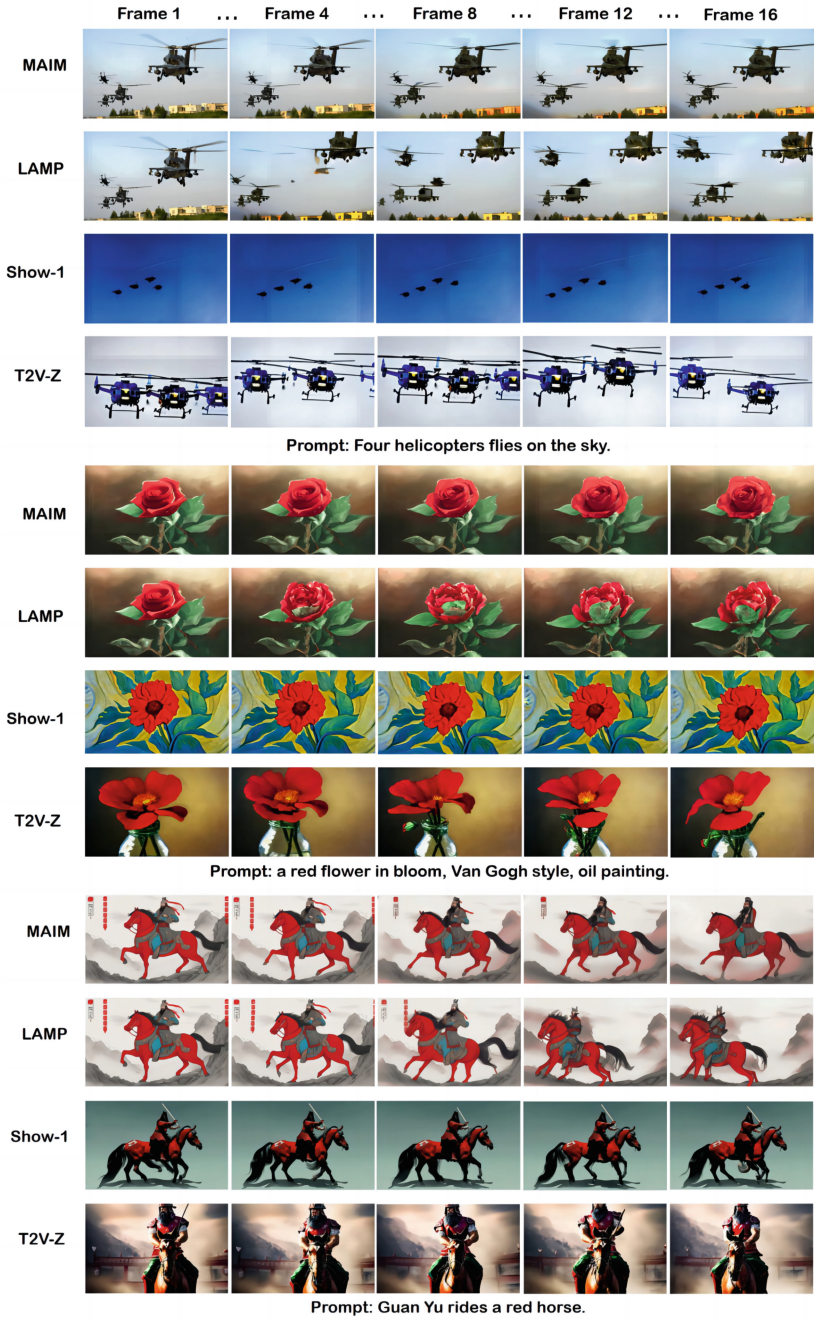


Fig. 4. The qualitative comparison between **MAIM**and three baselines. We generate videos encompassing three distinct styles and types of motion. Zoom in for best view.

Table 1. The quantitative comparison with three baselines.

Method	Alignment ↑	Consistency ↑	Diversity ↓
LAMP	31.3547	97.4123	**71.6535**
Show-1	29.4571	97.3422	75.6172
T2V-Z	26.9424	91.4713	73.0136
MAIM(Ours)	**32.1347**	**98.3125**	71.8813

User Study. To further subjectively evaluate the quality of generated contents, we conduct a questionnaire survey upon human observers as done in [10]. Fifty participants are instructed to rate the visual quality, textual alignment, and temporal consistency of the videos on a scale from 1 to 5. We sum up the rate scores for three items separately over all the generated videos, the results of which are in Table 2 for comparison methods. Although our method may not perform as well as large-scale models (e.g., Show-1) for certain prompts, most participants prefer our generated videos and consider them better than other baselines among all the three measurements.

Table 2. The user study result.

Method	Visual Quality ↑	Alignment ↑	Consistency ↑
MAIM(Ours)	**4.32**	**4.42**	4.15
LAMP	4.06	3.91	3.79
Show-1	3.82	3.31	**4.18**
T2V-Z	3.32	3.16	3.27

4.3 Ablation Study

We construct an ablation study to verify the efficacy of two modules, *dual cross-attention layer* and *latent-time-shift sampling*, in our proposed method. Figure 5 shows the result of the ablation study. We note that *without dual cross-attention layer*, significant content discrepancy among frames in the sequence emerge. E.g., one of the two helicopters become blurred or disappear in subsequent frames, and the dynamic blooming of one flower could not manifest itself at all. This deficiency makes it difficult to maintain complex backgrounds or multiple objects in the generative frames. Without the application of *latent-time-shift sampling*, the consistency across subsequent frames would be compromised. E.g., the detail of helicopters will be lost and the bud is missing. Our sampling method leverages the first frame latent, injecting it into each original latent noise to facilitate to maintain object detail. As shown in Fig. 5, with those two modules, our method is capable of generating video frames with temporal consistency and content stability. Furthermore, we generate around 40 videos with a distinct prompt per

video, and assess the video content quality with two objective metrics, alignment and consistency. As demonstrated in Table 3, the absence of these two components results in the generative videos underperforming in comparison to MAIM with respect to textual alignment and frame consistency. These results confirm the significant contributions of each modules to the overall efficacy of **MAIM**.

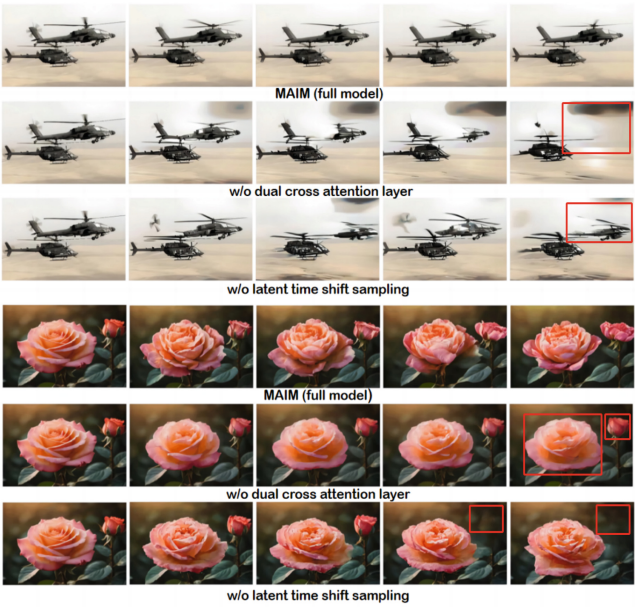


Fig. 5. Visual ablation results upon dual cross-attention layer and latent-time-shift sampling. **The first prompt**: Two helicopters fly on the desert. **The second prompt**: A pink rose and a bud are in bloom. (Color figure online)

Table 3. The ablation results with two objective metrics.

Component	Alignment ↑	Consistency ↑
MAIM(full model)	**32.1762**	**98.1547**
w/o dual cross-attention layer	30.8942	97.2531
w/o latent-time-shift sampling	30.3512	97.1045

5 Conclusion

In this paper, we introduce a new framework for few-shot Text-to-Video generation. We utilize CLIP to extract the image features and decouple image-text features in *dual cross-attention layer* module. To improve the consistency and stability of video generation, we propose *latent-time-shift sampling* method. Compared to other methods, our framework enhances the quality of generative videos in multiple objects or complex background across various image styles.

Acknowledgement. This work was partially supported by the Science and Technology Development Fund (FDCT) of Macau under Grant No. 0071/2022/A, the Wuyi University-Hong Kong-Macau Joint Funding Scheme (2022WGALH17, 2021WGALH18).

A Appendix

See Figs. 6 and 7.

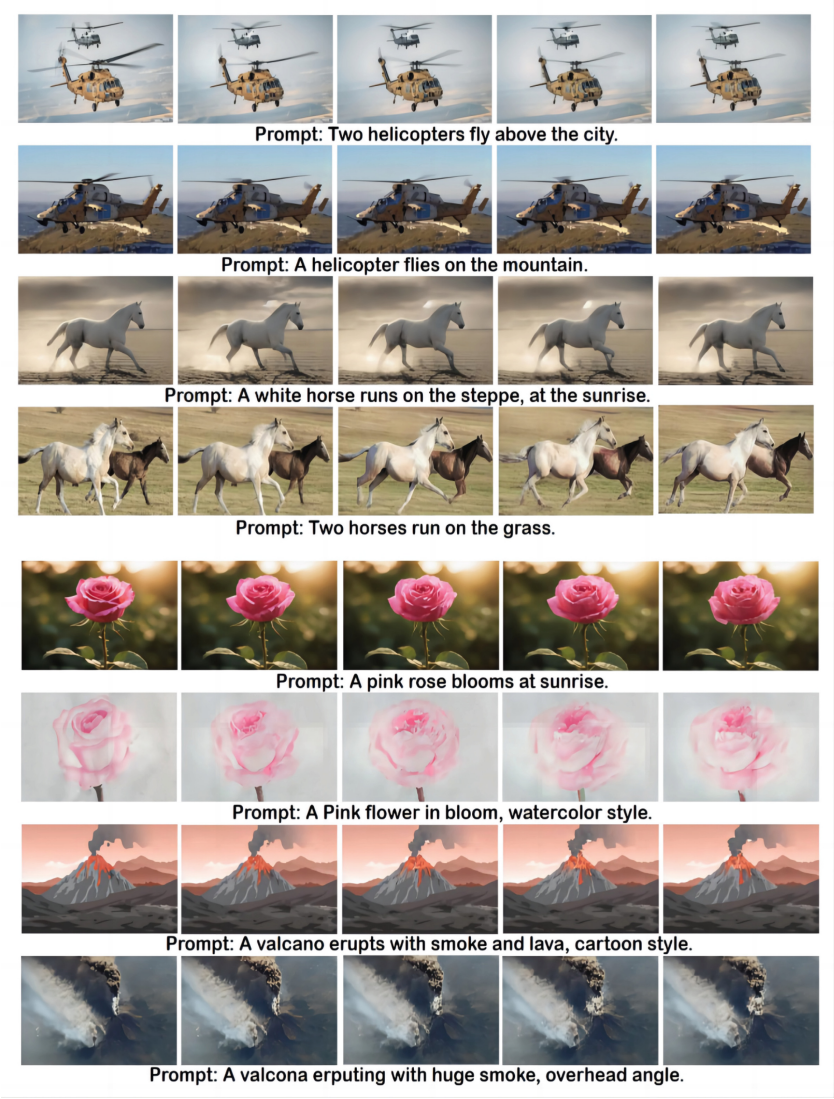


Fig. 6. More visual results generated by MAIM

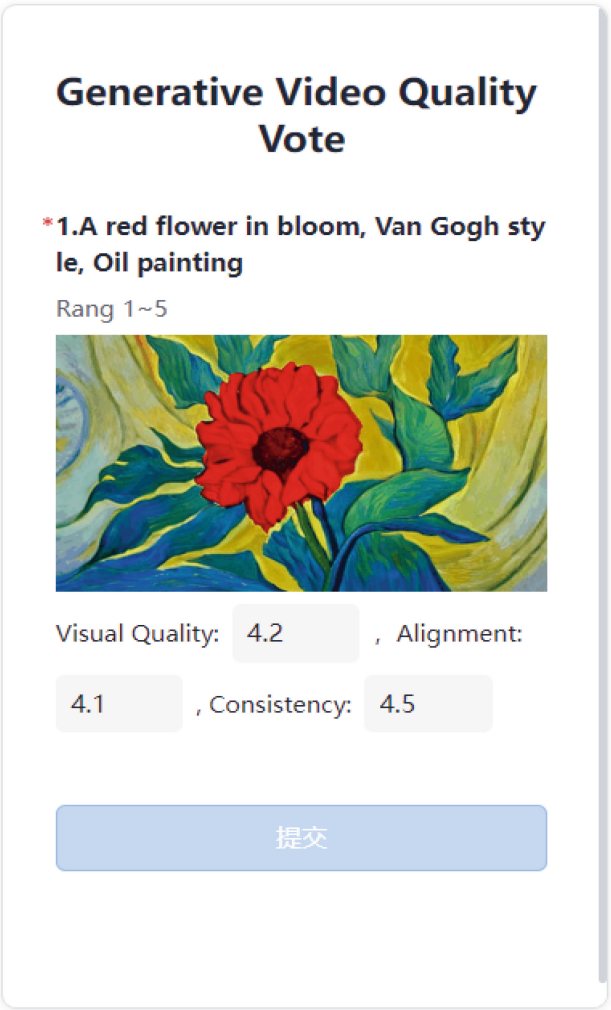


Fig. 7. User study format

References

1. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3836–3847 (2023)
2. Ye, H., Zhang, J., Liu, S., Han, X., Yang, W.: Text compatible image prompt adapter for text-to-image diffusion models. IP-adapter (2023)
3. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022)

4. Blattmann, A., et al.: Align your latents: high-resolution video synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22563–22575 (2023)
5. Guo, Y., et al.: AnimateDiff: animate your personalized text-to-image diffusion models without specific tuning. arXiv preprint arXiv:2307.04725 (2023)
6. Zhang, D.J., et al.: Show-1: marrying pixel and latent diffusion models for text-to-video generation. arXiv preprint arXiv:2309.15818 2023
7. Hong, W., Ding, M., Zheng, W., Liu, X., Tang, J.: CogVideo: large-scale pretraining for text-to-video generation via transformers. arXiv preprint arXiv:2205.15868 (2022)
8. Esser, P., Chiu, J., Atighehchian, P., Granskog, J., Germanidis, A.: Structure and content-guided video synthesis with diffusion models. arXiv preprint arXiv:2302.03011 (2023)
9. Khachatryan, L., et al.: Text2Video-Zero: text-to-image diffusion models are Zero-Shot video generators. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 15954–15964 (2023)
10. Huang, H., Feng, Y., Shi, C., Xu, L., Yu, J., Yang, S.: Free-bloom: zero-shot text-to-video generator with LLM director and LDM animator. In: Conference on Neural Information Processing Systems (2023)
11. Wu, J.Z., et al.: Tune-a-video: one-shot tuning of image diffusion models for text-to-video generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7623–7633 (2023)
12. Wu, R., Chen, L., Yang, T., Guo, C., Li, C., Zhang, X.: LAMP: learn a motion pattern by few-shot tuning a text-to-image diffusion model. arXiv preprint arXiv:2310.10769 (2023)
13. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763. PMLR (2021)
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol. 33, pp. 6840–6851 (2020)
15. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12873–12883 (2021)
16. Zhang, H., et al.: StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5907–5915 (2017)
17. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114 (2013)
18. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
19. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502 (2020)
20. Nichol, A.: GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
21. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with CLIP latents. arXiv preprint arXiv:2204.06125 (2022)
22. Saharia, C., et al.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems, vol. 35, pp. 36479–36494 (2022)

23. Gu, S., et al.: Vector quantized diffusion model for text-to-image synthesis. In: CVPR, pp. 10696–10706 (2022)
24. Ho, J., et al.: Imagen video: high definition video generation with diffusion models. arXiv preprint arXiv:2210.02303 (2022)
25. Singer, U., et al.: Make-a-video: text-to-video generation without text-video data. arXiv preprint arXiv:2209.14792 (2022)
26. Zhou, D., Wang, W., Yan, H., Lv, W., Zhu, Y., Feng, J.: MagicVideo: efficient video generation with latent diffusion models. arXiv preprint arXiv:2211.11018 (2022)
27. Bain, M., Nagrani, A., Varol, G., Zisserman, A.: Frozen in time: a joint video and image encoder for end-to-end retrieval. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1728–1738 (2021)
28. Hong, S., Seo, J., Hong, S., Shin, H., Kim, S.: Large language models are frame-level directors for zero-shot text-to-video generation. arXiv preprint arXiv:2305.14330 (2023)
29. Liu, S., Zhang, Y., Li, W., Lin, Z., Jia, J.: Video-P2P: video editing with cross-attention control. arXiv preprint arXiv:2303.04761 (2023)
30. Qi, C.: FateZero: fusing attentions for zero-shot text-based video editing. arXiv preprint arXiv:2303.09535 (2023)
31. Mikko, K., et al.: Multilayer networks. J. Complex Netw. **2**(3), 203–271 (2014)

A Framework for Image Synthesis Using Supervised Contrastive Learning

Yibin Liu, Jianyu Zhang, Li Zhang, Shijian Li(✉), and Gang Pan

Zhejiang University, Hangzhou, China
{yibinliu,jianyu.zhang,zhangli85,shijianli,gpan}@zju.edu.cn

Abstract. Text-to-image (T2I) generation aims at producing realistic images corresponding to text descriptions. Generative Adversarial Network (GAN) has proven to be successful in this task. Typical T2I GANs are 2-phase methods that first pre-train an inter-modal representation from aligned image-text pairs and then use GAN to train image generator on that basis. However, such representation ignores the inner-modal semantic correspondence, e.g. the images with same label. The semantic label in priory describes the inherent distribution pattern with underlying cross-image relationships, which is supplement to the text description for understanding the full characteristics of image. In this paper, we propose a framework leveraging both inter- and inner-modal correspondence by label guided supervised contrastive learning. We extend the T2I GANs to two parameter-sharing contrast branches in both pre-training and generation phases. This integration effectively clusters the semantically similar image-text pair representations, thereby fostering the generation of higher-quality images. We demonstrate our framework on four novel T2I GANs by both single-object dataset CUB and multi-object dataset COCO, achieving significant improvements in the Inception Score (IS) and Fréchet Inception Distance (FID) metrics of image generation evaluation. Notably, on more complex multi-object COCO, our framework improves FID by 30.1%, 27.3%, 16.2% and 17.1% for AttnGAN, DM-GAN, SSA-GAN and GALIP, respectively. We also validate our superiority by comparing with other label guided T2I GANs. The results affirm the effectiveness and competitiveness of our approach in advancing the state-of-the-art GAN for T2I generation.

Keywords: Text-to-image generation · GAN · Contrastive Learning

1 Introduction

Text-to-image (T2I) generation targets on generating realistic images that match the corresponding text description. This captivating task has gained widespread attention and popularity owing to its vast creative potentials in art generation, image manipulation, virtual reality and computer-aided design.

Y. Liu and J. Zhang—Equal Contribution.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 162–176, 2025.
https://doi.org/10.1007/978-3-031-78172-8_11

T2I generation methods based on Generative Adversarial Network (GAN) [3] have shown promising results. The typical approach can be decomposed the pre-training phase and GAN phase. They first pre-train the image and text features into a joint representation space, which provides effective understanding of the relationship between text descriptions and visual contents, and then use noval GAN to training the image generator on basis of joint representation. Since the introduction of notable AttnGAN [25], many subsequent works have utilized the Deep Attentional Multimodal Similarity Model (DAMSM) which employs contrastive learning to pull the paired image and text representations close while pushing away the unpaired ones. Consequently, DAMSM improve the consistency between image and text representations, resulting in effective downstream generation [9,14,25,32]. Despite contrasting on the inter-modal text-image pair, each image sample may have specific category of similar samples that being ignored or pushed away, resulting in scrapping the underlying inner-modal distribution. Moreover, a brief textual description is usually insufficient to describe all the characteristics of an image. UniCL [26] proposes a unified contrastive loss in image-text-label space to leverage label information during representation learning. However, UniCL does not consider the rareness of samples with the same label in a batch, and is only applicable to single-label datasets.

Taking the inner-modal semantic into consideration, we introduce supervised contrastive learning into T2I GAN by referring to the categorical information of images, which enhances both the representation encoders and GAN generator, thereby improving the quality of image generation. For single-object image generation, we incorporate single-label supervised contrastive learning [6]. During the pre-training phase, our proposed supervised contrastive loss leverages additional image labels to group the representations for image and text of the same class while distinguishing images of different classes. During the GAN phase, we also employ the supervised contrastive loss to simultaneously increase the synthetic images' similarities of same class and the matching degree to their text pair. For multi-object image generation, we leverage same approach on single-object scenario by changing the supervised contrastive loss to multi-label case [12]. We evaluate our method on datasets CUB [24] and COCO [11]. By comparing to four base models: AttnGAN [25], DM-GAN [32] SSA-GAN [9] and GALIP [21], our experiments show that our method is capable of improving the quality of generated images measured by common metrics: the Inception Score (IS) [18] and Fréchet Inception Distance (FID) [23].

The contributions of our work can be summarized as follows:

- We incorporate supervised contrastive learning to T2I generation which encourages the inherent data distribution patterns delineated by semantic labels, thereby enhancing the generation of coherent and faithful images.
- Our framework employs two symmetric parameter-sharing branches in the pre-training and GAN phase of T2I generation, which is compatible for single- and multi-object contrastive learning by corresponding loss. Such extension converges image representations carrying same semantics within proximity in

the pre-training phase, which enables the GAN generator to glean insights from a broader spectrum of related data instances.
– Our framework can improve famous T2I GANs' generation quality on both single-object CUB and multi-object COCO dataset. Most notably, on more complex COCO dataset, our framework improves the FID of AttnGAN, DM-GAN, SSA-GAN and GALIP by 30.1%, 27.3%, 16.2% and 17.1%, respectively. We also demonstrate the superiority of our framework comparing with other label guidance options.

2 Related Work

2.1 Contrastive Learning

Contrastive learning is a self-supervised method which has been successful in representation learning. It plays a crucial role in serving computer vision tasks and extends influence to other research field like natural language processing. Contrastive learning follows the intuition that similar data samples should be closer in the representation space, while dissimilar samples should be far apart. Typical contrastive learning setting SimCLR [1] augments image into two randomly warped views and extracts their representations through twin encoders. The two branches of representation are then projected to same feature space to apply contrastive loss [13], where the paired view of image is considered as positive sample and vice verca. Other variants of contrastive learning mainly differ in the formulation of negative samples [5], the asymmetric design of twin encoders [4], or contrastive loss definition [29]. All these methods have either comparable results or exceed supervised methods on many representation learning benchmarks [2]. In addition to construct the positive and negative samples by self supervision, researchers [6,12] also utilize image classification labels to formulate single- and multi-label contrastive loss, the former achieves high accuracy in image classification while the latter succeeds in visual reasoning. Contrastive learning has also been explored to bridge the modality gap and create unified representation for multi-modal pre-training. Trained by fine-curated large scale image text pairs, CLIP [15] has demonstrated great zero-shot capability for dozens of visual and image-text downstream tasks.

These contrastive learning progresses proves the feasibility of aligning different feature views at low annotation cost. We adopt the intuition that any data representation can be improved by referencing similar semantic concepts from both inter- and inner-modal data, therefore our framework designs multiple ways of feature alignment which will be detailed in Sect. 3.

2.2 GAN for Text-to-Image Generation

In recent years, image generation has experienced rapid development starting from the remarkable success of Generative Adversarial Network (GAN) which trains a generative model by adversarial discrimination [9,14,22,25,30–32]. Reed

et al. [16] were the first to employ GAN to generate images from text descriptions. To synthesize higher resolution images, Zhang et al. propose the StackGAN [30] and StackGAN++ [31] employing a multi-generator strategy that first generates a low-resolution image and then finetunes followup generators to produce high resolution realistic images. Many works follow this multi-stage stack structure [14,17,25,28,32] to improve image generation quality. On basis of StackGAN++, AttnGAN [25] introduced attention mechanism to refine the process of generating images from fine-grained textual descriptions at different stages of image generation. In addition, AttnGAN proposed the Deep Attentional Multi-modal Similarity Model (DAMSM) to improve multi-granular consistency between image and text. DM-GAN [32] proposed dynamic memory to store the intermediate generated images and retrieve the most relevant textual information with gated attention to update the image representation accordingly.

Although the multi-stage GAN is designate for high-resolution progressive image generation, its training complexity grows as the stage stacking. To overcome this, DF-GAN [22] proposed single-stage generation, whose generator uses a series of UPBlock specially designed for high resolution feature upsampling. DF-GAN further used Matching-Aware Gradient Penalty and hinge loss to train the UPBlocks. Followup SSA-GAN [9] used a Semantic Spatial Aware Convolution Network (SSACN) block to predict text aware mask maps based on the current generated image features, which facilitates the fusion and consistency between image and text. These conventionally designed single-stage methods greatly reduce the complexity of T2I generation, meanwhile others seek for utilizing famous visual-language pre-training techniques to bridge the inter-modal gap. GALIP [21] directly integrates CLIP [15] to harness the well-aligned image-text representation and extend GAN's ability to synthesize complex images. Hui et al. [27] propose a framework leveraging contrastive learning to enhance the consistency between caption generated images and the originals. All these T2I GANs focus on the inter-modal image text alignment without considering inner-modal association, which in some extent leads to flaws in the generation results. Our framework instead encourages both inter- and inner-modal association.

3 Method

In this section, we introduce a simple effective framework which integrates supervised contrastive learning to leverage the inner-modal association, thereby enhancing the generation quality of T2I GANs. Like novel contrastive learning approach, we adopt the dual tower structure and create two symmetric branches of contrast opponents for both pre-training and GAN phases. In pre-training phase, the supervised contrastive learning encourages the representation coherency for image-text pairs sharing same semantics. In favor of the coherent representation, in the GAN phase, the supervised contrastive learning establishes additional guidance for the semantic consistency of the generated images. We detail our framework adaptation and enhanced T2I GAN learning objectives for the two phases in the following respective sections.

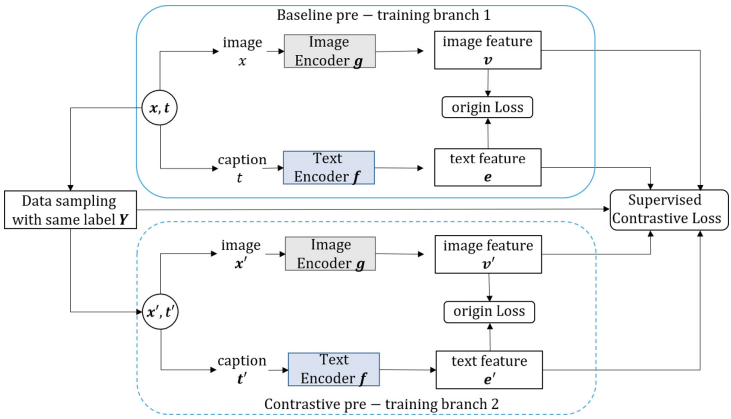


Fig. 1. Pre-training phase. Our data sampling strategy initiates two contrast branches with shared parameters to separately encode the image-text pairs of same label. The original Loss is consistent to the method our framework applied on. The supervised contrastive loss works on quadruple of image and text representations from both branches.

3.1 Supervised Contrastive Learning for Pre-training

Typical T2I GANs pre-train the image and text encoders by maximizing the paired image-text representation similarity and the unpaired dissimilarity. To enhance this learning process, we extend the pre-training by supervised contrastive learning on the image-text pair with shared label. The extension has three components shown in Fig. 1.

Data Sampling Strategy. At each training step, we randomly sample a batch of N examples which consist of N captions $\boldsymbol{t}$, corresponding images $\boldsymbol{x}$ and label set $\boldsymbol{Y}$. To construct contrastive pair, we ensure that each sample has reference example with the same labels: for each sample (t_i, x_i, Y_i), we select a sample (t_i', x_i', Y_i') as its pair where $Y_i \cap Y_i' \neq \emptyset$.

Image Encoder g And Text Encoder f. In pre-training phase, the encoder extracted representations usually have multi-granular features to encourage the deep fusion, e.g., the global/local views of image, and the sentence/word level of text. Our methods do not change the functionalities but extend them by applying shared image and text encoders $\boldsymbol{g}, \boldsymbol{f}$ to extract contrastive pair image representations $\boldsymbol{v} = \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{v'} = \boldsymbol{g}(\boldsymbol{x'})$ and text representations $\boldsymbol{e} = \boldsymbol{f}(\boldsymbol{t}), \boldsymbol{e'} = \boldsymbol{f}(\boldsymbol{t'})$. Our framework is indifferent for the type of encoders, where we keep them consistent to the baseline methods our framework applied to. Specifically, for AttnGAN [25], DM-GAN [32] and SSA-GAN [9], we use Inception-v3 [20] as image encoder $\boldsymbol{g}$ and Bi-LSTM [19] as text encoder $\boldsymbol{f}$. For GALIP [21], we use transformer-based CLIP image and text encoders. The weights of the text encoder and image encoder are frozen during the training phase of the GAN.

Learning Objective. With the data sampling strategy, we define the objective for training. For image-text matching using Inception-v3 and Bi-LSTM, we consider (t_i, x_i) and (t'_i, x'_i) as positive image-text pairs to calculate DAMSM loss same as AttnGAN [25]. As for CLIP encoder, we use symmetric cross entropy loss [15]. To apply supervised contrastive loss, we formulate positive pairs from sampling strategy for image-image, image-text and text-text associations. Specifically, (t_i, t'_i), (t_i, t_j) and (t_i, t'_j) are considered as positive text-text pairs where $Y_i \cap Y_j \neq \emptyset$. It is worth noting that in single-object dataset CUB, each corresponding image-text sample only has one label, while in complex multi-object dataset COCO, it has multiple labels. Therefore, we use different supervised contrastive loss functions to deal with different label sharing.

For **one label** scenario, we treat sample pairs with the same label as positive pairs and apply single-label supervised contrastive loss. Given a random batch of N instances, we pick $2N$ instances after data sampling stategy where each instance is guaranteed to have at least one same label in other instances. In order to facilitate the calculation, we concatenate the sampled instances with the original ones to obtain the image representation $\widetilde{\boldsymbol{v}} = \{\boldsymbol{v}, \boldsymbol{v'}\}$, text representation $\widetilde{\boldsymbol{e}} = \{\boldsymbol{e}, \boldsymbol{e'}\}$ and labels $\widetilde{\boldsymbol{Y}} = \{\boldsymbol{Y}, \boldsymbol{Y'}\}$ at this step. Let $sim(a, b) = a^T b/(||a|| \cdot ||b||)$ denote the cosine similarity between a and b. For a certain representation u_i and its relative batch of representations $\boldsymbol{w}$, the supervised contrastive loss function is calculated as

$$\mathcal{L}^{sup}(u_i, \boldsymbol{w}) = \frac{-1}{|P_s(i)|} \sum_{p \in P_s(i)} log \frac{exp(sim(u_i, w_p)/\tau)}{\sum_{j \neq i}^{2N} exp(sim(u_i, w_j)/\tau)} \tag{1}$$

where $P_s(i) = \{p \in \{1, ..., 2N\} : \widetilde{Y_p} = \widetilde{Y_i}\}$ is the set of indices of all positives in the batch distinct from i, $|P_s(i)|$ is the cardinality of $P_s(i)$ and τ denotes the temperature parameter. We can specifically compute supervised contrastive losses for image-image $\mathcal{L}^{sup}_{img}$, text-text $\mathcal{L}^{sup}_{txt}$ and image-text $\mathcal{L}^{sup}_{i2t}$ as follows:

$$\mathcal{L}^{sup}_{img} = \sum_{i=1}^{2N} \mathcal{L}^{sup}(\widetilde{v_i}, \widetilde{\boldsymbol{v}}) \tag{2}$$

$$\mathcal{L}^{sup}_{txt} = \sum_{i=1}^{2N} \mathcal{L}^{sup}(\widetilde{e_i}, \widetilde{\boldsymbol{e}}) \tag{3}$$

$$\mathcal{L}^{sup}_{i2t} = \sum_{i=1}^{2N} \mathcal{L}^{sup}(\widetilde{e_i}, \widetilde{\boldsymbol{v}}) + \sum_{i=1}^{2N} \mathcal{L}^{sup}(\widetilde{v_i}, \widetilde{\boldsymbol{e}}) \tag{4}$$

Similarly, for **multi-label** scenarios, we consider instances that have one or more common labels as positive pair. We employ multi-label supervised contrastive loss, which replaces $P_s(i)$ with $P_m(i) = \{p \in \{1, ..., 2N\} : \widetilde{Y_p} \cap \widetilde{Y_i} \neq \emptyset\}$ in the calculation process while keeping all other calculation the same as in the single-label contrastive loss.

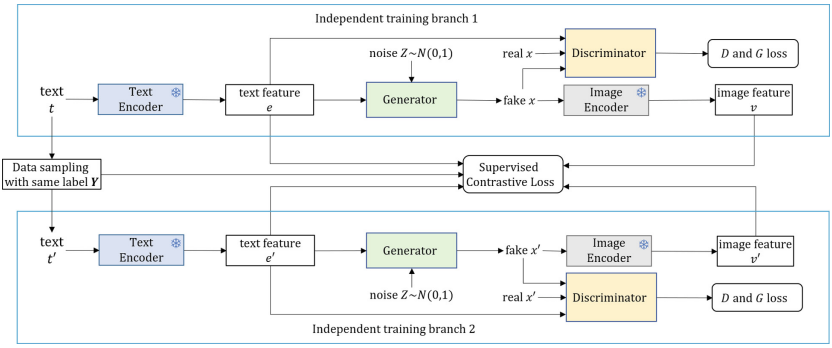


Fig. 2. GAN training phase. Same as pre-training phase, we use two parameter-sharing T2I GAN branches to contrast the text-image pairs sharing same label. The supervised contrastive loss is performed on quadruple of text and generated fake image representations from two branches. In this phase, the pre-trained encoders are inference-only.

The final objective function for the pre-training phase is a co-op of origin loss and supervised contrastive loss

$$\mathcal{L}_{pre} = \mathcal{L}_{orgin} + \lambda_1(\mathcal{L}_{img}^{sup} + \mathcal{L}_{txt}^{sup} + \mathcal{L}_{i2t}^{sup}) \tag{5}$$

where λ_1 is the weight of supervised contrastive loss. Depending on the baseline GAN methods, $\mathcal{L}_{orgin}$ can either be DAMSM or symmetric cross entropy loss.

3.2 Supervised Contrastive Learning for GAN

Intuitively, shared labels reflect common visual semantics within the images. In captioning datasets, the brief text annotation typically use concise descriptions to depict partial aspect of images. Therefore, during generator training, we provide instances sharing same label to encourage the generator to refer to the similar instances. Our generator training framework is illustrated in Fig. 2.

Data Sampling Strategy. Same as the pre-training phase, we sample a batch of images $\boldsymbol{x}$ and $\boldsymbol{x'}$, text captions $\boldsymbol{t}$ and $\boldsymbol{t'}$, labels $\boldsymbol{Y}$ and $\boldsymbol{Y'}$. The captions are extracted to text representations $\boldsymbol{e}$ and $\boldsymbol{e'}$ by pre-trained text encoder $\boldsymbol{f}$.

GAN Adaptation. As discussed in Sect. 2.2, the mainstream T2I GAN methods are based on two types: the multi-stage StackGAN series [31] and the one-stage DFGAN [22]. Our framework can be applicable to both types. Given the ground-truth real image $\boldsymbol{x}$, the generator G utilizes text representations $(\boldsymbol{e}, \boldsymbol{e'})$ and noise z to generate fake images $(\boldsymbol{x_f}, \boldsymbol{x'_f})$ in two branches. Subsequently, the discriminator calculates the generator losses $(\mathcal{L}_G^o, \mathcal{L}_G^{o'})$ and discriminator losses $(\mathcal{L}_D^o, \mathcal{L}_D^{o'})$ for two branches from $(\boldsymbol{x}, \boldsymbol{e}, \boldsymbol{x_f})$ and $(\boldsymbol{x'}, \boldsymbol{e'}, \boldsymbol{x'_f})$, respectively. Meanwhile, the generated images from both branches are encoded by an image encoder

and obtains fake image representations $(\boldsymbol{v_f}, \boldsymbol{v'_f})$. These representations are then paired with $(\boldsymbol{e}, \boldsymbol{e'})$ to calculate supervised contrastive loss.

Learning Objective. In our framework, the objective function for discriminator loss during the training process is identical to the GAN baselines in both branches, and the overall discriminator loss $\mathcal{L}_D$ is the sum of loss from two branches. As for the generator loss $\mathcal{L}_G$, one-stage GAN typically use conditional generation loss [10,22] while multi-stage GAN often incorporate additional non-conditional generation loss [31]. Our method does not vary the usage of baseline generator losses but adding extra supervised contrastive losses for image-to-image and image-text pairs.

Similar to pre-training phase, for sampled batch, we first concatenate the generated fake image representation $\overline{\boldsymbol{v}} = \{\boldsymbol{v_f}, \boldsymbol{v'_f}\}$, the corresponding text representations $\widetilde{\boldsymbol{e}} = \{\boldsymbol{e}, \boldsymbol{e'}\}$ and the labels $\widetilde{\boldsymbol{Y}} = \{\boldsymbol{Y}, \boldsymbol{Y'}\}$. The discriminator and generator loss function are then computed as follows:

$$\mathcal{L}_D = \mathcal{L}_D^o + \mathcal{L}_D^{o'} \tag{6}$$

$$\mathcal{L}_G = \mathcal{L}_G^o + \mathcal{L}_G^{o'} + \lambda_2(\mathcal{L}_{img}^{sup} + \mathcal{L}_{i2t}^{sup}) \tag{7}$$

where

$$\mathcal{L}_{img}^{sup} = \sum_{i=1}^{2N} \mathcal{L}^{sup}(\overline{v_i}, \overline{\boldsymbol{v}}) \tag{8}$$

$$\mathcal{L}_{i2t}^{sup} = \sum_{i=1}^{2N} \mathcal{L}^{sup}(\widetilde{e_i}, \overline{\boldsymbol{v}}) + \sum_{i=1}^{2N} \mathcal{L}^{sup}(\overline{v_i}, \widetilde{\boldsymbol{e}}) \tag{9}$$

and λ_2 is the weight of supervised contrastive loss.

4 Experiments

We choose novel multi-stage (AttnGAN, DM-GAN) and one-stage (SSA-GAN, GALIP) GANs to validate the superiority and universality of our framework on T2I generation for both single-object CUB [24] and multi-object COCO [11] datasets. We also conduct extensive ablations to assess the effectiveness of each component our framework proposes.

Evaluation Metric. We follow the baselines' evaluation protocol on the CUB and COCO datasets, which uses Inception Score (IS) [18] and Fréchet Inception Distance (FID) [23] as quantitative evaluation metrics. After training completion, we generate 30,000 images in resolution 256×256 on the test set and compute IS and FID scores. Several previous works [8,22] have pointed out that IS can not provide useful guidance to evaluate the quality of the synthetic images on dataset COCO, thus we only evaluate IS on CUB dataset. Since GALIP was not evaluated on IS, we only compared with GALIP on FID.

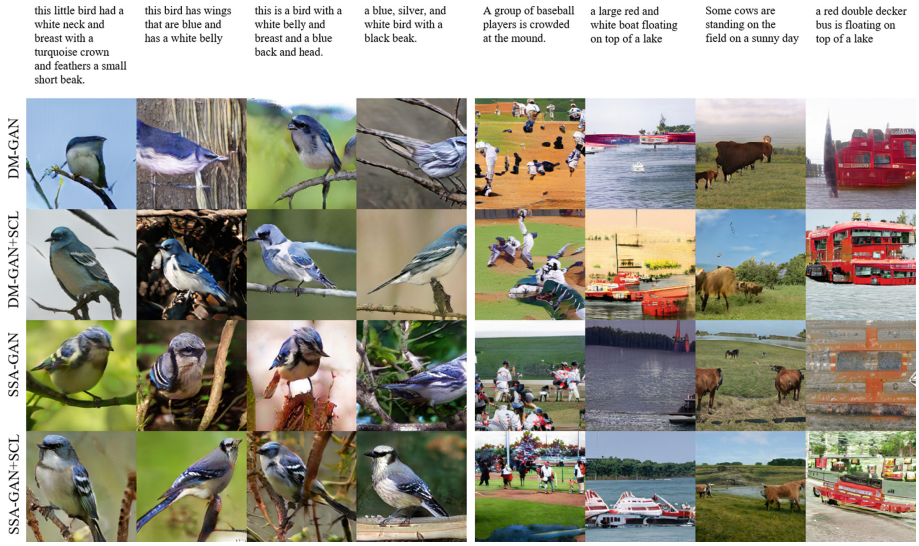


Fig. 3. Qualitative comparison on CUB and COCO datasets for DM-GAN and SSA-GAN baselines w/o the utilization of our framework (denoted as "+SCL"). The input text descriptions are given in the first row and the corresponding generated images from different methods are shown in the same column. The left 4 columns are from CUB, and right 4 columns from COCO.

Implementation Details. We apply our framework to four novel baselines (AttnGAN, DM-GAN, SSA-GAN and GALIP) on both CUB and COCO datasets. During pre-training phase, we set λ_1 to 0.5 for CUB and 0.05 for COCO. For GAN phase, we set λ_2 of the four baselines to 5, 2.5, 0.2, 0.15 for CUB and 2.5, 2.5, 0.1, 0.15 for COCO. The training epochs of the four baselines are 600, 800, 600, 2000 for CUB and 120, 200, 120, 2000 for COCO. Our training uses 1, 1, 3, 3 NVIDIA GeForce RTX 3090 GPU respectively for the four baselines.

4.1 Quantitative Results

The four baselines and our enhancement results are reported in Table 1. On single-object CUB dataset, our framework is able to improve the IS of AttnGAN by 5.7%, DM-GAN by 6.5%, and SSA-GAN by 1.4%. These results demonstrate that our framework effectively improves the clarity and diversity of generated images. Moreover, our framework improves the FID of AttnGAN by 25.6%, DM-GAN by 6.3%, SSA-GAN by 9.5% and GALIP by 1.8%. On more challenging multi-object COCO dataset, our framework is able to significantly improve the FID of all baselines. Specifically, we improves AttnGAN, DM-GAN, SSA-GAN and GALIP by 30.1%, 27.3%, 16.2% and 17.1% respectively. These results indicate that semantic relationship modeling is crucial for enhancing the T2I GAN generation quality, and the more complex scenario benefits more from it.

Table 1. Performance of IS and FID of AttnGAN, DM-GAN, SSA-GAN and these models with our framework increment on the CUB and COCO test set. ↑ denotes higher values indicate better quality. ↓ denotes lower values indicate better quality. * denotes results obtained from publicly released pre-trained models by the authors. "+SCL" represents the model trained by our framework. **Bold** for better performance.

Methods	CUB		COCO
	IS↑	FID↓	FID↓
AttnGAN*	$4.36 \pm .03$	23.98	33.10
AttnGAN+SCL	$\mathbf{4.61 \pm .06}$	**17.83**	**23.14**
DM-GAN*	$4.65 \pm .05$	15.31	26.56
DM-GAN+SCL	$\mathbf{4.95 \pm .05}$	**14.35**	**19.32**
SSA-GAN*	$5.07 \pm .08$	15.69	19.37
SSA-GAN+SCL	$\mathbf{5.14 \pm .09}$	**14.20**	**16.24**
GALIP	-	10.08	5.85
GALIP+SCL	-	**9.90**	**4.85**

4.2 Visual Quality

In this section, we further compare the visual quality of generated images by a subset of CUB and COCO datasets for DM-GAN, SSA-GAN baselines before and after applying our framework, which are shown in Fig. 3.

For the CUB dataset, we randomly select text-generated images belonging to the "Tree Swallow" category for comparison. In the first and second column, the images generated by DM-GAN exhibit severe error in producing bird head, while DM-GAN with supervised contrastive learning generates natural bird images. SSA-GAN on the other hand can generate natural bird images, but the generated bird images do not always match the descriptions or the desired bird species. For example, the bird generated in the 1st column exhibits yellow and green wings, and the bird in the 3rd column had red tails, which are not mentioned in the text description and do not align with the characteristics of Tree Swallows. On the contrary, SSA-GAN enhanced by our framework can produce birds that match the text description specifying blue-black-white wings, and is consistent with the features of Tree Swallows. In addition, the images generated by our framework exhibit strong similarity for same species, which further confirms the validity of supervised contrastive learning.

Generating realistic and textually coherent images that align with the descriptions is more challenging in the COCO dataset. However, our framework outperforms the baseline in terms of generating higher quality and more textually consistent images. For example, in 6th column, both DM-GAN and SSA-GAN failed to generate a red boat mentioned in the input text, but DM-GAN and SSA-GAN enhanced by our framework successfully generate the desired object. In 8th column, the bus generated by SSA-GAN is orange-yellow which devi-

ates from the "red" description, while SSA-GAN enhanced by our framework successfully produce a red bus matching the description.

4.3 Ablation Study

In both pre-training and GAN phases we incorporate image-image supervised contrastive loss L_{img}^{sup} and image-text L_{i2t}^{sup} supervised contrastive loss. In this section, we verify the effectiveness of *pre*, L_{img}^{sup} and L_{i2t}^{sup} in our framework by conducting extensive ablation study on the CUB and COCO dataset in Table 2.

Table 2. Ablations of AttnGAN baseline. Our pre-trained encoders (*pre*), image-image supervised contrastive loss (L_{img}^{sup}) and image-caption supervised contrastive loss (L_{i2t}^{sup}) are ablated independently.

ID	Components			CUB		COCO
	pre	L_{img}^{sup}	L_{i2t}^{sup}	IS↑	FID↓	FID↓
1	-	-	-	$4.36 \pm .03$	23.98	33.10
2	✓	-	-	$4.41 \pm .05$	20.83	26.90
3	✓	✓	-	$4.53 \pm .04$	**17.42**	24.14
4	✓	-	✓	$4.45 \pm .07$	18.53	25.09
5	✓	✓	✓	**4.61 ± .06**	17.83	**23.14**

We consider the AttnGAN as the baseline (ID 1). When using pre-trained encoders (ID 2), all metrics get improved, which indicates that the encoders with supervised contrastive learning obtain image and text representations with better semantic alignment and consistency (the visualization of representation is given in supplementary material). Building upon *pre*, introducing L_{img}^{sup} (ID 3) and L_{i2t}^{sup} (ID 4) individually also results in improvement for all metrics, which suggests that using L_{img}^{sup} and L_{i2t}^{sup} separately enhances the similarity between image-image and image-text representations with the same label. The usage of L_{img}^{sup} shows better improvement comparing to L_{i2t}^{sup}, indicating that previous work is more lack of the intrinsic image modeling on dataset semantic level. However, when L_{img}^{sup} and L_{i2t}^{sup} are used together (ID 5), both IS of CUB and FID of COCO are improved, but the FID of CUB inferior a little. The reason is that L_{i2t}^{sup} surges impact on facilitating text-image fusion and representation similarity, resulting in the IS improvement. On the other hand, when the encoded text features become more adaptive to the image features with same labels, the diversity of generated images also increases(more deeply constrained by the text descriptions with same label). Consequently, the FID slightly drops as it measures the KL divergence between the real images and generated images [9].

4.4 Comparison to Other Label-Supervised Methods

To our best knowledge, there is no existing approach in this field leveraging labels information as additional guidance like our framework does. To demonstrate

the novelty of our approach, we use two simple settings that commonly used for plug-in label learning as extra baselines. Firstly, we apply UniCL [26] to AttnGAN. On CUB dataset, UniCL can easily be adopted because each image only associates with one label. In order to apply UniCL to the COCO dataset, we replaced its single-label supervised contrastive loss to a multi-label supervised contrastive loss. Secondly, we introduce cross-entropy loss in classification task to AttnGAN. We introduce a pre-trained fully connected network as a image classifier and add the cross-entropy loss to the existing loss and train by multi-task learning. The results are shown in the Table 3. As the UniCL and cross-entropy improving the AttnGAN slightly, our framework demonstrate largest margin of visual enhancement for all metrics, indicating the compatibility of our framework with T2I GAN baselines.

Table 3. AttnGAN baseline comparison of other semantic label integration options including UniCL, cross-entropy and ours.

Methods	CUB		COCO
	IS↑	FID↓	FID↓
AttnGAN*	$4.36 \pm .03$	23.98	33.10
UniCL	$4.39 \pm .02$	19.42	28.67
cross-entropy	$4.34 \pm .05$	21.15	27.30
Ours	$\mathbf{4.61 \pm .06}$	**17.83**	**23.14**

5 Conclusions

In this work, we introduce a novel framework that harness semantic information with supervised contrastive learning to improve T2I GAN. Our framework use the two branch contrast to extend the original method across the pre-training and GAN phases. In pre-training phase, we employ label guided data sampling strategy, where we define positive pair as the images with same label. Driven by supervised contrastive loss on the positive image pairs and their corresponding text, the pre-training encoder elevates the representation similarity of images with same semantic concepts and push away those without. In the GAN phase, we first proceed original GAN for each branch independently and formulate a quadruple including the representations of generated positive image pair and their corresponding texts from two branches. We then employ augmented supervised contrastive loss to the quadruple which, like in pre-training phase, serves to elevate the similarity between images characterized by common semantic, thereby enhancing the image generation quality.

We apply our framework to famous four GAN baselines including AttnGAN, DM-GAN, SSA-GAN and GALIP and conduct experiments on single-object

CUB and multi-object COCO dataset. The results demonstrate that our framework can indifferently improve baselines on both datasets with considerable margin, especially the more complex COCO.

Although we only demonstrate the effectiveness on the datasets with detailed label annotation, our framework can be extended to other image-text pair only datasets by noun extraction from all text as labels, which will be the next step of our research interest. Recently, the advent of data-centric methodologies such as SAM [7] has further curtailed the expenses for semantic label acquisition, subsequently relaxing the prerequisites for implementing our framework. Furthermore, we expect this work to exhibit potential application for diffusion models especially on efficiency improving due to the adaptable nature of our framework. We defer the extension to future research endeavors.

Acknowledgments. This research was supported by STI 2030—Major Projects 2021ZD0200403. The authors like to thank the authors of DM-GAN for providing the details of its implementation and the anonymous reviewers for their review and comments.

References

1. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, pp. 1597–1607. PMLR (2020)
2. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20–25 June 2009, Miami, Florida, USA, pp. 248–255. IEEE Computer Society (2009). https://doi.org/10.1109/CVPR.2009.5206848
3. Goodfellow, I.J., et al.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8–13 December 2014, Montreal, Quebec, Canada, pp. 2672–2680 (2014). https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html
4. Grill, J., et al.: Bootstrap Your own latent - a new approach to self-supervised learning. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, 6–12 December 2020, virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html
5. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.B.: Momentum contrast for unsupervised visual representation learning. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, 13–19 June 2020, pp. 9726–9735. Computer Vision Foundation/IEEE (2020). https://doi.org/10.1109/CVPR42600.2020.00975
6. Khosla, P., et al.: Supervised contrastive learning. In: Advances in Neural Information Processing Systems, vol. 33, pp. 18661–18673 (2020)
7. Kirillov, A., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)

8. Li, W., et al.: Object-driven text-to-image synthesis via adversarial training. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019, pp. 12174–12182. Computer Vision Foundation/IEEE (2019). https://doi.org/10.1109/CVPR.2019.01245. http://openaccess.thecvf.com/content_CVPR_2019/html/Li_Object-Driven_Text-To-Image_Synthesis_via_Adversarial_Training_CVPR_2019_paper.html
9. Liao, W., Hu, K., Yang, M.Y., Rosenhahn, B.: Text to image generation with semantic-spatial aware GAN. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, 18–24 June 2022, pp. 18166–18175. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.01765
10. Lim, J.H., Ye, J.C.: Geometric GAN. CoRR abs/1705.02894 (2017). http://arxiv.org/abs/1705.02894
11. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
12. Małkiński, M., Mańdziuk, J.: Multi-label contrastive learning for abstract visual reasoning. IEEE Trans. Neural Netw. Learn. Syst. (2022)
13. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. CoRR abs/1807.03748 (2018). http://arxiv.org/abs/1807.03748
14. Qiao, T., Zhang, J., Xu, D., Tao, D.: MirrorGAN: learning text-to-image generation by redescription. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1505–1514 (2019)
15. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (2021). http://proceedings.mlr.press/v139/radford21a.html
16. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H.: Generative adversarial text to image synthesis. In: International Conference on Machine Learning, pp. 1060–1069. PMLR (2016)
17. Ruan, S., et al.: DAE-GAN: dynamic aspect-aware GAN for text-to-image synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13960–13969 (2021)
18. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
19. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997). https://doi.org/10.1109/78.650093
20. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 2818–2826. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.308
21. Tao, M., Bao, B., Tang, H., Xu, C.: GALIP: generative adversarial CLIPs for text-to-image synthesis. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, 17–24 June 2023, pp. 14214–14223. IEEE (2023). https://doi.org/10.1109/CVPR52729.2023.01366

22. Tao, M., Tang, H., Wu, F., Jing, X., Bao, B., Xu, C.: DF-GAN: a simple and effective baseline for text-to-image synthesis. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, 18–24 June 2022, pp. 16494–16504. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.01602
23. Unterthiner, T., et al.: Coulomb GANs: provably optimal Nash equilibria via potential fields. arXiv preprint arXiv:1708.08819 (2017)
24. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 dataset. California Institute of Technology (2011)
25. Xu, T., et al.: AttnGAN: fine-grained text to image generation with attentional generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1316–1324 (2018)
26. Yang, J., et al.: Unified contrastive learning in image-text-label space. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, 18–24 June 2022, pp. 19141–19151. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.01857
27. Ye, H., Yang, X., Takac, M., Sunderraman, R., Ji, S.: Improving text-to-image synthesis using contrastive learning. arXiv preprint arXiv:2107.02423 (2021)
28. Yin, G., Liu, B., Sheng, L., Yu, N., Wang, X., Shao, J.: Semantics disentangling for text-to-image generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2327–2336 (2019)
29. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow Twins: self-supervised learning via redundancy reduction. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event. Proceedings of Machine Learning Research, vol. 139, pp. 12310–12320. PMLR (2021). http://proceedings.mlr.press/v139/zbontar21a.html
30. Zhang, H., et al.: StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5907–5915 (2017)
31. Zhang, H., et al.: StackGAN++: realistic image synthesis with stacked generative adversarial networks. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1947–1962 (2018)
32. Zhu, M., Pan, P., Chen, W., Yang, Y.: DM-GAN: dynamic memory generative adversarial networks for text-to-image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5802–5810 (2019)

TMCSpeech: A Chinese TV and Movie Speech Dataset with Character Descriptions and a Character-Based Voice Generation Model

Dong Liu[1], Yueqian Lin[1], Yunfei Xu[2], and Ming Li[1(✉)]

[1] Suzhou Municipal Key Laboratory of Multimodal Intelligent Systems, Data Science Research Center, Duke Kunshan University, Kunshan, China
ming.li369@dukekunshan.edu.cn
[2] Guangdong OPPO Mobile Telecommunications Corp., Ltd., Dongguan, China

Abstract. Recent research on text-guided speech synthesis has sparked considerable interest. This study explores the potential of leveraging publicly available internet video data for speech synthesis and character-based new voice generation. We introduce a multi-modal extraction pipeline for automating the creation of speech synthesis datasets, extracting accurate character speech segments and descriptions from online videos. Additionally, we propose a person-description-based controllable voice synthesis system, establishing a mapping from character descriptions to speaker representation vectors. This system transforms character descriptions into new vectors, serving as input for zero-shot VITS to generate character-specific voices. Both objective and subjective metrics affirm our approach's capability to generate previously unheard character-specific voices with acceptable naturalness. We plan to release the annotation set of TMCSPEECH (We only provide our collected original video links and our annotated labels for non-commercial research purposes. Our shared annotation set does not contain any audio or video data. It is the user's responsibility to decide whether to download the video data and whether their intended purpose with the downloaded data is allowed in their country). Our audio samples can be accessed online (https://raydonld.github.io/TMCSPEECH/).

Keywords: Multi-modal data processing · speech synthesis · voice generation · zero-shot text-to-speech

1 Introduction

With the rapid development of deep learning, the quality of speech synthesis has significantly improved [27]. However, current mainstream speech synthesis systems still face substantial limitations. Though they achieve high-quality synthesis

We only provide our collected original video links and our annotated labels for non-commercial research purposes. Our shared annotation set does not contain any audio or video data. It is the userâĂŹs responsibility to decide whether to download the video data and whether their intended purpose with the downloaded data is allowed in their country.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 177–189, 2025.
https://doi.org/10.1007/978-3-031-78172-8_12

for speakers within the training set, for speakers not included in the training set, synthesis quality frequently fails to meet expectations. In the thriving era of Artificial Intelligence Generated Content (AIGC), there is a growing demand for personalized voice generation, especially in scenarios such as audiobooks where each character has unique traits. Specifically, in audiobooks, current systems struggle to provide voices that match the unique characteristics of each character. For example, characters with cautious or arrogant personalities are often not accurately represented in synthesized speech, as current systems fail to capture these nuances. Therefore, the current audiobook reading experience often suffers from a lack of variety in voices, making it challenging for listeners to distinguish between characters solely based on voice, thereby reducing immersion and comprehension. In addition, recording real voices for each character in a novel, tailored to their characteristics, is impractical and economically inefficient, as it would significantly increase data collection costs and fail to meet the rapidly expanding content demands. Consequently, synthesizing voices that align with character descriptions in audiobooks efficiently becomes the focus of this research.

Recent efforts have focused on synthesizing voices absent from the training set. For example, Stanton et al. [26] proposed Tacospawn, which utilizes Mixture Density Networks (MDN) to infer the conditional distribution of the voice representation of multiple speakers in a Tacotron model under discrete label conditions, achieving the conditional generation of new voices. Bilinski et al. [2]. implemented Tacospawn in a Flow-TTS based system, confirming the efficacy of GMM-based methods in characterizing voice representation distributions. However, these methods use discrete labels, and Gaussian Mixture Models have limited capabilities in modeling category boundaries, resulting in weak controllability. Recently, an increasing number of works have applied prompts to audio generation [13,19] and style-controlled speech synthesis [16,18,28]. Inspired by these works, using Prompt descriptions instead of discrete variables as parameters for voice-controllable descriptions can greatly enhance the freedom of voice generation. PromptTTS [10], proposed by Guo et al., introduced text descriptions of sound into TTS for the first time, achieving the generation of speech that meets specific requirements through natural language descriptions of sound. However, this method is not suitable for controllable voice generation, as there is a one-to-many relationship between the text description and the voice. PromptSpeaker [29], proposed by Zhang et al., introduced Glow [17] into a zero-shot VITS [15] to establish a mapping relationship between Speaker representation and Semantic representation and used Prompt Encoder to align Prompt descriptions with the Semantic Representation distribution.

Despite the achievements of the above methods in describing sound using Prompt, they all require manual annotation of audio data. This greatly limits the scale of the voice dataset with Prompt text descriptions, thereby restricting the effectiveness of controllable voice generation. Specifically, in the task of audiobooks, we need to customize the voice according to the appearance, facial features, and personality traits of the characters. The above methods are not well-suited to this requirement as they do not focus on descriptions of characters' appearances and personalities, and there is currently a lack of audiobook speech synthesis datasets with character descriptions. Therefore, the primary challenge in

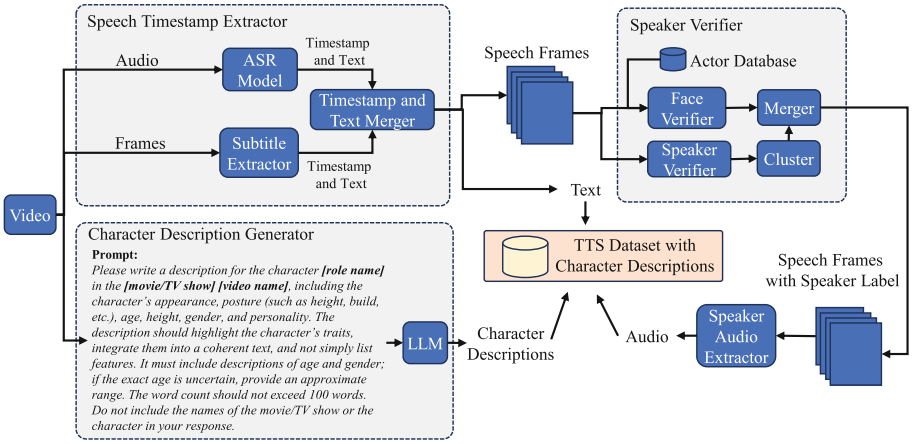


Fig. 1. The overview of our multi-modal extraction pipeline.

contemporary research lies in acquiring speech synthesis databases with detailed character descriptions and developing systems that leverage these descriptions.

To address these research challenges, we have developed an automatic multi-modal extraction pipeline and a controllable voice synthesis system based on character text descriptions. The purpose of this pipeline is to accurately extract the speech audio of each character from videos and simultaneously generate corresponding character descriptions. This pipeline consists of four components: the speech timestamp extractor, speaker verifier, speaker audio extractor, and character description generator. Use this pipeline to process abundant online video resources, substantially reducing data collection and annotation costs. Our proposed controllable voice synthesis system, grounded in character descriptions, encompasses four primary components: a pretrained large language model, a prompt encoder, a normalizing flow model, and a zero-shot TTS system. This system can generate speaker voices that match the feature descriptions based on character traits.

2 Methods

2.1 Automatic Multi-modal Extraction Pipeline

As illustrated in Fig. 1, the multi-modal extraction pipeline consists of a speech timestamp extractor, speaker verifier, speaker audio extractor, and character description generator. The speech timestamp extractor initially isolates timestamps and corresponding texts of speech segments from video data. Next, video segments linked to these timestamps undergo analysis by the speaker verifier, which ascertains the speaker's identity by correlating facial and voice features within each segment. The extracted audio from the video is then processed by the speaker audio extractor to remove background noise and perform noise reduction, selecting high-quality audio. Lastly, the character description generator takes the roles and their corresponding video names from the video, generating character

descriptions that match the characteristics of the characters. Detailed descriptions of each component within this pipeline are presented in subsequent sections.

Step 1: Speech Timestamp Extractor

The Speech Timestamp Extractor module accurately extracts timestamps and corresponding textual content from speech segments in video data. To achieve this, we employ the Paraformer-large speech recognition method [9], specifically designed for efficient extraction of speech timestamps and text from lengthy audio. Furthermore, we have developed a subtitle timestamp extractor by integrating Video-SubFinder[1] and PaddleOCR[2], enabling the recognition of subtitle timestamps and text. Finally, we built a text merger, which merges video subtitles and speech recognition results based on video subtitle recognition results and timestamps. We used Intersection over Union(IOU) and Levenshtein distance[3] to sift the timestamps and text results to obtain the final timestamps and corresponding text for speech segments in the current long video.

Step 2: Speaker Verifier

The core task of the speaker verifier is to confirm the attribution of each video segment to a specific speaker. This module comprises a speech speaker verifier and a face verifier. In the speech speaker verifier, we used a speaker verification extraction model employing the ResNet101-ASP architecture from [22] for the speaker representation extraction system. The ResNet101 model [11] with residual module channel settings $[32, 64, 128, 256]$ serves as the front-end feature extractor. It is followed by an Attentive Statistics Pooling (ASP) layer [20], and a 256-dimensional fully connected layer is used as the speaker representation layer. ArcFace [6] ($s = 32, m = 0.2$) is utilized as the classifier. The model is trained on the VoxCeleb2 dataset [4] (5994 speakers with 1092009 utterances). In the face verifier, we used the RetinaFace [5] as our Face detection model, and the IResNet [8] and ArcFace as the face recognition model. Due to the scarcity of Asian faces in existing open-source face recognition datasets, leading to inaccuracies in face recognition, we fine-tuned the face recognition model using a dataset consisting of images of 7,000 various celebrities obtained from the internet. Simultaneously, to automate the verification of characters in each film or TV show, we crawled information for 2,000 films and TV shows, including actor names, character names, dubbing names, and actor photos from Baidu Baike[4]. When using the face verifier, we register faces appearing in the video using the actor photos that we crawled online, match them with faces in the video segments, and extract speaker representation vectors from the corresponding audio segments using the speaker verification extraction model. Ultimately, we combine face verification outcomes with speaker verification results obtained through K-Means clustering to ascertain each video segment's speaker attribution.

[1] https://sourceforge.net/projects/videosubfinder/.
[2] https://github.com/PaddlePaddle/PaddleOCR.
[3] Python implement: https://pypi.org/project/fuzzywuzzy/.
[4] https://baike.baidu.com/.

Step 3: Speaker Audio Extractor

After successfully extracting all video segments attributed to speakers, we separate the corresponding audio and use a 5-stem spleeter [12] to extract human voices from the audio. To ensure audio quality, we only use the vocal track of the multi-channel signals derived from the spleeter toolkit.

Step 4: Character Description Generator

Drawing inspiration from Stanford Alpaca's application of large language models in data generation, we formulated a Prompt specifically for generating character descriptions in videos. Focused on processing Chinese-language video data, we utilized the powerful Chinese understanding and generation capabilities provided by the Baidu Large Language Model API called ERNIE-Bot 4.0[5]. The specific Prompt is shown in Fig. 1. This approach allows us to generate unique and distinctive character descriptions for each role.

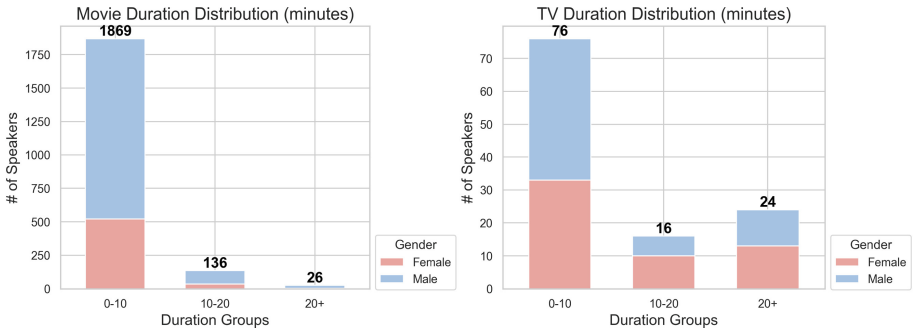


Fig. 2. Data statistics crawled from the Internet.

Table 1. Results of TV show data processing using the multi-modal pipeline.

Type	Number of Speakers	Duration (min)	Gender Ratio (Male/Female)
Movies	2032	10019.88	2.63
TV Shows	116	1706.16	1.07

After completing the aforementioned four steps, we obtain a speech synthesis dataset with character trait descriptions and high-quality audio.

Finally, we extracted 2,032 speakers from 783 movies, totaling 10,019.77 min, and 116 speakers from 3 TV Shows, totaling 1,706.16 min. As illustrated in Table 1, the proportion of male speakers in the extracted movie audio data is significantly higher than that of females, while the gender ratio is relatively balanced in TV show data.

Figure 2 reveals that 92% of speakers in movie data have speaking durations of less than ten minutes, with males outnumbering females. There are 136

[5] https://cloud.baidu.com/doc/WENXINWORKSHOP/s/clntwmv7t.

speakers within the 10 to 20-min range and 26 speakers with durations exceeding 20 min. This distribution aligns with the characteristics of movies, where each movie's duration is short, but the number of actors is high. TV show data indicates a more balanced gender ratio, with a significant proportion of speakers having durations exceeding 20 min, reflecting the longer speaking times of each actor in TV shows.

2.2 Controllable Voice Synthesis System Based on Character Descriptions

We present a controllable voice synthesis method based on character descriptions, as illustrated in Fig. 3. The method primarily consists of four components: a pretrained large language model, a zero-shot TTS system, a reversible normalizing flow model, and a prompt encoder. In our method, we can accept two types of input to synthesize a controllable voice: the collection of movie names and role names or the character description. The pretrained large language model can generate character descriptions by providing movie names and role names. The zero-shot TTS system is designed to generate speech for a specific speaker, utilizing the speaker representation vector as input. The reversible normalizing flow model [24] decouples the speaker representation vector into a semantic representation vector consisting of quantized and non-quantized regions. The quantized region explicitly encodes attributes such as gender, age, and SNR, while the non-quantized region encompasses other non-quantifiable sub-linguistic information about the speaker. The prompt encoder identifies gender and age through the natural language description of the character and predicts the mean and variance of the non-quantized region's representation vector's prior distribution. By

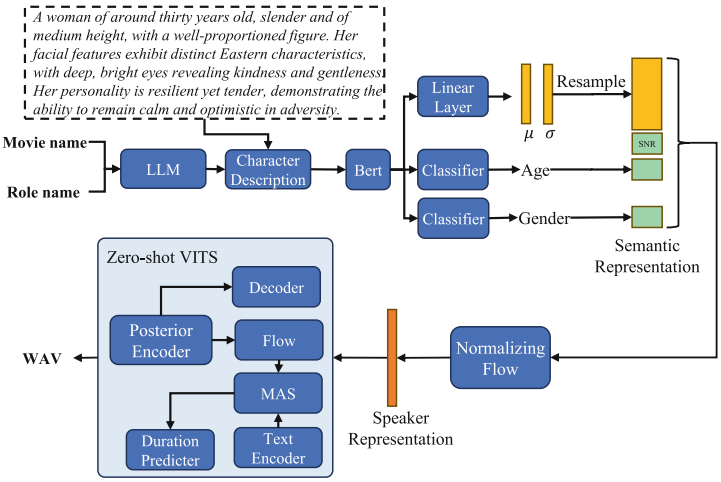


Fig. 3. Architecture of our model.

concatenating age, gender information, and the vector representation of the non-quantized region, the semantic representation vector is obtained after transformation through the normalizing flow model. Finally, this speaker vector is used as input to the zero-shot TTS system to synthesize speech that aligns with the character description.

Zero-Shot TTS System

To synthesize speech for speakers not present in the training set, we employ the zero-shot TTS system. In recent years, the VITS structure [15] has achieved significant success in fields such as zero-shot speech synthesis [3] and voice cloning [1,14]. Therefore, we select pre-trained VITS based on speaker representation vectors for our zero-shot TTS system. The system is built upon the VITS model, pre-trained on the AISHELL-3 dataset [23], and further fine-tuned using speech data obtained through our proposed automated multi-modal extraction pipeline. This fine-tuning process is designed to elevate the vividness and naturalness of the synthesized speech, ultimately leading to a more authentic and expressive speech synthesis outcome.

Reversible Normalizing Flow Model

To achieve stable controllability of synthesized voices concerning gender and age, ensuring consistency between synthesized voices and descriptions, we employ a reversible normalizing flow model [21]. We aim to decouple attributes such as gender and age from the speaker representation vector. We adopt the VoiceLens method proposed by Shi et al. [24], which utilizes conditional normalizing flow to map the voice represented by the speaker vector to a latent space. The semantic representation vector in this space can quantify attributes such as gender, age, and SNR. Hence, we choose VoiceLens as our decoupling inverse transformation model.

Table 2. Data usage description for each module

Module	Training set	Testing set
Zero-shot TTS	TV Show Dataset(Train)	Movies Dataset(Test)
Reversible Normalizing Flow Model	Movies Dataset(Train)	Movies Dataset(Test)
Prompt Encoder	Movies Dataset(Train)	Movies Dataset(Test)

Prompt Encoder

The goal of the prompt encoder is to extract semantic information from the textual character description, identify the character's gender and age, and predict the mean and variance of the prior distribution of the non-quantized region of the semantic representation vector. This module consists of a pre-trained Chinese BERT module [7] and a multi-head prediction layer. The multi-head prediction layer includes three parallel linear layers, each predicting gender and age, as well as the mean and variance of the prior distribution of the non-quantized region in the text. Initially, the character description text is processed through the pre-trained Chinese BERT module to extract semantic information. Then, the

semantic information is passed to the multi-head prediction layer to predict the character's gender, age, and the mean and variance of the prior distribution of the non-quantized region, which is then used to sample the vector representation of the non-quantized region of the semantic representation vector.

3 Experiments

3.1 Experiment Setting

Owing to the distinctive attributes of the tasks, we utilize TV show data featuring longer speaker's speech durations for fine-tuning the Zero-shot TTS System mentioned earlier, aiming to enhance the vividness of synthesized speech. To establish a correlation between speaker embedding and textual character descriptions, we employ movie data rich in diverse speakers for training and testing the Reversible Normalizing Flow Model and the Prompt Encoder. We excluded speakers with brief speaking durations from the multi-modal extraction pipeline's audio data, utilizing the remaining data for training and testing our model. We list the data utilized in each module in Table 2. TV Show Dataset(Train) including 45849 utterances for 18 speakers. Movies Dataset(Train) including 68640 utterances for 858 speakers. And Movies Dataset(Test) including 17160 utterances for 214 speakers.

3.2 Evaluation Metrics

This paper evaluates the proposed character description-based controllable voice synthesis system, encompassing both objective and subjective evaluations. For both evaluations, 20 speakers were selected from each category. In our objective evaluations, the speaker distance assessment was based on the x-vector method [25]. The calculation method is consistent with that described in the literature [26,29]. The x-vector represents speaker characteristics for each audio segment. The speech speaker verifier mentioned in Sect. 2.1 was employed to obtain the x-vector for each audio segment.

Subsequently, we calculated the average x-vector for each speaker, obtaining a speaker-level x-vector (denoted as V). We performed distance measurements on speaker-level embedding V from three different types:

- **Speaker-level x-vector for ground-truth target speaker utterances (gt)**: we compute V_i^{gt} by averaging the x-vectors of all utterances from speaker i on the ground-truth target audio in the training set.
- **Speaker-level x-vector for synthesized speech utterances generated by using the ground-truth speaker-level x-vector (syn)**: we compute V_i^{syn} by averaging the x-vectors of all utterances from speaker i on the synthesized audio in the training set.
- **Speaker-level x-vector for generated speech utterances using character text prompts(gen)**: we compute V_i^{gen} by averaging the x-vectors of all generated speech utterances from speaker i when given the speaker prompt from testing set.

We differentiated between different speakers by computing the cosine distance between different V. The cosine distance is defined as $d(V_1, V_2) = 1 - \frac{V_1}{\|V_1\|} \cdot \frac{V_2}{\|V_2\|}$. We utilized the cosine distance between V obtained from the speech synthesis of speakers in the training set as a threshold to assess the system's performance in voice generation. We define the set of training speakers as T, and the set of generated speakers by Prompt in the testing set as G, and the following metrics are the six metrics that were computed to evaluate performance in voice generation:

- **Syn2gt-same**: Compute the speaker-level x-vector distance between synthetically generated speech V_i^{syn} and the corresponding ground truth V_i^{gt} for the same speaker in the training set. The smaller, the better.

$$\underset{i \in T}{\text{mean}}\, d(V_i^{syn}, V_i^{gt}) \tag{1}$$

- **Syn2gt-near**: Calculate the average distance between the training speaker V_i^{syn} and the closest ground truth training speaker V_i^{gt}. This metric is used to assess the differences between synthesized speech and the ground true speech of other speakers in the training set. The larger, the better.

$$\underset{i \in T}{\text{mean}} \min_{j \in T, i \neq j} d(V_i^{syn}, V_j^{gt}) \tag{2}$$

- **Gt2gt-near**: Calculate the average distance between the different ground truth training speakers V_i^{gt}. This metric is used to assess the differences between the ground true speech of different speakers in the training set. The larger, the better.

$$\underset{i \in T}{\text{mean}} \min_{j \in T, i \neq j} d(V_i^{gt}, V_j^{gt}) \tag{3}$$

- **Syn2syn-near**: Compute the average minimum distance between synthesized speech V_i^{syn} for different speakers in the training set. This metric is applied to measure the worst-case performance in audio synthesis for different speakers within the training set. The larger, the better.

$$\underset{i \in T}{\text{mean}} \min_{j \in T, i \neq j} d(V_i^{syn}, V_j^{syn}) \tag{4}$$

- **Gen2syn-near**: Calculate the average minimum distance between audio generated from a Prompt-derived speaker V_i^{gen} in the testing set and speech synthesized V_j^{syn} for speakers in the training set. This metric assesses the worst-case performance in distances between a Prompt-generated speaker and other speakers' synthesized speech derived from the training set. The larger, the better.

$$\underset{i \in G}{\text{mean}} \min_{j \in T} d(V_i^{gen}, V_j^{syn}) \tag{5}$$

- **Gen2gen-near**: Compute the average-worst case distance between speakers generated under the same Prompt. This metric is utilized to evaluate the lower bound of the richness of speaker generation based on the same Prompt. The larger, the better.

$$\underset{i \in G}{\text{mean}} \min_{j \in G, i \neq j} d(V_i^{gen}, V_j^{gen}) \tag{6}$$

Table 3. Cosine distance between different speakers

syn2gt-same ↓	syn2gt-near ↑	syn2syn-near ↑	gen2syn-near ↑	gen2gen-near ↑	gt2gt-near ↑
0.135	0.389	0.306	0.377	0.234	0.327

Table 4. The naturalness MOS with 95% confidence intervals

System	MOS score
Ground truth	4.80 ± 0.03
Synthetic Speakers	3.53 ± 0.07
Generated Speakers	3.35 ± 0.07

In evaluating the gender prediction capabilities of our prompt encoder within the testing set, we calculate the accuracy based on character descriptions. Accuracy A is defined as $A = \frac{C_{\text{correct}}}{N_{\text{total}}} \times 100\%$, where C_{correct} is the number of correct gender predictions and N_{total} is the total predictions made.

For the subjective evaluation, the Naturalness Mean Opinion Score (MOS) was employed to assess 60 audio samples, including real, synthetic, and test speaker audio. The synthetic speaker's speaker embedding was generated solely from character descriptions of the ground truth speakers, converting text descriptions into speaker embeddings via text embeddings, followed by zero-shot TTS generation. Conversely, test speakers' audio was produced using character descriptions from the test set, following the same conversion process. This approach allowed us to evaluate the system's effectiveness in synthesizing voices based on new, unseen descriptions, with 12 listeners participating in the evaluation.

3.3 Experimental Results

Table 3 presents the objective evaluation results of speaker similarity. Considering the robustness of our speaker verification model, the scores of syn2gt-same and syn2gt-near reflect our ability to clone the reference speaker's voice well, with good distinguishing ability between different speakers and a good speech synthetic performance. The syn2syn-near indicates that the speech generated by our synthetic model has distinctiveness between different speakers. The result of gen2syn-near suggests that our method generates new speaker voices which are much more distinct from those in the training set, demonstrating its capability to produce novel speaker voices absent in the training data. The gen2gen-near result shows that our method can generate different new voices when the same character description is used multiple times. The gt2gt-near represents the differences between ground truth speeches of different speakers. Additionally, we also conducted gender accuracy evaluation on the newly generated voices by our method. In the testing set, the gender accuracy is 98.69%, indicating that our

method effectively captures gender characteristics from text inputs, enabling the TTS system to generate speech outputs with correct gender attribution.

In the subjective evaluation results provided in Table 4, our system achieves a good naturalness Mean Opinion Score (MOS) when using the genuine speaker embeddings from the testing set. The synthesized speech of new speakers generated based on character descriptions shows only a slight decrease in naturalness MOS compared to the speech generated using genuine speaker embeddings. These results indicate that our system is capable of producing high-quality new speaker voices based on textual character descriptions. Our audio samples can be accessed by visiting the following URL: https://raydonld.github.io/TMCSPEECH/.

4 Conclusions

This paper introduces an innovative multi-modal extraction pipeline efficiently designed to extract speech segments and corresponding character descriptions for each role from video data. The experimental results demonstrate that our pipeline can automatically obtain substantial quantities of accurate and high-quality character speech segments and descriptions. Additionally, we construct a controllable voice synthesis system based on character descriptions. This system establishes a one-to-many mapping relationship between character description text and speaker representation vectors, achieving the transformation from character descriptions to new speaker representation vectors. Specifically, the character description undergoes joint processing by the prompt encoder and normalization flow model, generating the input vector for the zero-shot TTS system and subsequently synthesizing speech that aligns with the character description. Experimental validation indicates that our approach is capable of generating a diverse range of voices from unseen character descriptions while maintaining a high degree of naturalness in the synthesized speech. In this work, our method still converts the character texts into speaker embeddings and then feeds them to the TTS module. In our future works, we plan to directly feed the character text prompt to the large generative TTS model with discrete tokens, hoping to achieve better speech synthesis results.

Acknowledgement. This research is funded in part by the National Natural Science Foundation of China (62171207), Guangdong Science and Technology Plan (2023A1111120012) and OPPO. Many thanks for the computational resource provided by the Advanced Computing East China Sub-Center.

References

1. Arik, S., Chen, J., Peng, K., Ping, W., Zhou, Y.: Neural voice cloning with a few samples. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
2. Bilinski, P., et al.: Creating new voices using normalizing flows. arXiv preprint arXiv:2312.14569 (2023)

3. Casanova, E., Weber, J., Shulby, C.D., Junior, A.C., Gölge, E., Ponti, M.A.: YourTTS: towards zero-shot multi-speaker TTS and zero-shot voice conversion for everyone. In: International Conference on Machine Learning, pp. 2709–2720. PMLR (2022)
4. Chung, J.S., Nagrani, A., Zisserman, A.: VoxCeleb2: deep speaker recognition. arXiv preprint arXiv:1806.05622 (2018)
5. Deng, J., Guo, J., Ververas, E., Kotsia, I., Zafeiriou, S.: RetinaFace: single-shot multi-level face localisation in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5203–5212 (2020)
6. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: ArcFace: additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4690–4699 (2019)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
8. Duta, I.C., Liu, L., Zhu, F., Shao, L.: Improved residual networks for image and video recognition. In: 2020 25th International Conference on Pattern Recognition, pp. 9415–9422. IEEE (2021)
9. Gao, Z., Zhang, S., McLoughlin, I., Yan, Z.: Paraformer: fast and accurate parallel transformer for non-autoregressive end-to-end speech recognition. arXiv preprint arXiv:2206.08317 (2022)
10. Guo, Z., Leng, Y., Wu, Y., Zhao, S., Tan, X.: PromptTTS: controllable text-to-speech with text descriptions. In: IEEE ICASSP, pp. 1–5. IEEE (2023)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
12. Hennequin, R., Khlif, A., Voituret, F., Moussallam, M.: Spleeter: a fast and efficient music source separation tool with pre-trained models. J. Open Source Softw. **5**(50), 2154 (2020). https://doi.org/10.21105/joss.02154. Deezer Research
13. Huang, R., et al.: Make-an-audio: text-to-audio generation with prompt-enhanced diffusion models. arXiv preprint arXiv:2301.12661 (2023)
14. Jia, Y., et al.: Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
15. Kim, J., Kong, J., Son, J.: Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In: International Conference on Machine Learning. pp. 5530–5540. PMLR (2021)
16. Kim, M., Cheon, S.J., Choi, B.J., Kim, J.J., Kim, N.S.: Expressive text-to-speech using style tag. arXiv preprint arXiv:2104.00436 (2021)
17. Kingma, D.P., Dhariwal, P.: Glow: generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
18. Liu, G., et al.: PromptStyle: controllable style transfer for text-to-speech with natural language descriptions. arXiv preprint arXiv:2305.19522 (2023)
19. Liu, H., et al.: AudioLDM: text-to-audio generation with latent diffusion models. arXiv preprint arXiv:2301.12503 (2023)
20. Okabe, K., Koshinaka, T., Shinoda, K.: Attentive statistics pooling for deep speaker embedding. arXiv preprint arXiv:1803.10963 (2018)
21. Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. J. Mach. Learn. Res. **22**(1), 2617–2680 (2021)

22. Qin, X., et al.: The DKU-Tencent system for the VoxCeleb speaker recognition challenge (2022). arXiv preprint arXiv:2210.05092
23. Shi, Y., Bu, H., Xu, X., Zhang, S., Li, M.: AISHELL-3: a multi-speaker mandarin TTS corpus and the baselines. arXiv preprint arXiv:2010.11567 (2020)
24. Shi, Y., Li, M.: VoiceLens: controllable speaker generation and editing with flow. arXiv preprint arXiv:2309.14094 (2023)
25. Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S.: X-vectors: robust DNN embeddings for speaker recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5329–5333. IEEE (2018)
26. Stanton, D., et al.: Speaker generation. In: IEEE ICASSP, pp. 7897–7901. IEEE (2022)
27. Tan, X., Qin, T., Soong, F., Liu, T.Y.: A survey on neural speech synthesis. arXiv preprint arXiv:2106.15561 (2021)
28. Yang, D., et al.: InstrucTTTS: modelling expressive TTS in discrete latent space with natural language style prompt. arXiv preprint arXiv:2301.13662 (2023)
29. Zhang, Y., et al.: PromptSpeaker: speaker generation based on text descriptions. arXiv preprint arXiv:2310.05001 (2023)

Deterministic Synthesis of Defect Images Using Null Optimization

Hyunwook Jo, Marcellino Sahadewa, William Gazali, and In Kyu Park(✉)

Department of Electrical and Computer Engineering, Inha University, Incheon 22212, Korea
pik@inha.ac.kr

Abstract. In the manufacturing industry, defect classification is crucial but is hampered by challenges from imbalanced data, which often leads to model overfitting when data are scarce. One common solution is to use synthetic data from generative models; however, these models typically produce results that are structurally inconsistent. Addressing these concerns, this paper introduces a novel null embedding optimization technique that generates latent representations closely resembling the original images, significantly enhancing the fidelity of the generated images. This method ensures that synthetic images are not only visually similar to the original images but also subjected to a more extensive and diverse augmentation process, increasing the variability within the dataset. Consequently, this approach effectively doubles the usable dataset size and notably increases the accuracy of classification models by up to 11%. For those interested, the dataset and source code are accessible at https://github.com/ugiugi0823/DISN.

Keywords: Defect classification · Diffusion · Image synthesis

1 Introduction

Defect detection is critical in manufacturing, where a 95% yield rate corresponds to a 5% defect rate. Even when manufacturers strive to improve yields, enhancements often result in insufficient data collection and labeling for defects. Moreover, computer vision technologies for defect detection continue to evolve. However, with the rise in deep learning, datacentric learning approaches have become the standard. Training a deep learning model for defect classification requires normal and defect samples. Ideally, actual defect data collection would be most effective. However, given the efforts of engineers to reduce defects, this is nearly impossible. Moreover, generative models generate irregular or inconsistent structure images [12], resulting in synthesized images that are deformed and inconsistent with the intended output. Consequently, considerable research has focused on generating synthetic defects [1,11,15,19,23,26]. We believe that

M. Sahadewa and W. Gazali—Equal Contribution.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 190–205, 2025.
https://doi.org/10.1007/978-3-031-78172-8_13

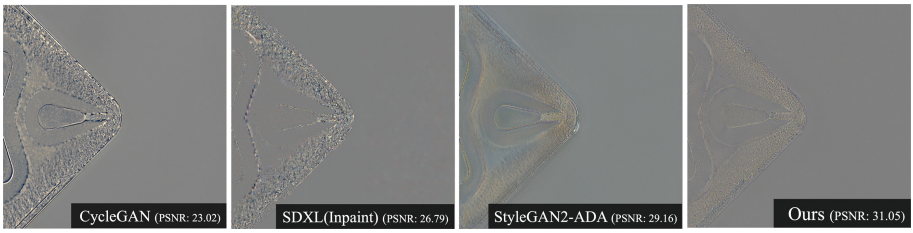


Fig. 1. This residual corresponds to the difference between the original and synthesized images. Our method outperforms previous works [9,16,29] by achieving lowers error and higher PSNR, highlighting its effectiveness in high-fidelity image synthesis.

to be applicable in real industrial environments, synthetic defect samples should meet three stringent criteria: closely resembling actual defects, generating defects without structural inconsistency, and being suitable for automatic labeling.

We argue that generating defects without structural inconsistency is essential for enabling automatic labeling. However, a review of various generative models currently in use reveals a significant contradiction. These models meet the first criterion by producing results that closely mimic real defects; however, they often fail the second criterion because of their inherent structural inconsistency. This structural inconsistency might be due to the stochastic nature inherent in all generative algorithms, which are typically based on Gaussian noise [3,6,7,16,18, 20,21]. This randomness makes them work sufficiently, but it also makes them hard to control. When the same noise vector is fed into the model multiple times, it results in different images each time. This inconsistency is a direct consequence of the stochastic nature of generative algorithms and the challenging task of managing Gaussian noise.

This problem raises a crucial question: how can we effectively manage structural inconsistency? To answer this question, we propose a novel methodology that closely resembles the original product and maintains a high peak signal-to-noise ratio (PSNR), as illustrated in Fig. 1. We improve the existing method by applying LoRA-weights [7] using Stable Diffusion XL (SDXL) [16] and optimizing null embedding and null pool embedding to achieve an optimal latent representation, allowing for the generation of defects highly similar to the original. Our contributions are summarized as follows.

- We propose the use of null embedding optimization to acquire the same latent space as the original thereby generating synthetic defects that closely resemble real defects without structural inconsistency.
- We can obtain various defect samples via attention map blending and amplification.
- We demonstrate the increase in the accuracy of off-the-shelf defect classification models by using synthetically generated images.

2 Related Works

2.1 Defect Classification

Defect classification is an essential task that is based on machine learning and is required in various real-life application areas, such as industrial manufacturing, safety inspections, and architectural structure examinations. With the rise in the deep learning era, which has demonstrated outstanding performance in image classification and object detection, these techniques have begun to be applied in the field of defect classification [2,10,14]. This approach enables us to classify objects without the need for manual feature extraction, reduces the difficulty of algorithm development, and enhances the performance of defect classification algorithms.

2.2 Synthetic Data Generation via GAN

Recently, there have been numerous attempts [1,15,19,26] have focused on the use of GANs to generate synthetic defect data to increase the performance of defect classification models. Notably, Defect-GAN [26] is a new method that enhances the diversity and reliability of data by synthesizing various defect samples to improve a defect classification network.

However, a significant challenge persists in acquiring a substantial amount of training data, a requirement essential for developing effective GAN models [3]. Given complexity of training data, traditional data augmentation techniques struggled to meet the requirements of GAN models. Faced with this challenge, the SyNDGAN [1] methodology via StyleGAN2-ADA [9] was proposed to address the limited data issue. However, with the emergence of diffusion models, the research focus on image generation has shifted.

2.3 Synthetic Data Generation via Diffusion

Diffusion models [11] can generate high-quality defect images. However, training a model suitable for a domain requires substantial data, which is not always feasible. Thus, many methodologies [7,28] that utilize pretrained models have been introduced. In particular, the application of the LoRA [7] methodology to stable diffusion (SD) [18] has generated various LoRA weights that can be used for a specific type of generation task. The LoRA methodology involves partially freezing the model weight layers in the NLP and training only some of them. Compared with fine-tuning, this methodology achieves performance equivalent to or superior to model quality, offering the advantages of higher training throughput and no additional inference delay.

The DreamBooth methodology [20,23] realized the 'personalization' of text-to-image diffusion models capable of rendering a specific subject in various contexts via a small number of images (typically 3–5 images), by learning a unique identifier for the subject. By fine-tuning a pretrained model, the subject can be

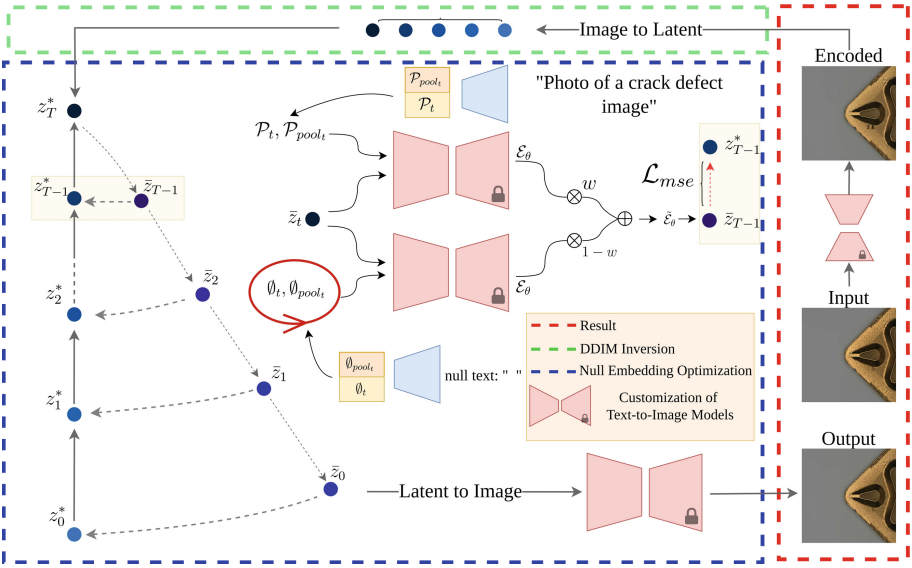


Fig. 2. Our proposed defect synthesis network. *Image to Latent* stage (green dashed line) converts image into latents over several time steps. Through null optimization (blue dashed line), these latents are refined to optimize the original image characteristics and used to generate synthetic images with detailed defects. (Color figure online)

included in the output domain of the model, allowing the creation of new, realistic images of the subject contextualized in different scenes via the identifier. However, this method suffers from structural inconsistency [12].

3 Proposed Method

As illustrated in Fig. 2, our method leverages the capability of the pre-trained SDXL model to generate highly realistic defect images. By leveraging pre-trained models, we can harness the ability of the method to approximate intricate probability distributions, which are crucial for high-quality image generation. We employ a customization strategy that includes fine tuning with DreamBooth [20] and LoRA techniques to tailor these models specifically for defect generation (Sect. 3.1). To further enhance our methodology, we utilize DDIM inversion [13] for efficient latent value extraction, enabling faster and more precise image generation (Sect. 3.2). Then, to address the structural inconsistency issue, we utilize null embedding optimization (Sect. 3.3). This method improves our model's performance in generating detailed images closely resembling the original defect characteristics. Our scalable approach allows for nuanced control over the generated images, employing techniques such as a control step and activation of attention maps [13] to adjust the defect generation process diversely. The detailed information is presented in Sect. 4.3.

3.1 Customization of Text-to-Image Models

Our method uses the DreamBooth [20] methodology to obtain LoRA-weights for future use. The cornerstone of this methodology lies in fine-tuning a pre-trained text-to-image diffusion model to learn unique identifiers for specific subjects. A key approach involves employing unique identifiers for subject personalization and integrating the class-specific prior preservation loss [20] to mitigate linguistic drift during the generation of distinct instances of a particular subject. This process enables the generation of new, realistic images across various contexts related to the subject, while preserving its core feature. The class-specific prior preservation loss is represented as followings.

$$\mathcal{L} = \mathbb{E}_{\mathbf{x},t}\left[w_t \left\|\hat{\mathbf{x}}_\theta(\alpha_t \mathbf{x} + \sigma_t, c) - \mathbf{x}\right\|_2^2 + \lambda w_t \left\|\hat{\mathbf{x}}_\theta(\alpha_t \mathbf{x}_{pr} + \sigma_t, c_{pr}) - \mathbf{x}_{pr}\right\|_2^2\right] \quad (1)$$

where $\mathbf{x}$ and $\hat{\mathbf{x}_\theta}$ denote the actual image and the image generated by the model. c is the conditional vector corresponding to the text prompt, α_t and σ_t are the functions that determine the noise schedule of the diffusion process, w_t is a function that determines sample quality, λ is a hyperparameter that adjusts the relative weight of the second term, and $\mathbf{x}_{pr}$ and c_{pr} signify the model-generated data and its conditional vector, respectively. The text prompts for the images are presented as "[unique identifier] [class name]". Currently, meaningless tokens have been employed for training.

3.2 DDIM Inversion

After loading the LoRA-weights trained for SDXL, we utilize DDIM inversion to extract latent values from images. The DDIM [13] enables fast sampling while maintaining the same learning objectives as the traditional Denoising Diffusion Probabilistic Models [6] approach by using a non-Markovian diffusion process. This approach is achieved by deviating from the conventional method of adding noise at each time step, leading to deterministic outcomes instead. The specific approach is expressed as followings.

$$z_{t+1} = \sqrt{\bar{\alpha}_{t+1}} \underbrace{\left(\frac{z_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(z_t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\text{predicted } z_t = f_\theta(z_t)} + \underbrace{\sqrt{1-\bar{\alpha}_{t+1}} \cdot \epsilon_\theta(z_t)}_{\text{direction pointing to } z_t} \quad (2)$$

The core of this process involves obtaining the latent from the image and adding noise to it. Here, z_{t+1} represents the subsequent state, z_t denotes the current state, and $\epsilon_\theta(z_t)$ is the noise predicted via the trained network. This noise prediction function is a core element of diffusion models, typically implemented via a U-Net architecture [16] The function $f_\theta(z_t) = (z_t - \sqrt{1-\bar{\alpha}t} \cdot \epsilon\theta(z_t))/\sqrt{\bar{\alpha}t}$ estimates the original clean data from the noisy state at time t. The term $\sqrt{1-\bar{\alpha}t+1} \cdot \epsilon_\theta(z_t)$ represents the direction pointing to z_t, guiding the process of adding noise to the latent. This method allows for a nuanced control by adjusting the noise ratio at each time step using $\bar{\alpha}_t := -\prod_{i=1}^{t}(1-\beta_i)$ with a

schedule $\beta_0, ..., \beta_T \in (0, 1)$ coefficient. This function, which inverts the denoising process, is crucial for obtaining a latent representation from the original image and progressively adding noise to reach the final latent state at time-step t.

3.3 Null Embedding Optimization

Our goal is to manage the structural inconsistency inherent in generative models. Typically, when generating images via DDIM [13], the extraction of latents during the diffusion process often results in differences from the original image when these latents are combined with text embeddings. To address this issue, we optimize the latents obtained from the reverse transformation and null embeddings at each time step. Commonly, SD [18] models utilize positive and negative inputs. Here, we employ negative embedding as a null embedding component. This approach ensures that the model maintains a complete embedding reflecting the original latent at all stages. Specifically, In particular, we perform optimization across 50 time-steps. However, applying conventional methods [13] to SDXL does not yield satisfactory results. This is because SDXL's architecture includes two encoders and two ClipTokenizers [17]. Therefore, when entering the U-Net architecture is used, null embedding and null pool embedding are utilized. The goal of the optimization is to minimize the following equation.

$$\min_{\emptyset_t, \emptyset_{pool_t}} ||z^*_{t-1} - z_{t-1}(\bar{z}_t, \emptyset_t, \emptyset_{pool_t}, C)||_2^2 \tag{3}$$

where $\emptyset_t$ and $\emptyset_{pool_t}$ are the null embeddings being optimized, z^*_{t-1} is the target latent, $z_{t-1}(\bar{z}_t, \emptyset_t, \emptyset_{pool_t}, C)$ is the predicted latent, and C is the conditional text embedding. The optimization process involves initializing $\emptyset_t$ and $\emptyset_{pool_t}$ randomly, to compute the loss via (3), calculating the gradients of the loss with respect to $\emptyset_t$ and $\emptyset_{pool_t}$, updating $\emptyset_t$ and $\emptyset_{pool_t}$ via the Adam optimizer, and repeating these steps for a fixed number of iterations or until convergence. This optimization process ensures that the generated images closely resemble the original while maintaining the desired characteristics. Our experiments consistently showed convergence with 50 time-steps and 10 iterations per step, demonstrating the method's efficiency and stability

4 Experimental Result

4.1 Experimental Setup

Dataset Gathering. In this paper, we collect authentic product data from a real-world industrial firm specializing in manufacturing insert inspection machinery. The focus is on cemented carbide inserts. Our methodology for collecting insert data follows the approach [1], which involves zooming in on the regions in contact with the machining surface. We captured images of the same area under four distinct lighting setups, *i.e.*, *bright field*, *dark field*, *coaxial*, and *backlight*, ensuring comprehensive data acquisition. Upon classifying the collected data

on the inserts, we identify a pronounced disparity between the counts of normal (3,575 samples) and defect images (761 samples), with defects constituting approximately 18% of the total dataset. This disparity presents a more intricate challenge than prior investigations [15,24,26], especially in terms of data volume and clarity.

Dataset. The original dataset is not gathered via a strict process, resulting in only 8% of the data being usable. It has 761 data points across different defect types, such as *bright field*, *dark field*, *backlight*, and *coaxial*. The *bright field* image setting produces a high-quality image, where the defects are distinctly visible, and the product is clearly delineated. Thus, this experiment focuses solely on data collected via the *bright field* setting from carefully selected products with actual defects to ensure data quality. Given data imbalance, where defect data are less common than normal data. This data imbalance can lead to the model overfitting to the underrepresented classes [8], which is a significant concern in training. This experiment aims to determine whether adding synthesized data to the underrepresented classes in an imbalanced dataset can help reduce the overfitting problems. We also explore whether data augmentation techniques such as MixUp [27] and CutMix [25] provide better strategies to prevent overfitting. Our dataset composition, as detailed in Table 1, reflects the typical class imbalance found in industrial manufacturing settings. We deliberately construct our test set with a greater proportion of defect images than the training set does. This approach allows for a more rigorous and conservative evaluation of the defect detection capabilities of our model, particularly in the presence of limited defect data availability.

Synthetic Data Generation. We construct a dataset for this experiment, which includes both normal and defective data, with an emphasis on varying proportions of synthetic defects to evaluate their impact on model training. The datasets are structured to explore different ratios of normal to defective samples. D_o consists of 236 normal data points and 23 defect data points, presenting a ratio of 1:10, which maintains the data imbalance condition recommended in [8]. D_{S1} keeps the same number of normal and original defective data points, but adds an equal number of synthetic defects, adjusting the ratio to 1:5. D_{S2} increases the number of synthetic defects to 46, thereby increasing the ratio to 1:3. The details about this division are provided in Table 1.

Implementation Details. In the defect classification experiment, models such as ResNet-50, ResNet-101, EfficientNetV2-M, and EfficientNetV2-L [4,22] are utilized. The experiment also includes a comparison of results when using data augmentation techniques such as MixUp [27] and CutMix [25] are used. The entire code developed in this paper was implemented based on the PyTorch framework. To optimize the embedding parameters, we utilize the Adam optimization algorithm, setting the learning rate to $1e{-}4$. This configuration is

Table 1. Training and testing dataset for classification.

Dataset	Original # of Defects	Added Synthetic Defects	Total Defects	Normal	Ratio
Test Dataset (D_{test})	39	0	39	58	1:1.5
Original Dataset (D_o)	23	0	23	236	1:10
Synthetic Dataset 1 (D_{S1})	23	23	46	236	1:5
Synthetic Dataset 2 (D_{S2})	23	46	69	236	1:3

Table 2. Quantitative comparison of image synthesis across diverse models and image size.

Model	Image Size	PSNR↑	SSIM↑	LPIPS↓
Null-text inversion [13]	512	27.73	0.862	0.069
DISN (Ours)		**28.67**	**0.875**	**0.066**
ControlNet(Depth) [28]	1024	22.28	0.721	0.190
CycleGAN [29]		23.02	0.774	0.164
SyNDGAN [1]		23.14	0.783	0.161
ControlNet(Canny) [28]		23.86	0.783	0.160
SDXL (Inpaint) [16]		26.79	0.786	0.303
StyleGAN2-ADA [9]		29.16	0.873	0.226
DISN (Ours)		**31.05**	**0.892**	**0.088**

selected to improve efficiency and boost performance during the embedding optimization process. Computational resource considerations are crucial in this optimization-centric approach. For a single image of 1024×1024 resolution, the process requires approximately 70.07 GB of GPU memory and takes 600.01 s (10 min). Moreover, optimizing for 512×512 image requires approximately 31.96 GB of GPU memory and takes approximately 238.65 s (4 min). These results demonstrate that while increasing the image resolution from 512×512 to 1024×1024 (a fourfold increase in pixels) leads to a fourfold increase in memory usage, the processing time increases by a factor of 2.5.

4.2 High-Quality Defect Synthesis

Table 2 shows that the proposed method (DISN) has superior performance in image synthesis, with significant improvement over prior approaches. Using SDXL and LoRA, we achieve up to 31.05 dB in PSNR, 0.892 in SSIM, and 0.088 in LPIPS at 1024×1024 resolution, consistently outperforming prior methods across all metrics. In terms of the PSNR and SSIM, StyleGAN2-ADA [9] (PSNR: 29.16 dB, SSIM: 0.873) achieves the second-best performance, but our method still outperforms it. With respect to LPIPS, ControlNet (Canny) [28] achieves the second-best performance at 0.160, following our method. These results prove that our approach consistently achieves superior performance in all aspects, including

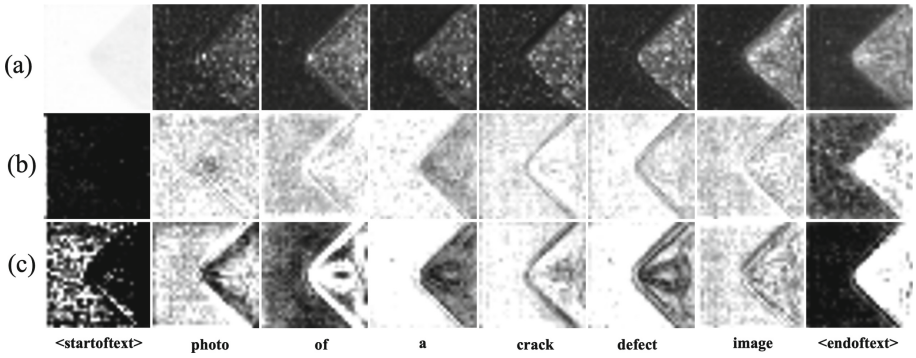


Fig. 3. Attention maps visualized for each token in *photo of a crack defect image* via Stable Diffusion (a), Stable Diffusion XL (b), and Stable Diffusion XL with LoRA (c). This visualization demonstrates how each word contributes to image interpretation, emphasizing the value of individual token analysis in understanding visual defects. Notably, the attention maps become more defined as the model size increases, with the sharpest definition achieved when LoRA is applied. This progression illustrates the effectiveness of LoRA in forming more refined semantic structures, significantly enhancing token-level image comprehension.

image quality, structural similarity, and perceptual similarity. By incorporating a minimum of 62 images in LoRA training, we enable high-resolution attention map generation, which is crucial for producing synthetic images that closely resemble real-world scenarios. This approach, which combines SDXL and LoRA, sets a new standard for high-quality defect image synthesis, consistently outperforming previous methods across multiple evaluation metrics.

4.3 Various Defects Generation

We now have the capability to manage structural inconsistency effectively. Typically, the SD and SDXL models utilize two types of embeddings. Although our approach marginally reduces the PSNR, it employs one of these embeddings to simulate various defects. We can substitute the term *photo of a crack defect image* with various alternatives. Initially, we select the most representative words for defects from the activation maps in Fig. 3. We identified the words *photo*, *crack*, and *defect* as the most influential. Consequently, we experiment with different replacements for each word and find this collection to be the most impactful, significantly altering the appearance of the object. Mokady *et al.* [13] introduced two hyperparameters: the *word amplifier*, which alters the text guidance, and the *replace step*, which controls the number of diffusion processes. The *replace step* specifically modifies the pivotal inversion process to enhance image fidelity and consistency. In particular, we set the *word amplifier* to 0.8 and the *replace step* to 0.7, and employ the blend word technique to change *crack* to *dent*, or *defect* to *dent* thereby altering the appearance of the object. This approach enables the creation of defects in diverse combinations, as shown in Fig. 4.

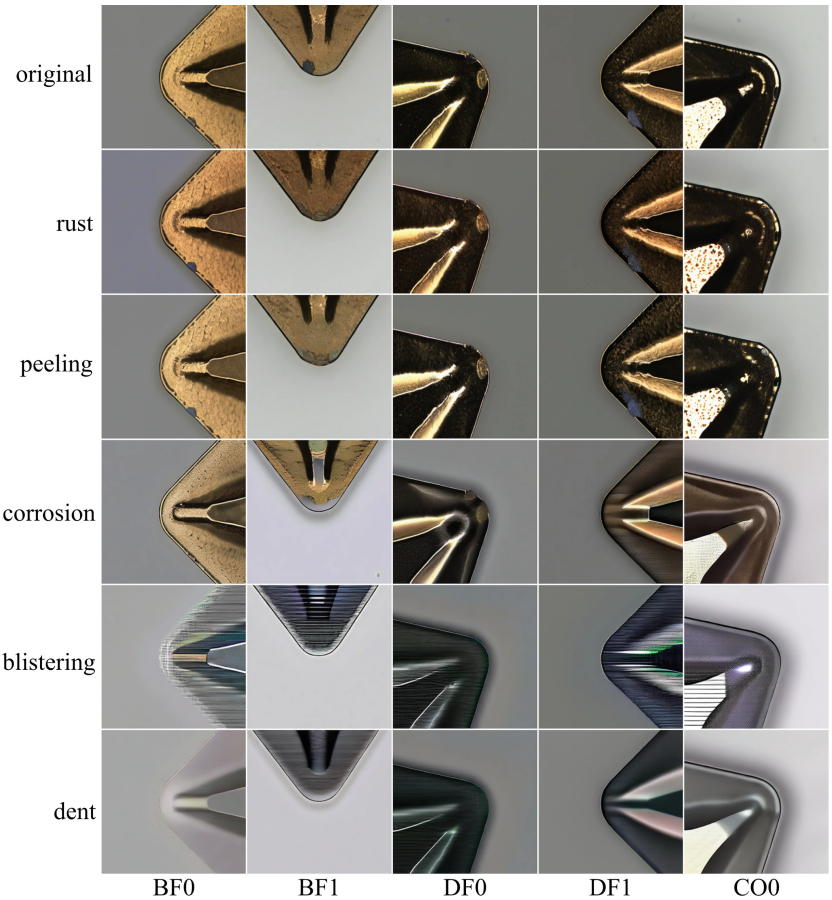


Fig. 4. Various types of defects are generated by replacing the prompt pairs *crack* with *blistering*, *crack* with *dent*, *defect* with *rust*, *photo* with *corrosion*, and *defect* with *peeling*. We tested this approach on five different images labelled BF0, BF1, DF0, DF1, and CO0 and found that the effect of these prompts remained consistent throughout the different images.

4.4 Optimizing Parameters for Diverse and Precise Synthesis

In our exploration of generating diverse defects via the SDXL model, we optimize both text prompts and generation parameters. An analysis of base prompt *photo of a crack defect image*, reveals that modifying the *defect* text has the most significant impact on performance metrics (PSNR, SSIM, and LPIPS). Among the various alternatives tested, *degradation* emerges as the most effective replacement, yielding superior results as shown in Table 3. We also optimize two key generation parameters, *i.e. word amplifier* and *replace step*. Extensive testing (0.0 to 10.0 in 0.1 increments) revealed that a word amplifier value of 2.0 and a replace step of 0.8 produce the best outcomes, differing notably from previous

Table 3. Performance comparison of defect text prompts for optimized synthesis.

Image Size	Defect Type	PSNR ↑	SSIM ↑	LPIPS ↓
512	Original	28.79	0.879	0.046
	Blistering	21.90	0.909	0.090
	Dent	27.75	0.944	0.047
	Rust	27.54	0.938	0.051
	Peeling	28.16	0.941	0.042
	Corrosion	29.35	0.949	0.035
	Wear	30.31	0.953	0.043
	Degradation	31.68	0.954	0.027
1024	Original	31.05	0.892	0.088
	Blistering	24.72	0.939	0.059
	Dent	28.34	0.960	0.025
	Rust	28.76	0.948	0.040
	Peeling	28.29	0.950	0.038
	Corrosion	31.24	0.959	0.040
	Wear	28.71	0.950	0.030
	Degradation	32.12	0.960	0.028

settings (0.8 and 0.7). Our text selection process is informed by defect types common in manufacturing environments, enhancing the practical applicability of our method. This comprehensive optimization significantly improves our text-based diverse defect generation, enabling the synthesis of a wide range of defect types with enhanced precision for industrial applications.

4.5 Defect Classification

Table 4 shows our defect classification results. The experiment is conducted with the dataset settings listed in Table 1. The accuracy consistently increases across all the CNN-based models following the introduction of synthetic defective data. In particular, ResNet-50 [4] achieves improvements in test accuracy from 55.35% to 66.96%, ResNet-101 [4] from 58.92% to 70.53%, EfficientNetV2-M [22] from 68.75% to 75.89%, and EfficientNetV2-L [22] achieves a high accuracy level, peaking at 80.35%. The integration of synthetic defective data noticeably improves the performance of defect detection models, surpassing augmentation techniques such as MixUp [27] and CutMix [25]. This enhancement is apparent across different datasets and model architectures, illustrating the advantages of employing synthetic data in diverse training contexts. The underrepresented class in this dataset is defects, and we add more synthesized defect data. Table 4 shows that the accuracy on the defect set is the highest, indicating that our synthetic data effectively prevent the model from overfitting to the larger class.

Table 4. Accuracy of defect classification. D_O, D_{S1}, and D_{S2} denote the datasets used for the training. Detailed information on the dataset is available in Table 1.

Model	Trained on	Augmented with	Test Accuracy	Defects Set Accuracy	Good Set Accuracy
ResNet-50 [4]	D_O	-	55.35(%)	33.33(%)	71.87(%)
		MixUp [27]	55.35	29.16	79.69
		CutMix [25]	58.03	6.25	89.06
	D_{S1}	-	60.71	**50.00**	68.75
		MixUp [27]	56.25	27.08	73.43
		CutMix [25]	58.92	41.66	70.31
	D_{S2}	-	**66.96**	47.91	**81.25**
		MixUp [27]	58.92	29.16	82.81
		CutMix [25]	61.60	43.75	76.56
ResNet-101 [4]	D_O	-	58.92	20.83	51.56
		MixUp [27]	52.67	37.50	79.68
		CutMix [25]	55.35	22.91	**90.62**
	D_{S1}	-	**70.53**	**43.75**	**90.62**
		MixUp [27]	61.60	27.08	82.81
		CutMix [25]	58.92	47.91	76.56
	D_{S2}	-	66.96	39.58	87.50
		MixUp [27]	60.71	39.58	75.00
		CutMix [25]	66.96	47.91	70.31
EfficientNetV2-M [22]	D_O	-	68.75	41.67	89.06
		MixUp [27]	60.71	29.16	84.37
		CutMix [25]	63.39	39.58	81.25
	D_{S1}	-	74.10	52.08	**90.62**
		MixUp [27]	64.28	52.08	52.08
		CutMix [25]	58.03	18.75	87.50
	D_{S2}	-	**75.89**	**56.25**	**90.62**
		MixUp [27]	61.60	31.25	84.37
		CutMix [25]	65.17	33.33	89.06
EfficientNetV2-L [22]	D_O	-	75.00	54.16	**90.62**
		MixUp [27]	75.00	64.58	82.81
		CutMix [25]	69.64	43.75	89.06
	D_{S1}	-	**80.35**	52.08	**90.62**
		MixUp [27]	75.00	66.66	81.25
		CutMix [25]	69.64	45.83	87.50
	D_{S2}	-	**80.35**	56.25	90.62
		MixUp [27]	74.10	**70.83**	76.56
		CutMix [25]	75.00	58.33	87.5

Notably, EfficientNetV2-L demonstrates less pronounced improvement with synthetic data than the other models do. This is due to its extensive 117M parameter set, aligns with [5] indicating a complex relationship between model size, dataset size, and performance gains. The diminishing returns observed in the

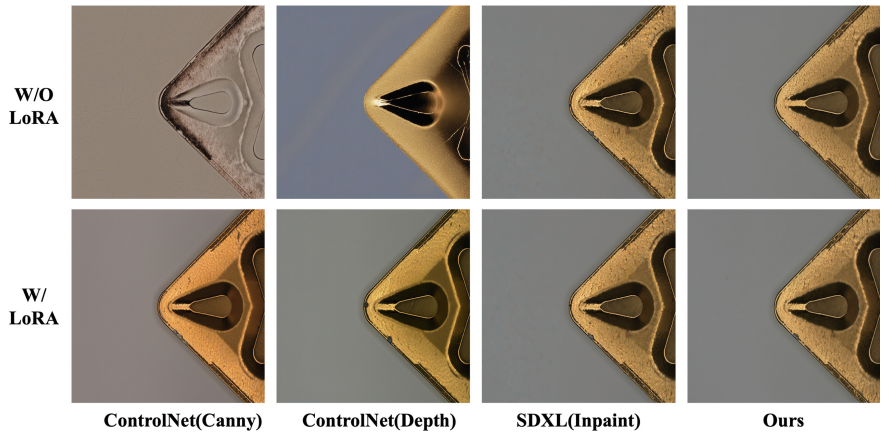


Fig. 5. Impact of LoRA on 1024×1024 defect image quality across diffusion models. The lower-performing models show stark LoRA improvements, whereas the high-performing models exhibit minimal differences.

large model underscore the need for future investigations into optimizing the balance between model architecture and data augmentation strategies, particularly for the imbalanced defect classification task.

5 Ablation Study

5.1 Effect of LoRA

Our paper has demonstrated that LoRA enhances both qualitative and quantitative performance across various diffusion models. As shown in Fig. 5, the impact of LoRA on 1024×1024 defect image quality is more significant in lower-performing models, whereas high-performing models exhibit minimal differences. This trend is supported by Table 5, which shows improvements in the PSNR, SSIM, and LPIPS metrics with LoRA incorporation. For instance, our DISN method shows a modest PSNR increase from 30.62 dB to 31.05 dB. Additionally, Fig. 3 confirms that applying LoRA to SDXL results in improved activation maps, indicating better feature interpretation. Overall, the application of LoRA consistently improves image quality and model interpretability, facilitating the generation of images that more closely align with the original defects.

5.2 Effect of Text

As shown in Table 6, we discover that the specific content of the text does not significantly impact the results. To substantiate this, we construct LoRA models trained with various texts on the same dataset, apply our methodology, and then compare and analyze the outcomes. The analysis confirms that variation

Table 5. Effects of the LoRA on the various translation models.

Model	Image Size	PSNR↑	SSIM↑	LPIPS↓
Null-text inversion [13]	512	27.73	0.863	0.074
Null-text inversion (with LoRA)		27.73	0.862	**0.069**
DISN(Ours)		28.54	0.873	0.084
DISN(Ours) (with LoRA)		**28.67**	**0.875**	**0.066**
ControlNet(Canny) [28]	1024	14.95	0.627	0.474
ControlNet(Canny) (with LoRA)		**23.86**	**0.783**	**0.160**
ControlNet(Depth) [28]		18.63	0.666	0.464
ControlNet(Depth) (with LoRA)		**22.28**	**0.721**	**0.190**
SDXL(Inpaint) [16]		24.58	0.775	0.330
SDXL(Inpaint) (with LoRA)		**26.79**	**0.786**	**0.303**
DISN(Ours)		30.62	0.884	0.090
DISN(Ours) (with LoRA)		**31.05**	**0.892**	**0.088**

Table 6. Comparison result for different text tokens.

Token	Image Size	FID ↓	KID ↓	PSNR ↑	SSIM ↑	LPIPS↓
photo of a defect sks	512	85.17	0.686	28.62	0.874	0.074
abcdefghijklmnop		84.82	0.685	28.63	0.875	0.071
a i o u e		83.36	0.460	28.42	0.872	0.070
<|startoftext|> <|endoftext|>		78.10	**0.346**	**28.75**	**0.876**	**0.065**
photo of golden retriever		77.28	0.608	28.57	0.873	0.068
photo of a crack defect image		**65.89**	0.646	28.67	0.875	0.066

in text content does not meaningfully influence the result. Although the phrase *<|startoftext|> <|endoftext|>* yields the best result, it is crucial to note that this token actually corresponds to an empty word. For practical application in the industry, the text should be in human language to facilitate efficient interaction. Therefore, we chose a *photo of a crack defect image*, which has the highest score on the basis of FID metric, as our base text for the evaluation.

6 Conclusion

We presented a novel method for generating high-fidelity synthetic defect images using SDXL, LoRA weight, and null embedding optimization. Our approach achieved superior performance over existing methods. The null embedding optimization resolved structural inconsistency issues, enabling the creation of diverse, realistic defect images. Our synthetic data significantly improved model accuracy in the classification task. We believe this work represented a significant advancement in both technical aspects and practical applicability for industrial defect detection.

Acknowledgment. This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean

government (MSIT) (No. RS-2022-00155915, Artificial Intelligence Convergence Innovation Human Resources Development (Inha University) and No. RS-2021-II212068, Artificial Intelligence Innovation Hub and IITP-2024-RS-2024-00360227, Leading Generative AI Human Resources Development). This work was supported by Inha University Research Grant.

References

1. Cho, E., Jeon, B., Park, I.K.: Synthesizing industrial defect images under data imbalance. IEEE Access **11**, 111335–111346 (2023)
2. Faghih-Roohi, S., Hajizadeh, S., Núñez, A., Babuska, R., Schutter, B.D.: Deep convolutional neural networks for detection of rail surface defects. In: Proceeding IEEE International Joint Conference on Neural Networks, pp. 2584–2589, July 2016
3. Goodfellow, I., et al.: Generative adversarial nets. In: Proceedings Advances in Neural Information Processing Systems, pp. 2672–2680, December 2014
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, June 2016
5. Hestness, J., et al.: Deep learning scaling is predictable, empirically. CoRR abs/1712.00409, December 2017
6. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Proceedings Advances in Neural Information Processing Systems, December 2020
7. Hu, E.J., et al.: LoRA: low-rank adaptation of large language models. In: Proceedings International Conference on Learning Representations, April 2022
8. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. J. Big Data **6**, 27 (2019)
9. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. In: Proceedings Advances in Neural Information Processing Systems, December 2020
10. Li, Y., Zhao, W., Pan, J.: Deformable patterned fabric defect detection with fisher Criterion-Based deep learning. IEEE Trans. Autom. Sci. Eng. **14**(2), 1256–1264 (2017)
11. Liu, W., Liu, C., Liu, Q., Yu, D.: Assigned MURA defect generation based on diffusion model. In: Proceedings IEEE/CVF Conference on Computer Vision Pattern Recognition Workshop, pp. 4395–4402, June 2023
12. Lu, W., Xu, Y., Zhang, J., Wang, C., Tao, D.: HandRefiner: refining malformed hands in generated images by diffusion-based conditional inpainting. CoRR (2023)
13. Mokady, R., Hertz, A., Aberman, K., Pritch, Y., Cohen-Or, D.: Null-text inversion for editing real images using guided diffusion models. In: Proceedings IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6038–6047, June 2023
14. Mundt, M., Majumder, S., Murali, S., Panetsos, P., Ramesh, V.: Meta-learning convolutional neural architectures for multi-target concrete defect classification with the COncrete DEfect BRidge IMage dataset. In: Proceedings IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11196–11205, June 2019
15. Niu, S., Li, B., Wang, X., Lin, H.: Defect image sample generation with GAN for improving defect recognition. IEEE Trans. Autom. Sci. Eng. **17**(3), 1611–1622 (2020)

16. Podell, D., et al.: SDXL: improving latent diffusion models for high-resolution image synthesis. In: Proceedings International Conference on Learning Representations, May 2024
17. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: Proceedings International Conference on Machine Learning, pp. 8748–8763, July 2021
18. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10674–10685, June 2022
19. Rozanec, J.M., Zajec, P., Theodoropoulos, S., Koehorst, E., Fortuna, B., Mladenic, D.: Synthetic data augmentation using GAN for improved automated visual inspection. CoRR abs/2212.09317, 11094–11099 (2022)
20. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: DreamBooth: fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22500–22510, June 2023
21. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: Proceedings International Conference on Learning Representations, May 2021
22. Tan, M., Le, Q.V.: EfficientNetV2: smaller models and faster training. In: Proceedings International Conference on Machine Learning, pp. 10096–10106, July 2021
23. Valvano, G., Agostino, A., Magistris, G.D., Graziano, A., Veneri, G.: Controllable image synthesis of industrial data using stable diffusion. In: Proceedings IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 5354–5363, January 2024
24. Wang, Y., Wu, C., Herranz, L., van de Weijer, J., Gonzalez-Garcia, A., Raducanu, B.: Transferring GANs: generating images from limited data. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11210, pp. 220–236. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01231-1_14
25. Yun, S., Han, D., Chun, S., Oh, S.J., Yoo, Y., Choe, J.: CutMix: regularization strategy to train strong classifiers with localizable features. In: Proceedings IEEE/CVF International Conference on Computer Vision, pp. 6022–6031, October 2019
26. Zhang, G., Cui, K., Hung, T., Lu, S.: Defect-GAN: high-fidelity defect synthesis for automated defect inspection. In: Proceedings IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2523–2533, January 2021
27. Zhang, H., Cissé, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: beyond empirical risk minimization. In: Proceedings International Conference on Learning Representations, April 2018
28. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings IEEE/CVF International Conference on Computer Vision, pp. 3813–3824, October 2023
29. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings IEEE International Conference on Computer Vision, pp. 2242–2251, October 2017

Adaptive Refiner Based Few-Shot Font Generation

Pratikhya Ranjit[1(✉)], Mohit Gupta[2], Jon von Gillern[2], Venkat Yetrintala[2], and Vipul Arora[1]

[1] Indian Institute of Technology, Kanpur, India
pranjit@iitk.ac.in
[2] Monotype, Delhi, India

Abstract. Creating new fonts is labour and time-intensive. It involves generating a glyph for each character in a given style. Few-shot Font Generation (FFG) uses generative models to generate glyphs that match the target style specified by a few reference glyphs supplied by designers. While state-of-the-art methods generate sharp glyphs, they fail to precisely capture stylistic details of various font styles, leading to inaccurate generation. We propose a refiner model that applies style refinement to the output of any base glyph generator through a series of global and local transformations using spatial transformers and deformable convolutions to better match the target font style. We propose a few-shot adaptation method to adapt the refiner to the target style using the reference glyphs only. The proposed method can be used as an add-on to any existing glyph generator to correct the local incongruencies and better match the target style. Experiments show improvement in the quality of generated glyphs and assessment scores as compared to state-of-the-art methods indicating the effectiveness of the method.

Keywords: font generation · domain adaptation · few-shot learning · adversarial learning

1 Introduction

Fonts provide a visual tone to written text. Different fonts are designed to serve different purposes, such as posters, branding, ancient texts and formal documents. As the usage of text has grown rapidly worldwide, the demand to produce new fonts has also increased. Each font is associated with a font library, which consists of the complete glyph set (the graphic representation of all the characters belonging to the set of alphabets, numeric characters and special symbols) in that font. Traditionally, human experts design font libraries, and the process is quite expensive in terms of time taken, manual labour, and consequently, cost.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_14.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 206–221, 2025.
https://doi.org/10.1007/978-3-031-78172-8_14

For complex fonts and fonts for logographic languages such as Chinese (over 60K glyphs) and Japanese (over 40K glyphs), a human expert can take over a year to complete the font design process.

Recently using generative models for font generation has become increasingly popular. Initial works in font generation include zi2zi [1], which uses pix2pix [12] to learn multiple font styles of logographic languages. [2] employs a CNN model based on tower architecture to generate a fixed glyph set in the target font style. Few-shot Font Generation (FFG) methods developed recently can generate the entire glyph set in a target style using only a few reference glyphs. In this process only a few reference glyphs need to be created by the human designer, leading to significantly faster font generation. FFG methods can be divided into two categories: rasterized glyph generation, which generates glyphs as images, and vector glyph generation, which generates glyphs as vectors (sequential data that encode the glyph structure). A popular approach for rasterized glyph generation used in [15,27,33] is to pose it as an image-to-image translation task where a neural network transforms rasterized glyphs from a base style to a target style while preserving the content. First, the style and content features are disentangled from the reference and base style glyphs respectively using separate encoder branches, and then fused and passed to a decoder branch which generates the rasterized glyph in target style. The base style font, henceforth referred to as the base font, is chosen such that its entire glyphset is available in the font library. Apart from their architecture, FFG models differ in the method of fusing style and content features. DG-Font [33] uses adaptive instance normalization to fuse the style embedding in intermediate layers in the decoder network. FS-Font [27] re-weighs the feature maps obtained from the style encoder based on the content using a multi-head attention network before passing it to the decoder network. DC-Font [15] concatenates the style and content embeddings into a single feature vector, which is then used by the decoder network.

Rasterized glyph generation, while effective for fixed-resolution outputs, encounters scalability limitations since the FFG models are designed to produce glyphs at a predetermined resolution. Unlike their rasterized counterparts, vector glyphs are inherently resolution-independent, facilitating efficient storage and seamless scaling to any desired resolution. In DeepVecFont-v2 [31], the vector glyph is a sequence of drawing commands consisting of move, line and curve commands and a set of control points corresponding to Bézier curves. These commands can be followed sequentially to generate the glyph. In Dual Vector [20], the vector glyph consists of a set of dual parts: a positive and a negative part, where each part is a closed Bézier curve - the combination of these dual parts results in the glyph. The neural networks of DeepVecFont-v2 [31] and Dual Vector [20] consist of two branches: an image branch, which takes as input the rasterized reference glyphs and generates the rasterized glyph in target style, and a vector branch, which takes as input the reference vector glyphs and generates the vector glyph in target style. A differential rasterizer generates the rasterized glyph from the vector glyph, which is compared with the rasterized glyph from the image branch to refine the vector glyph in an end-to-end trainable fash-

ion. VecFusion [28] employs a two-stage cascaded diffusion model to generate high-quality vector glyphs which consist of a sequence of control points forming cubic Bézier curves. The model first produces a low-resolution rasterized glyph and then synthesizes its vector representation, ensuring precise geometry and control point locations.

Despite the success of FFG models in generating high-quality glyphs, these models generalize poorly over various font styles. It is often observed that although FFG models efficiently learn the global shape of glyphs for most of the font styles, they fail to learn the localized style details such as serif, apex or stroke weights of a target style accurately. Finetuning the model for new fonts is not always feasible due to the requirement of a large training dataset and the high cost of training large models. CF-Font [30] uses Iterative Style-vector Refinement (ISR) to adapt to fonts during inference. It finetunes the generator using reference glyphs and its ground truth in a supervised way by minimizing a reconstruction loss. However, the method only applies to models architecturally similar to CF-Font.

The Main Contributions of Our Work Include:

1. a lightweight refiner that can refine the rasterized glyphs generated by any FFG method to achieve style refinement. Throughout the paper, we refer to these FFG methods as the *base generator*. The refiner is agnostic of the base generator.
2. a few-shot adaptation method to adapt the refiner to target font styles by finetuning its layers for a few iterations. Unlike ISR, the adaptation method does not need to finetune the base generator, thus making it possible to apply refinement to a wide-range of base generators.
3. an overall loss which improves the quality of adaptation of the refiner to the target font styles.

2 Related Works

2.1 Image-to-Image Translation

Image-to-Image translation (I2I) refers to converting images from one domain to another domain using a mapping function while preserving the content of the image. I2I has several applications in computer vision such as image inpainting [24,26], denoising [5] and super-resolution [7,19]. Several CNN-based methods have been developed for I2I tasks with high-quality outputs being achieved by methods based on residual networks [36], generative adversarial networks (GANs) [10,38], attention mechanism [34], and multi-scale feature aggregation [4,18]. Recently, transformers [8,35] and diffusion models [21,25] have also been employed widely in I2I tasks outperforming the CNN-based methods. Few-shot image generation is an I2I task where the model combines the style of a few reference images with the content of the source image to produce the corresponding image in the target domain.

2.2 Few-Shot Font Generation

FFG methods generate glyphs in the target style using a set of reference glyphs. Most of the work in FFG has been done in Chinese font generation which has over 60K glyphs. [32] generates glyph images in target style using font manifolds learned from other fonts. LF-Font [22] extracts component-wise features of glyphs using a reference encoder and combines them to synthesize new glyphs, whereas MX-Font [23] uses multiple reference encoders to extract style and content features for each reference glyph. DG-Font [33] uses deformable convolutions to learn intricate font styles in an unsupervised framework. CF-Font [30] uses a content fusion module to fuse the content of multiple fonts and iterative style refinement to finetune the model to perform well on unseen fonts. FS-Font [27] uses multi-head attention to learn fine-grained style from the reference glyphs. Unlike rasterized glyphs, vector glyphs are represented as a sequence of commands or control points that can be converted to rasterized glyphs of any resolution. Dual Vector [20] utilizes CNN and distance fields to learn vector glyphs as a set of dual parts, which are closed Bezier curves complementary to each other. In DeepVecFont-v2 [31], reference glyphs are processed by CNN and transformers to generate raster and vector glyphs, respectively, followed by contour-refinement. In our work, we use different FFG methods as base generators and apply adaptive refiner on their output to generate rasterized glyphs that are perceptually closer to the target style.

2.3 Deformable Convolution

[6] proposed deformable convolutional layers that can expand their receptive field using learnable offsets, unlike convolutional layers with fixed kernels. The 2D offsets are learned as a function of the feature maps of preceding layers, thus allowing the deformations to be local and adaptive. The free-form deformation of the sampling grids helps deformable convolutions learn intricate styles better than standard convolutions. Deformable convolutions have been applied in object detection [3,6], image-to-image translation [33], and semantic segmentation [39]. DG-Font employed deformable convolutions to learn offsets from style encoding and used them to deform glyphs from one font style to another.

The refiner uses deformable convolutions to learn target style features that were missing or incorrectly generated by the base generator in a localized manner thus bringing it closer to the target style.

2.4 Spatial Transformer Network

STN is a neural network augmented with the spatial transformer module [14] and is capable of learning spatial transformations. It takes as input feature maps and outputs the parameters for affine or non-rigid transformations, which is then applied to the feature maps to generate spatially transformed features. Spatial transformer modules are differentiable, making STN end-to-end trainable. We employ STN in the refiner to learn global style features such as slant and thickness of target style conditioned on the content glyph.

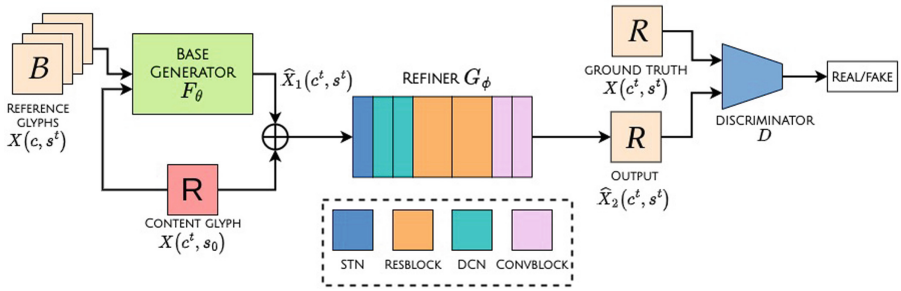


Fig. 1. Overview of the proposed pipeline consisting of a base generator, refiner and discriminator. Detailed description in Sect. 3.1

3 Methodology

A rasterized glyph with content $c \in \mathcal{C}$ and style s is denoted as $X(c, s)$. Our goal is to generate the glyph $X(c^t, s^t)$ as a function of target character c^t and target style s^t. The target style is supplied by the rasterized reference glyphs drawn by human designers for characters $c \in \mathcal{C}_{ref} \subset \mathcal{C}$. The target character c^t is supplied as a rasterized glyph $X(c^t, s_0)$ in some base style s_0 for which glyphs for all characters are available. A base generator F_θ, with parameters θ, takes as input the set of rasterized glyphs $\{X(c, s^t) : c \in \mathcal{C}_{ref}\}$ and $X(c^t, s_0)$, and gives as output $\hat{X}_1(c^t, s^t)$. For style refinement, we use a refiner model G_ϕ, with parameters ϕ, which processes $\hat{X}_1(c^t, s^t)$ locally using a fully convolutional architecture to produce a refined rasterized glyph $\hat{X}_2(c^t, s^t)$. The refined glyph $\hat{X}_2(c^t, s^t)$ contains the local style details which were missing or incorrectly generated in $\hat{X}_1(c^t, s^t)$ and is thus closer to the target style.

In this work, we use DG-font [33], CF-font [30] and DeepVecFont-v2 [31] as base generators. We discuss the data comprising the content and style in detail in Sect. 4.1.

3.1 Refiner

The refiner G_ϕ is a light-weight fully convolutional model consisting of spatial transformer (ST) module [14], deformable convolutions (DCN) [6], residual blocks and convolutional blocks. The detailed architecture is mentioned in Table 1. As shown in Fig. 1, the input to the refiner is the rasterized glyph $\hat{X}_1(c^t, s^t)$ generated by the base generator F_θ and the content glyph $X(c^t, s_0)$ in the base font concatenated channel-wise.

The first layer of the refiner is the ST module which applies affine transformations on the entire feature map to learn global features such as slant and stroke width. The transformed feature map is then passed to the DCN layers which applies convolutional operation at a local level to learn intricate style features such as serif and different stroke terminals. The output of DCN layers is processed by the residual blocks and convolutional layers before generating

the final output. The skip connections in the residual block allow a stable back-propagation within refiner layers avoiding the vanishing gradient problem. The output of the refiner is the fully refined rasterized glyph $\hat{X}_2(c^t, s^t)$ in the target style.

Table 1. Refiner Architecture

Layer	Kernel	Pad	Features	Norm	Activation
STN	–	–	2	–	–
DCN	7	3	16	IN	ReLU
DCN	7	3	16	IN	ReLU
Resblock	5	2	16	IN	LeakyReLU
Resblock	5	2	16	IN	LeakyReLU
Convblock	7	3	16	IN	LeakyReLU
Convblock	7	3	1	IN	LeakyReLU

STN: Spatial transformer network [14], DCN: Deformable convolutions [6], Resblock: Residual block, IN: Instance normalization [29], LeakyReLU with $\alpha = 0.3$

Refiner Pretraining. The refiner G_ϕ is pre-trained using a large dataset of rasterized glyphs in different font styles and for a given F_θ. The dataset consists of $\{\hat{X}_1(c^t, s^t), X(c^t, s^t), X(c^t, s_0) : c^t \in \mathcal{C}, s^t \in \mathcal{S}\}$, where $\mathcal{S}$ is a finite set of styles available for training. The rasterized glyph $\hat{X}_1(c^t, s^t)$ is obtained from F_θ and $X(c^t, s_0)$ from the base font s_0. $X(c^t, s^t)$ serves as the ground truth for the target style s^t. The refiner pre-training is carried out in a supervised setting, where G_ϕ learns to map the input glyph $\hat{X}_1(c^t, s^t)$ to the target $X(c^t, s^t)$ by minimizing the overall loss defined in Eq. (6) in Sect. 3.2. For adversarial loss in Eq. (6), we adopt the discriminator model defined in [33] and initialize it with pretrained weights obtained by training DG-Font as the base generator. We use the same discriminator weight initialization while pretraining the refiner for other base generators as well. Since we train the refiner to adapt to one font at a time, instead of the multi-class discriminator in [33], we use a single-class discriminator by keeping a single neuron in the last layer of the discriminator. Only the last two layers of the discriminator are re-trained during refiner pretraining. The discriminator classifies whether a glyph was generated by the refiner or not. We empirically observe that freezing the discriminator's initial layers, except for the last two, maintains its ability to discern real glyphs, stabilizing GAN training and allowing the refiner to focus on refining intricate stylistic details. Further, we propose to use a few-shot adaptation method to adapt G_ϕ to the target style s^t as follows.

Refiner Adaptation. For each target font to be generated, the human designers provide a set of rasterized reference glyphs $\{X(c, s^t) : c \in \mathcal{C}_{ref}\}$ in the target style s^t. We use these glyphs as ground truth to finetune the refiner parameters ϕ so that it adapts to the target style. Thus, for each target font s^t, we use the dataset $\{\hat{X}_1(c^t, s^t), X(c^t, s_0), X(c^t, s^t) : c^t \in \mathcal{C}_{ref}, s^t \in \bar{\mathcal{S}}\}$ to finetune the refiner parameters. Here, $\bar{\mathcal{S}}$ is the set of styles not seen during training i.e., $\bar{\mathcal{S}} \cap \mathcal{S} = \emptyset$. We use differential learning rate [13] for finetuning: different refiner layers are finetuned using different learning rate and beta values (parameters of Adam optimizer). Specifically, the initial layers comprising of ST module and DCN layers are finetuned using a lower β_1 value compared to the later layers. In the Adam optimizer, β_1 determines how quickly the optimizer updates the running average of gradients for a new batch of data. We empirically observe that a lower β_1 in the initial layers allows better adaption of the refiner to new font styles. The fine-tuned refiner is then used for refining other glyphs $\{\hat{X}_1(c^t, s^t) : c^t \in \mathcal{C} \backslash \mathcal{C}_{ref}\}$. During adaptation, the discriminator weights are frozen and only the refiner is finetuned. We observed that finetuning the discriminator during adaptation led to degradation in refiner performance.

3.2 Loss Terms

This subsection describes the loss functions used to train the refiner. $\hat{X}_2(c, s)$ is the refiner output and $X(c, s)$ is the ground truth (target glyph). We use five losses: (1) adversarial loss to generate realistic glyphs (2) perceptual loss to generate glyphs that are perceptually similar to the ground truth (3) L1 loss to generate glyphs that match the ground truth at pixel-level (4) offset loss to ensure that the model converges to a solution (5) reconstruction loss to train the model to reconstruct the ground truth.

Adversarial Loss: During adaptation, the refiner is trained to minimize the Wasserstein GAN loss with gradient penalty as defined in [33]. The refiner G_ϕ tries to trick the discriminator D by generating glyphs that are close to the target style. Gradient penalty is used to penalize the discriminator for large gradient norms which helps it converge [10].

$$\mathcal{L}_{adv} = \min_{G_\phi} \max_{D} \mathbb{E}[\log D(X(c, s)) + \log(1 - D(\hat{X}_2(c, s)))] \tag{1}$$

Perceptual Loss: We utilize the feature reconstruction loss component of perceptual loss defined in [16] which minimizes the L1 loss between the feature maps of the refined output and the ground truth obtained from a pretrained network. We use `relu1_2`, `relu2_2`, `relu3_3` and `relu4_3` layers of the VGG16 model to obtain the feature maps.

$$\mathcal{L}_{perc} = \sum_{p=1}^{P} ||h_p(\hat{X}_2(c, s)) - h_p(X(c, s))||_1 \tag{2}$$

where $h_p(.)$ refers to the layer p.

L1 Loss: We train the refiner to minimize the loss between the refined output and ground truth at pixel level. Since training generative models using L2 loss results in blurry outputs, we minimize L1 loss between the output and the target.

$$\mathcal{L}_{L1} = ||\hat{X}_2(c,s) - X(c,s)||_1 \tag{3}$$

Offset Loss: Deformable convolution used in G_ϕ can have a very large receptive field which makes it difficult for the model to converge to a solution. By minimizing the offset loss defined in [33], we ensure that the model converges during training.

$$\mathcal{L}_{off} = \frac{1}{N}\sum_{i=1}^{N} ||\text{offset}_i||_1 \tag{4}$$

where offset_i corresponds to the offset learned by ith deformable convolutional layer.

Reconstruction Loss: Reconstruction loss minimizes the pixel-wise loss between ground truth and the glyph generated by passing the ground truth through the refiner. This encourages the refiner to learn an identity transformation in case $\hat{X}_1(c,s)$ is the same as $X(c,s)$. The reconstruction loss term is

$$\mathcal{L}_{rec} = ||G_\phi(X(c,s)) - X(c,s)||_1 \tag{5}$$

Overall Loss: The overall loss terms for the refiner and the discriminator are

$$\begin{aligned} \mathcal{L}_{gen} &= \lambda_{L1} * \mathcal{L}_{L1} + \lambda_{perc} * \mathcal{L}_{perc} + \lambda_{adv} * \mathcal{L}_{adv,R} + \lambda_{rec} * \mathcal{L}_{rec} + \lambda_{off} * \mathcal{L}_{off} \\ \mathcal{L}_{disc} &= \lambda_{adv} * \mathcal{L}_{adv,D} + \lambda_{gp} * \mathcal{L}_{gp} \end{aligned} \tag{6}$$

where $\lambda_{L1}, \lambda_{perc}, \lambda_{adv}, \lambda_{off}, \lambda_{rec}$ and λ_{gp} are hyperparameters.

4 Experiments

In this section, we discuss the dataset preparation, implementation, and evaluation metrics used in this work, along with the comparison of our proposed model with state-of-the-art methods. All the experiments are conducted using a single NVIDIA Tesla V100 GPU. We observe that the refiner-based pipeline outperforms the state-of-the-art models on all the evaluation metrics in generating various font styles accurately.

4.1 Dataset Preparation

We collect a dataset of 2000 fonts in the truetype and opentype format. In each font, we use the 26 English alphabets {A,...Z} as the complete glyphset $\mathcal{C}$. We use B, A, S, Q as the set of reference glyphs $\mathcal{C}_{ref}$ [2]. We do not demonstrate results with other scripts and glyphs in this work but the proposed method can be easily extended to them. The dataset is split into two parts: training fonts $\mathcal{S}$ and testing fonts $\bar{\mathcal{S}}$. The training fonts consist of 1000 fonts selected randomly

from the dataset and are used to train the base generator F_θ and the refiner G_ϕ. A sans-serif font is chosen from the training fonts to be used as the base font s_0. The testing fonts, which comprise the remaining 1000 fonts in the dataset, are split into two parts: plain fonts and italic fonts to evaluate the refiner's ability to adapt to different kinds of fonts. Plain fonts primarily comprise of the non-italicized sans-serif and serif typefaces. Italic fonts include the italicized forms of the plain fonts along with other font styles.

4.2 Implementation Details

The hyperparameters comprising of the loss coefficients in Eq. (6) are tuned for optimal learning. λ_{L1} & λ_{perc} are higher to ensure high style accuracy. λ_{off} is lower to prevent over-constraining the deformable convolution layers and enable the refiner to adapt to new font styles. Reconstruction and adversarial losses are balanced with similar coefficients for effective learning. During pretraining, $\lambda_{L1} > \lambda_{perc}$ such that the loss function emphasizes accurate glyph content generation, while during adaptation, equal emphasis is given to style accuracy by setting $\lambda_{L1} = \lambda_{perc}$. We achieve similar performance with a range of hyperparameters values: $\lambda_{adv} = 1.0$, $10.0 \leq \lambda_{L1} \leq 50.0$, $5.0 \leq \lambda_{perc} \leq 10.0$, $0.5 \leq \lambda_{off} \leq 1.0$ and $\lambda_{rec} = 2.0$ following the relative order described above.

For refiner pretraining, we use $\lambda_{adv} = 1.0$, $\lambda_{L1} = 50.0$, $\lambda_{perc} = 5.0$, $\lambda_{off} = 1.0$ and $\lambda_{rec} = 2.0$. For refiner adaptation, we use $\lambda_{L1} = 10.0$, $\lambda_{perc} = 10.0$, $\lambda_{off} = 0.5$ and $\lambda_{rec} = 2.0$. We use the Adam optimizer [17] with $lr = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to train the refiner and discriminator. During adaptation, we use Adam with a differential learning rate for each refiner layer as described in 3.1. The spatial transformer and deformable convolutions, which are closer to the input, are set to $lr = 1e^{-3}$, $\beta_1 = 0$ and $\beta_2 = 0.999$. The rest of the layers are set to $lr = 1e^{-3}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The refiner is trained with a batch size of 32 for 50 epochs which takes nearly 6 h on our system. During adaptation, we finetune the refiner to learn one font at a time using 1000 training steps. In each training step, the model is trained in a supervised way using the rasterized reference glyphs with a batch size of 4. The overall adaptation process takes about 1 min per target style.

4.3 Evaluation Metrics and Comparison with State-of-the Art Models

We evaluate the refiner based on pixel-level and perceptual metrics. The pixel-level metrics are L1 loss and Structural Similarity Index Measure (SSIM), which compute the pixel-wise difference between the refined output and ground truth. The perceptual metrics are Frechet Inception Distance (FID) [11] and Learned Perceptual Image Patch Similarity (LPIPS) [37], which uses a pretrained neural network to measure the perceptual similarity between the refined output and ground truth. We apply the refiner on 3 base generators: DG-Font, CF-Font, and DeepVecFont-v2. We evaluate their performance in two cases: (1) with adaptive

refiner based style refinement (2) without style refinement. We modify the DG-Font network to use 4 reference glyphs instead of 1 and a fixed base font. For CF-Font, we fix 4 reference glyphs instead of 16, but the base font is chosen dynamically using the content fusion technique. We train DG-Font and CF-Font on our dataset after modifying their networks whereas for DeepVecFont-v2, we use pre-trained weights.

The outputs of the base generators and adapted refiner for various font styles are illustrated in Fig. 2a and the quantitative results are shown in Table 2. After applying the adaptive refiner on the base generators, there is consistent improvement in L1 loss and SSIM index. However, the LPIPS and FID scores sometimes show degradation (indicated by red bounding boxes), likely due to reduced sharpness in the refined outputs compared to the base generator. These metrics (LPIPS and FID) compare the difference between feature maps of the rasterized glyphs obtained using pretrained networks (e.g. VGG16), and may emphasize sharpness of the glyphs over style accuracy [9], resulting in the reduced score. Despite this, our method achieves scores that are comparable to the best in

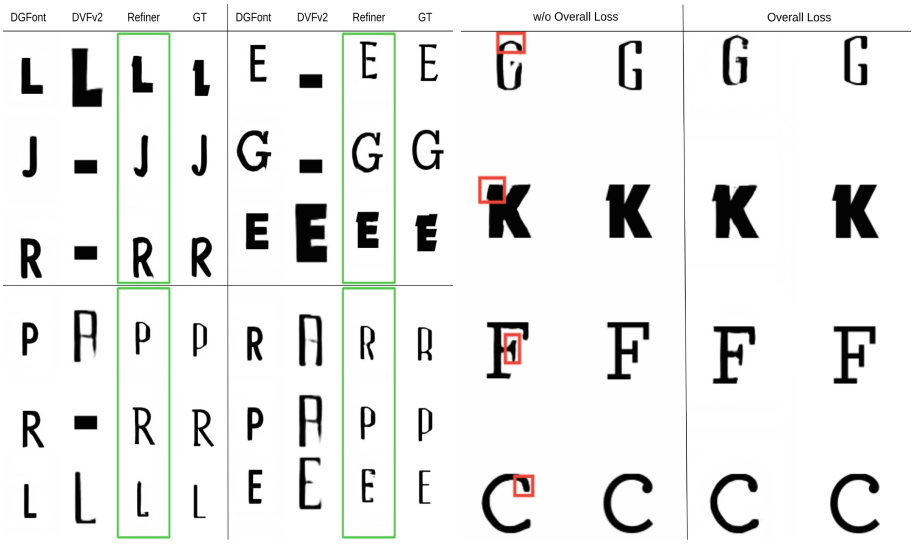


Fig. 2. (a) Comparison of the output of base generators and adapted refiner: DGFont (col 1), Deepvecfont-v2 (col 2), adapted refiner (col 3) and the ground truth (col 4). In col 2, - denotes invalid fonts for Deepvecfont-v2 (For Deepvecfont-v2, a font style is valid if its vector glyphs have sequence length less than a certain threshold. It cannot generate invalid font styles.). (b) Refiner adaptation without overall loss (Left) & with overall loss (Right): One can observe that the refiner output (col 1) in the left image contains local incongruencies (marked by red boxes), which are nearly eliminated in the refiner output (col 1) in the right image. Thus, the overall loss improves refiner adaptation.

Table 2. Comparison with state-of-the-art methods

Font	Model	L1 ↓	LPIPS ↓	FID ↓	SSIM ↑
Plain	DGFont	0.0603	0.1014	0.0390	0.8674
	CFFont	0.0756	0.1404	0.0271	0.8618
	DVFont-v2	0.0807	0.0993	0.0169	0.8220
	DGFont+Refiner	**0.0381**	**0.0689**	**0.0090**	**0.9082**
	CFFont+Refiner	0.0547	0.1178	0.0162	0.8899
	DVFont-v2+Refiner	0.0750	0.1020	0.0231	0.8385
Italic	DGFont	0.0768	0.1127	0.0582	0.8502
	CFFont	0.0858	0.1491	0.0281	0.8462
	DVFont-v2	0.1031	0.1119	0.0358	0.7880
	DGFont+Refiner	**0.0465**	**0.0744**	**0.0100**	**0.8974**
	CFFont+Refiner	0.0672	0.1323	0.0649	0.8788
	DVFont-v2+Refiner	0.0855	0.1042	0.0287	0.8210

Fig. 3. Refiner output with DGFont [33], CFFont [30] and DeepVecFont-v2 [31] as base generators. BG: Base Generator output, UR: Unadapted refiner output, AR: Adapted refiner output, GT: Ground truth. It can be observed that the output of the adapted refiner closely matches the ground truth.

these metrics, underscoring its effectiveness. Qualitative results in Fig. 3 further demonstrate the efficacy of our approach in terms of achieving better style accuracy compared to the base generators. The base generator output (col 1) is markedly different from the ground truth (col 4). The adapted refiner (col 3) rectifies both local (e.g. serifs) and global (e.g. italics) style inconsistencies, producing glyphs that closely resemble the ground truth. This indicates that the adaptive refiner enhances the base generator's output, aligning it more closely to the target style by addressing style inconsistencies effectively.

5 Ablation Studies

5.1 Effect of Different Loss Terms

We analyze the effect of each loss term during refiner adaptation after pretraining it with the overall loss defined in Eq. (6). We perform adaption of the pretrained refiner using different loss terms, starting from the L1 loss and then incrementally adding new loss terms, followed by evaluation of the adapted refiner. The results are shown in Table 3. The L1 loss and SSIM index consistently improve with the addition of each loss term, with the best score being achieved using the overall loss in Eq. (6). It is observed that dropping $\mathcal{L}_{rec}$ from the overall loss during adaptation gives comparable performance (marked in blue). The LPIPS and FID scores achieved using the overall loss are best or comparable (marked in blue) to the best score (marked in black) achieved using other losses. As discussed in Sect. 4.3, these metrics compare difference between the rasterized glyphs at the feature level, and may give emphasis to sharpness of the glyphs over their style accuracy [9], leading to comparable scores. Figure 2b clearly demonstrates the improvement in style accuracy of the refined output compared to the output of the base generator, underscoring the efficacy our method.

5.2 Effect of Different Layers

We analyze the effect of the Spatial Transformer (ST) and Deformable Convolutio (DCN) layers in the refiner's performance. We pretrain and adapt different refiner models using the overall loss defined in Eq. (6), starting with only convolutional and residual blocks and then incrementally adding ST and DCN layers, followed by evaluation of the adapted refiner. The results are shown in Table 4. It can be seen that both the pixel-wise metrics (L1 loss and SSIM index) and perceptual metrics (LPIPS and FID scores) improve consistently with the addition of ST and DCN layers, with the best score being achieved using the proposed refiner model which contains one ST and two DCN layers. Thus, ST and DCN layers enhance the refiner's ability to adapt to various font styles.

Table 3. Effect of different loss terms on adaptation to font styles

Model	Loss	L1 ↓	LPIPS ↓	FID ↓	SSIM ↑
PLAIN FONTS					
DGFont	–	0.0603	0.1014	0.0390	0.8674
Refiner	$1.0\mathcal{L}_{L1}$	0.0393	0.0808	0.0160	0.9046
	$10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc}$	0.0385	0.0699	**0.0064**	0.9084
	$1.0\mathcal{L}_{adv} + 10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc}$	0.0382	**0.0685**	0.0121	0.9085
	$1.0\mathcal{L}_{adv} + 10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc} + 0.5\mathcal{L}_{off}$	**0.0381**	**0.0686**	0.0101	**0.9088**
	$1.0\mathcal{L}_{adv} + 10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc} + 0.5\mathcal{L}_{off} + 2.0\mathcal{L}_{rec}$	**0.0381**	**0.0689**	**0.0090**	**0.9086**
ITALIC FONTS					
DGFont	–	0.0768	0.1127	0.0582	0.8502
Refiner	$1.0\mathcal{L}_{L1}$	0.0489	0.0915	0.0172	0.8913
	$10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc}$	0.0473	0.0767	**0.0075**	0.8963
	$1.0\mathcal{L}_{adv} + 10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc}$	0.0475	0.0778	0.0085	0.8949
	$1.0\mathcal{L}_{adv} + 10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc} + 0.5\mathcal{L}_{off}$	0.0483	0.0792	**0.0082**	0.8944
	$1.0\mathcal{L}_{adv} + 10.0\mathcal{L}_{L1} + 10.0\mathcal{L}_{perc} + 0.5\mathcal{L}_{off} + 2.0\mathcal{L}_{rec}$	**0.0465**	**0.0744**	0.0100	**0.8974**

Table 4. Effect of ST and DCN layers on adaptation to font styles

Refiner Model	L1 ↓	LPIPS ↓	FID ↓	SSIM ↑
PLAIN FONTS				
2 ConvBlock + 3 ResBlock + 2 ConvBlock	0.0456	0.0934	0.0149	0.8965
1 ST + 2 ConvBlock + 3 ResBlock + 2 ConvBlock	0.0412	0.0783	0.0145	0.9021
2 DCN + 3 ResBlock + 2 ConvBlock	0.0393	0.0737	0.0143	0.9057
1 ST + 2 DCN + 3 ResBlock + 2 ConvBlock **(Proposed)**	**0.0381**	**0.0689**	**0.0090**	**0.9082**
ITALIC FONTS				
2 ConvBlock + 3 ResBlock + 2 ConvBlock	0.0545	0.1043	0.0259	0.8855
1 ST + 2 ConvBlock + 3 ResBlock + 2 ConvBlock	0.0489	0.0819	0.0168	0.8924
2 DCN + 3 ResBlock + 2 ConvBlock	0.0487	0.0853	0.0181	0.8927
1 ST + 2 DCN + 3 ResBlock + 2 ConvBlock **(Proposed)**	**0.0465**	**0.0744**	**0.0100**	**0.8974**

6 Conclusion

In this work, we propose an adaptive refiner model capable of transforming rasterized glyphs into the target font style through a few-shot adaptation process. The adaptive refiner improves the style accuracy of FFG models, which often fail to generate diverse font styles accurately, by refining the glyphs these models produce. Our model demonstrates its ability to adapt to various font styles and outperform FFG models in addressing and correcting local style inconsistencies within the generated glyphs. This is achieved through the use of spatial transformers and deformable convolutions, which learn the global and local transformations essential for refining the output of FFG models to achieve superior

style accuracy. Extensive experiments validate the effectiveness of our proposed method in enhancing the style accuracy of FFG models, specifically for English alphabets. However, the approach is not limited to English and can be easily extended to other scripts and glyphs. Further exploration is needed to investigate its effectiveness across a broader spectrum of scripts. Our findings pave the way for more sophisticated approaches in font generation, offering promising insights for the development of more versatile and accurate font adaptation models. This work provides a robust solution to the current limitations of FFG models and encourages further research to validate and extend our method to diverse writing systems.

Acknowledgement. This research received funding support from Monotype Solutions India Pvt. Ltd. Authors are also grateful to Emilios Theofanous and Mrinal Mouza of Monotype for many insightful discussions.

References

1. Zi2zi: Master chinese calligraphy with conditional adversarial networks. https://github.com/kaonashi-tyc/zi2zi
2. Baluja, S.: Learning typographic style: from discrimination to synthesis. Mach. Vis. Appl. 551–568 (2017). https://doi.org/10.1007/s00138-017-0842-6
3. Bertasius, G., Torresani, L., Shi, J.: Object detection in video with spatiotemporal sampling networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11216, pp. 342–357. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01258-8_21
4. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 833–851. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_49
5. Cho, S.J., Ji, S.W., Hong, J.P., Jung, S.W., Ko, S.J.: Rethinking coarse-to-fine approach in single image deblurring. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4641–4650 (2021)
6. Dai, J., et al.: Deformable convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 764–773 (2017)
7. Dong, C., Loy, C.C., He, K., Tang, X.: Image super-resolution using deep convolutional networks. IEEE Trans. Pattern Anal. Mach. Intell. **38**(2), 295–307 (2015)
8. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale (2021)
9. Ghildyal, A., Liu, F.: Attacking perceptual similarity metrics. arXiv preprint arXiv:2305.08840 (2023)
10. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein GANs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
11. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 (2017)

13. Ivanov, S.: "differential learning rates" (2017). https://blog.slavv.com/differential-learning-rates-59eff5209a4f. Accessed 20 Nov 2017
14. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial Transformer Networks. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
15. Jiang, Y., Lian, Z., Tang, Y., Xiao, J.: Dcfont: an end-to-end deep Chinese font generation system. In: SIGGRAPH Asia 2017 Technical Briefs, pp. 1–4 (2017)
16. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Lee, J.H., Han, M.K., Ko, D.W., Suh, I.H.: From big to small: multi-scale local planar guidance for monocular depth estimation. arXiv preprint arXiv:1907.10326 (2019)
19. Lim, B., Son, S., Kim, H., Nah, S., Mu Lee, K.: Enhanced deep residual networks for single image super-resolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 136–144 (2017)
20. Liu, Y.T., Zhang, Z., Guo, Y.C., Fisher, M., Wang, Z., Zhang, S.H.: Dualvector: unsupervised vector font synthesis with dual-part representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14193–14202 (2023)
21. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Repaint: Inpainting using denoising diffusion probabilistic models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11461–11471 (2022)
22. Park, S., Chun, S., Cha, J., Lee, B., Shim, H.: Few-shot font generation with localized style representations and factorization. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 2393–2402 (2021)
23. Park, S., Chun, S., Cha, J., Lee, B., Shim, H.: Multiple heads are better than one: few-shot font generation with multiple localized experts. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13900–13909 (2021)
24. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536–2544 (2016)
25. Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D.J., Norouzi, M.: Image super-resolution via iterative refinement. IEEE Trans. Pattern Anal. Mach. Intell. **45**(4), 4713–4726 (2022)
26. Suvorov, R., et al.: Resolution-robust large mask inpainting with Fourier convolutions. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2149–2159 (2022)
27. Tang, L., et al.: Few-shot font generation by learning fine-grained local styles. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7895–7904 (2022)
28. Thamizharasan, V., et al.: Vecfusion: vector font generation with diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7943–7952 (2024)
29. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: the missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)

30. Wang, C., Zhou, M., Ge, T., Jiang, Y., Bao, H., Xu, W.: CF-font: content fusion for few-shot font generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1858–1867 (2023)
31. Wang, Y., Wang, Y., Yu, L., Zhu, Y., Lian, Z.: Deepvecfont-v2: exploiting transformers to synthesize vector fonts with higher quality. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18320–18328 (2023)
32. Xie, H., Fujita, Y., Miyata, K.: Learning perceptual manifold of fonts. arXiv preprint arXiv:2106.09198 (2021)
33. Xie, Y., Chen, X., Sun, L., Lu, Y.: DG-font: deformable generative networks for unsupervised font generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5130–5140 (2021)
34. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5505–5514 (2018)
35. Zamir, S.W., Arora, A., Khan, S., Hayat, M., Khan, F.S., Yang, M.H.: Restormer: efficient transformer for high-resolution image restoration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5728–5739 (2022)
36. Zhang, K., Li, Y., Zuo, W., Zhang, L., Van Gool, L., Timofte, R.: Plug-and-play image restoration with deep denoiser prior. IEEE Trans. Pattern Anal. Mach. Intell. **44**(10), 6360–6376 (2021)
37. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–595 (2018)
38. Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B., Fu, Y.: Image super-resolution using very deep residual channel attention networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 294–310. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_18
39. Zhu, X., Hu, H., Lin, S., Dai, J.: Deformable convnets v2: more deformable, better results. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9308–9316 (2019)

Controllable 3D Object Generation with Single Image Prompt

Jaeseok Lee and Jaekoo Lee(✉)

College of Computer Science, Kookmin University, Seoul, Korea
jaekoo@kookmin.ac.kr

Abstract. Recently, the impressive generative capabilities of diffusion models have been demonstrated, producing images with remarkable fidelity. Particularly, existing methods for the 3D object generation tasks, which is one of the fastest-growing segments in computer vision, predominantly use text-to-image diffusion models with textual inversion which train a pseudo text prompt to describe the given image. In practice, various text-to-image generative models employ textual inversion to learn concepts or styles of target object in the pseudo text prompt embedding space, thereby generating sophisticated outputs. However, textual inversion requires additional training time and lacks control ability. To tackle this issues, we propose two innovative methods: (1) using an off-the-shelf image adapter that generates 3D objects without textual inversion, offering enhanced control over conditions such as depth, pose, and text. (2) a depth conditioned warmup strategy to enhance 3D consistency. In experimental results, ours show qualitatively and quantitatively comparable performance and improved 3D consistency to the existing text-inversion-based alternatives. Furthermore, we conduct a user study to assess (i) how well results match the input image and (ii) whether 3D consistency is maintained. User study results show that our model outperforms the alternatives, validating the effectiveness of our approaches. Our code is available at GitHub repository: https://github.com/Seooooooogi/Control3D_IP/

Keywords: 3D Generative Models · Neural Radiance Fields · Diffusion Models

1 Introduction

Generative models are widely used in real-world industrial applications, including creating artworks [33], video generation [25,39], automatic colorization [14,45], virtual try-on [48], and 3D computer graphics [2,3,20,23,26,27]. Especially, 3D object generation tasks have significant demand in computer vision industries such as Artificial Reality (AR), Virtual Reality (VR), and Gaming. Conventionally, generating 3D objects has been a delicate and intricate process that requires experts well-versed in 3D graphics tools with a considerable amount

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 222–238, 2025.
https://doi.org/10.1007/978-3-031-78172-8_15

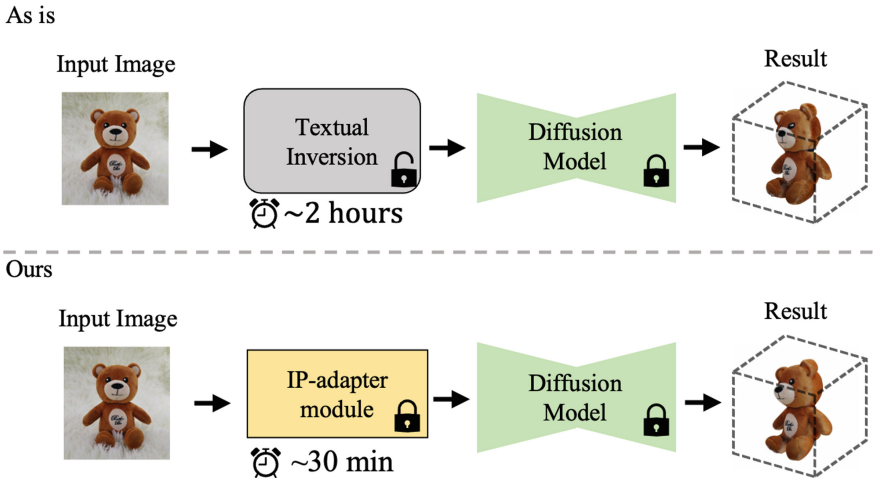


Fig. 1. Comparison between (a) existing method using text-guided priors via textual inversion and (b) our method that employs a single image prompt.

of time. However, recent breakthroughs have introduced 3D generative models (e.g., text-to-3D model and image-to-3D model) that allow even non-experts to effortlessly produce 3D objects.

Especially, several methods [20,27] have achieved significant advancements in image-to-3D generation by using text-guided priors and textual inversion [6], which is subsequently used in text-to-image diffusion model. However, as shown in Fig. 1, textual inversion requires additional time and cost to get a new prompt because it needs training process to optimize pseudo text prompt.

To address these issues, we propose the following contributions.

- We propose a controllable image prompt score distillation sampling (called SDS) method that uses a single image as a prompt without textual inversion to generate novel views of 3D object.
- Our method leverages depth estimated from the 3D object as a condition for ControlNet [46] to prevent the degradation of 3D consistency, and employs a depth conditioned warmup strategy to alleviate instability of depth in early training epochs.
- We experimentally show that our proposed method generates diverse, controllable 3D objects using a given single image and optional prompts such as depth, pose, and text. Our method quantitatively and qualitatively outperforms alternative approaches.
- We also conduct a user study that participants score two evaluation metrics: fidelity and 3D consistency. Our method exhibits superior performance compared to SOTA methods.

2 Related Work

Neural 3D Representations. Neural Radiance Fields (NeRF) [22] trains deep neural network with sparse input views and camera poses to optimize the inherent continuous volumetric scene function, synthesizing novel views of given scenes. Although NeRF has capability of 3D reconstruction task, optimization process is slow and hard to get high-resolution results. To overcome computational cost, Instant-NGP [24] uses smaller neural networks, leverages multi resolution hash grids without sacrificing quality. On the other hand, Deep Marching Tetrahedra (DMTet) [37] leverages novel hybrid implicit-explicit 3D representations, achieving high resolution, finer geometric details with fewer artifacts with efficient memory consumption. In here, we utilize Instant-NGP and DMTet for generate 3D object from a single image, aiming for higher quality results while maintaining memory efficiency.

Diffusion Models. In computer vision tasks such as inpainting [19,34], image editing [21,40], super-resolution [15,35], image generation [29,33] and video generation [25,39], diffusion models are gaining popularity due to great result quality. Diffusion models train neural networks to denoise images blurred with Gaussian noise and to reverse the diffusion process [9]. Especially, Stable diffusion [33], which is open source, achieved great success with datasets consisting of large scale text-image pairs [36]. Subsequent work, ControlNet, leverages the expressiveness of the Stable diffusion model, expanding it to incorporate a variety of conditional priors such as sketches, depth, and human poses [46]. Gal, Rinon *et al.* [6] introduce textual inversion, which optimizes a pseudo text prompt by using only 3–5 images of a user-provided concept to utilize the expressiveness of Stable diffusion model. However, it requires additional training time and usually needs several images. To overcome these shortcomings, IP-Adapter [44] proposes a lightweight adapter to directly input an image prompt into a pretrained text-to-image diffusion model, thereby facilitating multi-modal image generation without additional training time.

3D Generative Models. By integrating NeRF and diffusion models, several approaches aim to train 3D generative models, either by using text [3,16,26] or a single image [20,27]. For text-to-3D generation, DreamFusion [26] proposes score distillation sampling to guide the training of NeRF models using a pretrained text-to-image diffusion model. Although DreamFusion successfully achieved first text-to-3D generative task with pretrained diffusion models, it fails to generate high-resolution objects, requiring up to one day to train a single 3D object. To produce high-resolution 3D objects, several methods are proposed. Magic3D [16] integrates DMTet [37] during the training process. Fantasia3D [3] disentangles geometry and appearance to generate high-fidelity 3D objects that closely align with real graphics rendering. For image-to-3D generation task, RealFusion [20] employs textual inversion from an image to derive custom tokens for training a 3D generative model instead of text prompts. On the other hand, Zero-1-to-3 [18] finetunes the Stable diffusion model to simultaneously input an image and geometry-related camera pose priors, facilitating the synthesis of images

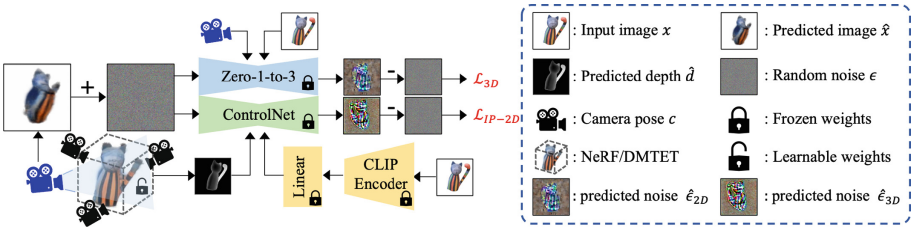


Fig. 2. Overall architecture of our proposed 3D generative model. During training, our method iteratively use the following stages: (i) controllable image prompt score distillation sampling, and (ii) depth conditioned warmup strategy.

from specific 3D viewpoints. Magic123 [27] combines the Stable diffusion model with textual inversion and Zero-1-to-3 simultaneously. Despite RealFusion and Magic123 utilize textual inversion for single image, these methods often fail to capture fine details of given images and requires additional training time.

3 Methods

The main goal of our method is to synthesize a 3D object without relying on text prompts obtained by textual inversion, instead directly using controllable image prompts (e.g., only image or with optional conditions). To achieve this goal, as shown in Fig. 2, we introduce *controllable image prompt score distillation sampling* and *depth conditioned warmup strategy*. We build the proposed model, which consists of two main components, using the Magic123 [27] architecture as the baseline. The proposed model performs alternately two stage training process (i.e., coarse-to-fine training). During the 1st stage for training coarse grained representation, we train NeRF. In detail, initialized NeRF model with learnable parameters θ, predicts the volume density σ and color c of each pixel given a random camera pose c.

$$(c, \sigma) = NeRF(\theta; c) \tag{1}$$

By computing each pixel's volume density and color, NeRF predicts image $\hat{x} \in \mathbb{R}^{H \times W \times 3}$ and depth $\hat{d} \in \mathbb{R}^{H \times W \times 1}$.

When the 1st training stage finished, we finetune the NeRF with DMTet to represent the target 3D object at high resolution during 2nd stage for training fine grained representation. Unlike NeRF which has entangled geometry and texture representations, DMTet has disentangled geometry and texture representations. For geometry representation, DMTet uses a deformable tetrahydral grid (V_T, T) [37], where tetrahydral grid denotes T and its vertices denote V_T. DMTet represents the 3D object using a Signed Distance Function (SDF) s at vertex $v_i \in V_T$ and a triangle deformation vector Δv_i. In here, s is initialized from the

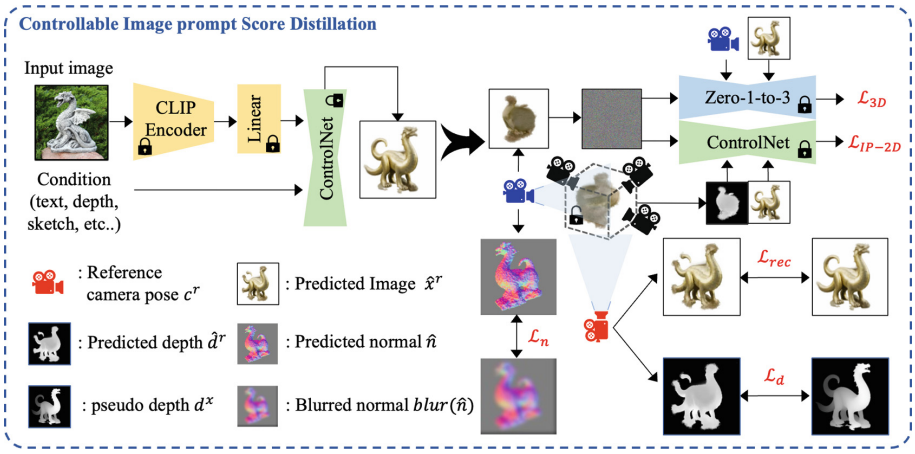


Fig. 3. An overview of controllable image prompt score distillation sampling. See details on Sect. 3.1.

NeRF, while triangle deformation vector Δv_i is initialized as zero. For texture representation, DMTet uses a neural color field, as mentioned in Magic3D [16], is also initialized from the NeRF.

3.1 Controllable Image Prompt Score Distillation Sampling

As shown in Fig. 3, a single input image x, optionally including various conditions such as text, depth, sketch, is embedded into a controllable image prompt through the well-known pretrained Image Prompt Adapter (IP-Adapter) [44], which consists of the pretrained CLIP [28] image encoder and the single linear layer. The embedded controllable image prompt and depth $\hat{d}$, obtained from NeRF/DMTet, are then injected as conditions into a stable diffusion model like ControlNet [46] to predict $\hat{\epsilon}_{2D}$.

The $\hat{x}$ obtained from NeRF/DMTet is added time-dependent noise ϵ to produce a noisy latent $\hat{x}_t$. $\hat{x}_t$ is then fed into the ControlNet [46]. Consequently, ControlNet outputs $\hat{\epsilon}_{2D}$, corresponding to the random diffusion timestep t. The loss function $\mathcal{L}_{IP-2D}$ for 2D score distillation sampling is as follows:

$$\mathcal{L}_{IP-2D} = \mathbb{E}_{t,\epsilon}\left[w(t)(\hat{\epsilon}_{2D}(\hat{z}_t; x, \hat{d}, t) - \epsilon)\frac{\partial x}{\partial \theta}\right] \tag{2}$$

where $w(t)$ represents the time-dependent weighting function. By directly injecting input image into the stable diffusion model, the 2D score distillation sampling loss guides NeRF/DMTet to adhere to the input image from any camera pose.

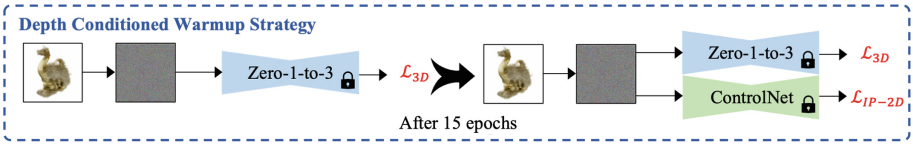


Fig. 4. An overview of depth conditioned warmup strategy. See details on Sect. 3.2.

Similar to ControlNet, Zero-1-to-3 predicts $\hat{\epsilon}_{3D}$ when given $\hat{x}_t$ as input. The loss function $\mathcal{L}_{3D}$ for 3D score distillation sampling is as follows:

$$\mathcal{L}_{3D} = \mathbb{E}_{t,\epsilon}\left[w(t)(\hat{\epsilon}_{3D}(\hat{z}_t; x, c, t) - \epsilon)\frac{\partial x}{\partial \theta}\right] \tag{3}$$

Although $\mathcal{L}_{3D}$ serves the same purpose as $\mathcal{L}_{IP-2D}$, Magic123 [27] found that training with $\mathcal{L}_{3D}$ can lead NeRF/DMTet to have better 3D consistency.

Additionally, a primary requirement of image-to-3D generative models is to reconstruct input image from a reference camera pose c^r. Given the input image x masked with $\mathcal{M}$, which is acquired by a dense prediction transformer [30], and the predicted image $\hat{x^r}$ masked with $\mathcal{M}^r$, we perform the reconstruction by comparing x and $\hat{x^r}$. The reconstruction loss $\mathcal{L}_{rec}$ is as follows:

$$\mathcal{L}_{rec} = \|\mathcal{M} \odot (x - \hat{x}^r)\|_2^2 + \|\mathcal{M} - \mathcal{M}^r\|_2^2 \tag{4}$$

where $\odot$ is element-wise product. Xu, Dejia *et al.* [43] introduce that reconstructing 3D object from 2D images often results in flat geometry. To alleviate this issue, we incorporate a monocular depth regularization loss $\mathcal{L}_d$, inspired by Magic123 [27]. Given predicted depth $\hat{d}^r$ from the reference camera pose c^r and the input image's pseudo depth d^x, estimated by pretrained monocular depth estimator [31], the loss function $\mathcal{L}_d$ is calculated using the negative Pearson correlation between $\hat{d}^r$ and d^x, where are masked with $\mathcal{M}$.

$$\mathcal{L}_d = \frac{1}{2}\left[1 - \frac{Cov(d^x \odot \mathcal{M}, \hat{d}^r \odot \mathcal{M})}{\sigma(d^x \odot \mathcal{M}), \sigma(\hat{d}^r \odot \mathcal{M})}\right] \tag{5}$$

The normal smoothness loss function $\mathcal{L}_n$ is designed to ensure smooth normals for the 3D object, as generated 3D objects often exhibit noisy artifacts on their surfaces. Specifically, following RealFusion [20], we compare the normals $\hat{n} \in \mathbb{R}^{H \times W \times 3}$ both before and after applying a Gaussian filter *blur*. Then the loss function $\mathcal{L}_n$ is defined as follows:

$$\mathcal{L}_n = \|\hat{n} - stopgrad(blur(\hat{n}, k))\|_2^2 \tag{6}$$

where *stopgrad* is stop-gradient operation. We use a kernel size $k = 9$ for the Gaussian filter.

Ultimately, we use the following total loss $\mathcal{L}_{total}$ as follows:

$$\mathcal{L}_{total} = \lambda_{IP-2D}\mathcal{L}_{IP-2D} + \lambda_{3D}\mathcal{L}_{3D} + \lambda_d\mathcal{L}_d + \lambda_n\mathcal{L}_n + \mathcal{L}_{rec} \tag{7}$$

3.2 Depth Conditioned Warmup Strategy

Typically, text-to-image diffusion models suffer from 3D inconsistencies due to insufficient 3D field information [26]. To mitigate this issue, we integrate the depth $\hat{d}$ obtained from NeRF/DMTet model as a condition for the ControlNet. As shown in Fig. 4, ControlNet [46] injecting depth $\hat{d}$ introduce a 3D prior into the NeRF/DMTet, thereby enhancing 3D consistency. However, $\hat{d}$ tends to be unstable in early epochs. Therefore, for the initial 15 epochs, we only use the Zero-1-to-3, exploiting on its enhanced geometric capabilities during the 1st stage of training. After 1st stage, we use both ControlNet and Zero-1-to-3 for 2nd stage of training.

Consequently, depth conditioned warmup strategy optimizes a unified 3D field, ensuring that the image synthesized from any viewpoint aligns with high probabilities as evaluated by Zero-1-to-3 by distilling pretrained diffusion models with a depth condition to refine the 3D consistency.

4 Experiments

4.1 Implementation Details

We use Instant-NGP [24] as the NeRF backbone model, consisting of three layers with 64 hidden dimensions, which is same as [26,27]. We integrates depth-conditioned ControlNet v1.1 with Stable Diffusion v1.5 and the pretrained Zero-1-to-3 model (100, 500 iteration checkpoint). For the IP-Adapter, we utilize a vanilla model which extracts 4 tokens from the input image. We maintain the same hyperparameters, including the elevation angle for the input image, for all experiments.

Datasets. We conduct experiments on the publicly available NeRF4 [22], Realfusion15 (RF15) [20], Google Scanned Objects (GSO) [5], and Common Objects in 3D (CO3D) [32] datasets. The NeRF4 dataset includes one simple object (mic), two objects with complex geometries (ficus, drums), and one challenging case (chair) where the input image is from the back view. The RealFusion15 dataset comprises real-world photos, including bananas, Barbie cake, bird sparrow, blue bird, cactus, cat statue, colorful teapot, fish, metal dragon statue, microphone, stone dragon statue, teddy bear, two cherries, two donuts, and watercolor horse. The GSO dataset includes 3D scans of common household items, providing five views for each object. We used single images of five objects: squirrel, school bus, hammer, dog, and shark. The CO3D dataset offers multi-view images of real-world objects from 50 MS-COCO [17] categories in the form of video frames. For our experiments, we used four scenes: bench, car, bicycle, and hydrant.

Fig. 5. Qualitative image-to-3D generation performance comparison with SOTA alternatives (Dreamfusion [26], RealFusion [20], LGM [41], LRM [11], Zero-1-to-3 [18], Magic123 [27]) on realfusion15 [20] and NeRF4 [22] datasets. More detailed results can be found on our GitHub.

Fig. 6. Comparison of qualitative image-to-3D generation performance with the previous methods (Dreamfusion [26], RealFusion [20], LGM [41], LRM [11], Zero-1-to-3 [18], Magic123 [27]) on GSO [5] and CO3D [32] datasets. More detailed results can be found on our GitHub.

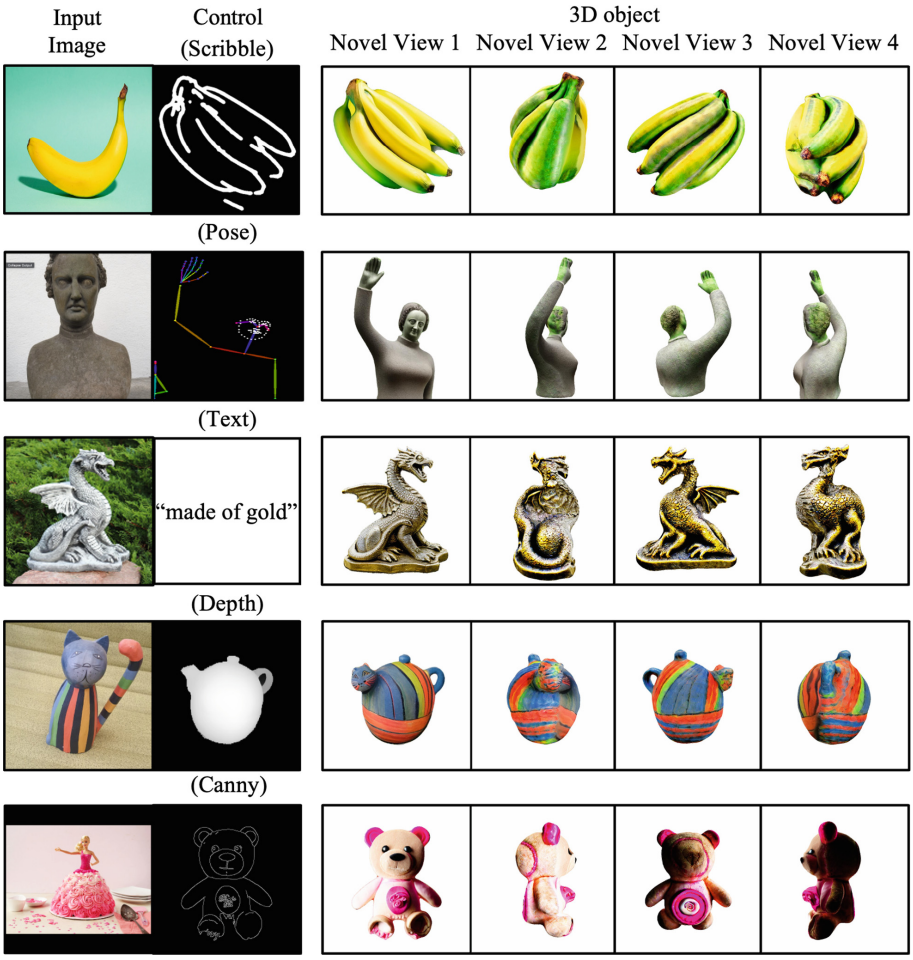


Fig. 7. Various controllable 3D synthesis with image prompt and optional conditions.

4.2 Quantitative Results

We perform quantitative experiments using contrastive language-image pre-training-similarity (CLIP-similarity) [20,28] and adjacent learned perceptual image patch similarity (A-LPIPS) [10] as evaluation metrics for 3D object generation. CLIP-similarity measures the consistency between the rendered image from evenly spaced azimuths and the category of its corresponding reference image. A-LPIPS is derived from the learned perceptual image patch similarity (LPIPS) metric [47], predicted on the notion that two images from adjacent viewpoints should be perceptually similar if the 3D object is consistent. To evaluate 3D consistency, we measure the average A-LPIPS between images adjancent in viewpoint, maintaining the same elevation angle as used in the CLIP-similarity

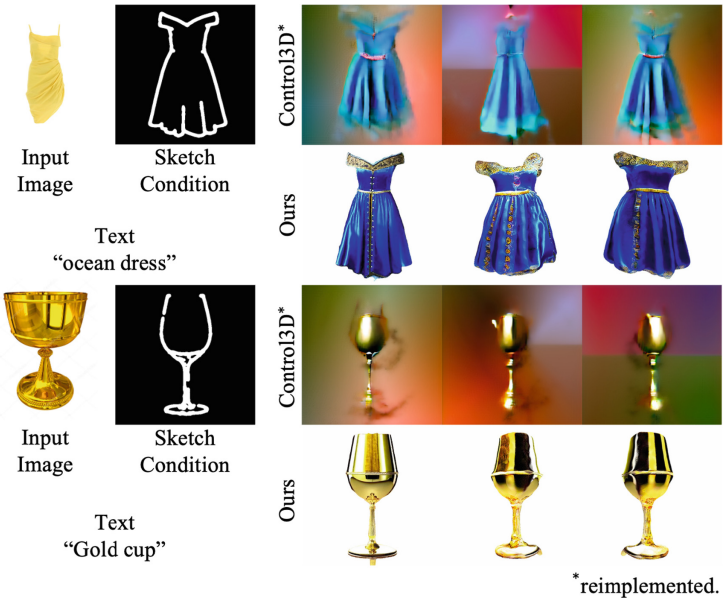


Fig. 8. Qualitative comparison with Control3D [4] and ours on sketch condition. Note that Control3D takes sketch and text as input, not image.

assessment. For A-LPIPS measurements, we employ neural network backbones such as Alex [13], and VGG [38], as referenced in previous work [10].

As shown in Table 1, quantitative experiments compare our method with the state-of-the-art (SOTA) alternatives described in [11,27,41]. We verify the superiority of our method, which relies solely on image prompts without textual inversion. In practice, the baseline having textual inversion needs to learn a pseudo text prompt from the input image, which requires an additional 1–2 hours of training time. In contrast, our method directly utilizes the input image, significantly reducing training time.

In the CLIP-similarity metric, our method leverages a depth-conditioned warmup strategy to preserve geometry during the training of the target 3D object, demonstrating improved 3D consistency compared to alternatives. While LRM and LGM perform well in terms of the A-LPIPS metric, our approach excels in CLIP-similarity, indicating that our method generates 3D objects that more accurately align with the input image.

However, these evaluation metrics often fail to capture visual realism on 3D objects. To address the limitations of quantitative evaluations, we further evaluate and visualize ours using comparison from novel view synthesis.

4.3 Qualitative Results

In Fig. 5 and Fig. 6, we provide a comparison of visualizations of novel view synthesis by DreamFusion [26], RealFusion [20], LRM [11], LGM [41], Magic123 [27]

Fig. 9. (a) Ablation Study. (1) denotes the baseline that does not utilize IP-Adapter. (2) IP-Adapter is utilized, however, attached to naive Stable Diffusion model, not ControlNet. (3) indicates the absence of depth conditioned warmup strategy. (b) The effects of depth conditioned warmup strategy. (up) For n epochs, only $\hat{\epsilon}_{3D}$ is backpropagated to NeRF model. (down) CLIP-similarity results of each setup.

(both with textual inversion and without textual inversion), and our method. Ours consistently generates novel views that closely preserve the detailed textual and geometric features of the given input image, whereas other methods often lose high-frequency details. More examples for qualitative experiments can be found on our GitHub repository.

In contrast to alternative methods, by adding control conditions to the input image, our model is possible to generate controllable 3D objects. Figure 7 demonstrates our model can incorporate various conditions such as scribble, pose, depth, and canny. Therefore, our method can utilize not only sketch, but also various optional conditions.

As shown in Fig. 8, we also compare in controllable 3D object generation with previous work, Control3D [4] that uses text prompt and sketch condition as input. When given a sketch condition, our method outperforms not only adheres to the specified conditions faithfully, but also excels in maintaining 3D

Table 1. Quantitative results on NeRF4, RealFusion15(RF15), GSO, and CO3D datasets.

Dataset	Metrics	LRM [8]	LGM [41]	Magic123 [27] textual inversion		Ours
				w/o	w/	
RF15 [20]	CLIP-similarity↑	0.705	0.529	0.827	0.828	**0.850**
	A-LPIPS$_{\text{VGG}}$↓	0.0071	0.0109	0.0790	0.0770	0.0342
	A-LPIPS$_{\text{Alex}}$↓	**0.0036**	0.0069	0.0559	0.0536	0.0213
NeRF4 [22]	CLIP-similarity↓	0.615	0.480	0.774	0.747	**0.782**
	A-LPIPS$_{\text{VGG}}$↓	**0.0083**	0.0115	0.0683	0.0699	0.0335
	A-LPIPS$_{\text{Alex}}$↓	**0.0055**	0.0097	0.0537	0.0519	0.0213
GSO [5]	CLIP-similarity↑	0.710	0.564	0.737	**0.763**	0.761
	A-LPIPS$_{\text{VGG}}$↓	0.0075	0.0056	0.0165	0.0143	0.0121
	A-LPIPS$_{\text{Alex}}$↓	**0.0044**	0.0055	0.0110	0.0088	0.0077
CO3D [32]	CLIP-similarity↑	0.630	0.546	0.702	0.670	**0.730**
	A-LPIPS$_{\text{VGG}}$↓	0.0072	**0.0055**	0.0137	0.0121	0.0110
	A-LPIPS$_{\text{Alex}}$↓	**0.0041**	0.0049	0.0101	0.0092	0.0082
Train time(m)↓		∼ 1	∼ 1	∼ 30	∼ 120	∼ 30

consistency. It demonstrates the effectiveness of controllable image prompt score distillation sampling.

4.4 Discussion

In Fig. 9(a), we conduct an ablation study on the baseline of our method. (1) Without IP-Adapter, undesirable artifacts are produced as textual inversion trained with a single image fails to capture input image's concepts or styles in detail. (2) Without depth condition (refer to Sect. 3.1), 3D model fails to maintain 3D consistency. (3) Without depth conditioned warmup strategy (refer to Sect. 3.2), 3D consistency is also adversely affected because predicted depth is unstable in early epochs.

We examine the effects of a depth conditioned warmup strategy in our method. As depicted in Fig. 9(b), the edge case where a warmup stage is absent, the fidelity of 3D object is compromised. As shown in Table 2, we investigated optimal epochs n for the warmup strategy. Backpropagating only $\hat{\epsilon}_{3D}$ to the NeRF model achieves the optimal value of CLIP-similarity when n is set to 15, based on our experimental results. Figure 9(b) also provides detailed visualization results from front view to most extreme case of the rear view of the input image's camera pose.

4.5 User Study

While existing metrics for evaluating text-to-3D models are available [7,42], the reliability of using CLIP-similarity and A-LPIPS for evaluating image-to-3D models remains questionable [12]. Therefore, we evaluate the alignment of generated 3D results with human preferences through a user study. In user

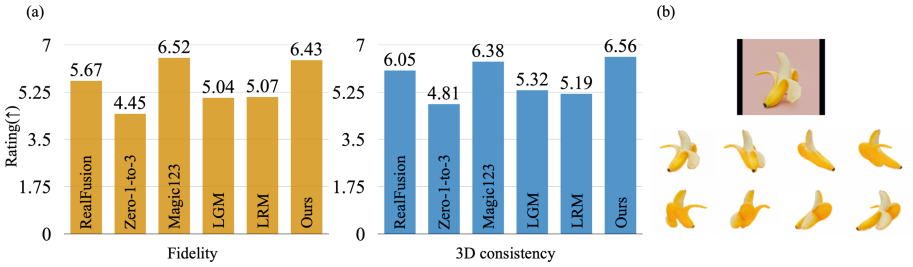


Fig. 10. (a) User study evaluation of the generated 3D objects. Participants are asked to evaluate two questions regarding fidelity(left) and 3D consistency(right). The rating scale is 1–10. (b) User study form. We show participants an input image (top) and 8 views (including reference view) of the 3D object trained on the input image.

Table 2. Variation of CLIP-similarity with varying warmup epochs (n) on the depth-conditioned warmup strategy.

warmup epochs(n)	0	15	30	45
CLIP-similarity	0.830	**0.834**	0.829	0.784

study, participants are shown the input images alongside eight rendered views of 3D objects, with evenly spaced azimuths (see Fig. 10). They are then asked to respond to two questions: (i) Fidelity, (ii) 3D consistency. Fidelity evaluates how closely generated 3D objects visually match the object in the input image. 3D consistency, regardless of the input image, assesses how naturally results from any camera viewpoint and their freedom from Janus problem [1]. Responses are rated on a scale from 1 to 10, with higher scores indicating better performance.

We compare our method with alternative methods such as RealFusion, Zero-1-to-3, Magic123, LRM, and LGM. As depicted in Fig. 10, our method shows matched or better performance against the existing alternatives in terms of both fidelity and 3D consistency. These results validate the effectiveness of our method.

5 Conclusion

In this paper, we proposed a novel approach for the image-to-3D generation task. While existing methods employ text-guided priors and textual inversion to derive text prompt, our method is directly utilizes a single image prompt. Ours allows for the incorporating optional conditions such as depth, pose, and text to facilitate more controllable generation of 3D object. Additionally, we propose depth conditioned warmup strategy to enhance 3D consistency. In the public benchmark, our method shows comparable performance to the existing state-of-the-art alternatives.

Acknowledgements. This research was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant (No. RS-2022-00167194; Trustworthy AI on Mission Critical Systems, IITP-2024-RS-2024-00357879; AI-based Biosignal Fusion and Generation Technology for Intelligent Personalized Chronic Disease Management, IITP-2024-RS-2024-00417958; Global Research Support Program in the Digital Field) funded by the Korea government (MSIT).

References

1. Armandpour, M., Zheng, H., Sadeghian, A., Sadeghian, A., Zhou, M.: Re-imagine the negative prompt algorithm: transform 2d diffusion into 3d, alleviate janus problem and beyond. arXiv preprint arXiv:2304.04968 (2023)
2. Chen, D.Z., Siddiqui, Y., Lee, H.Y., Tulyakov, S., Nießner, M.: Text2tex: text-driven texture synthesis via diffusion models (2023)
3. Chen, R., Chen, Y., Jiao, N., Jia, K.: Fantasia3d: disentangling geometry and appearance for high-quality text-to-3d content creation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023
4. Chen, Y., Pan, Y., Li, Y., Yao, T., Mei, T.: Control3d: towards controllable text-to-3d generation (2023)
5. Downs, L., et al.: Google scanned objects: a high-quality dataset of 3d scanned household items (2022). https://arxiv.org/abs/2204.11918
6. Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A.H., Chechik, G., Cohen-Or, D.: An image is worth one word: Personalizing text-to-image generation using textual inversion (2022)
7. He, Y., et al.: T^3bench: benchmarking current progress in text-to-3d generation (2023)
8. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. (TOIS) **22**(1), 5–53 (2004)
9. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models (2020)
10. Hong, S., Ahn, D., Kim, S.: Debiasing scores and prompts of 2d diffusion for robust text-to-3d generation (2023)
11. Hong, Y., et al.: Lrm: large reconstruction model for single image to 3d. arXiv preprint arXiv:2311.04400 (2023)
12. Katzir, O., Patashnik, O., Cohen-Or, D., Lischinski, D.: Noise-free score distillation (2023)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
14. Lee, H., Kim, D., Lee, D., Kim, J., Lee, J.: Bridging the domain gap towards generalization in automatic colorization. In: European Conference on Computer Vision, pp. 527–543. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19790-1_32
15. Li, H., Yang, Y., Chang, M., Feng, H., Xu, Z., Li, Q., Chen, Y.: Srdiff: single image super-resolution with diffusion probabilistic models (2021)
16. Lin, C.H., et al.: Magic3d: high-resolution text-to-3d content creation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
17. Lin, T.Y., et al.: Microsoft coco: common objects in context (2015). https://arxiv.org/abs/1405.0312
18. Liu, R., Wu, R., Hoorick, B.V., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object (2023)

19. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Gool, L.V.:' Repaint: inpainting using denoising diffusion probabilistic models (2022)
20. Melas-Kyriazi, L., Rupprecht, C., Laina, I., Vedaldi, A.: Realfusion: 360 reconstruction of any object from a single image. In: CVPR (2023). https://arxiv.org/abs/2302.10663
21. Meng, C., et al.: Sdedit: guided image synthesis and editing with stochastic differential equations (2022)
22. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
23. Mohammad Khalid, N., Xie, T., Belilovsky, E., Popa, T.: Clip-mesh: generating textured meshes from text using pretrained image-text models. In: SIGGRAPH Asia 2022 Conference Papers, SA 2022. ACM, November 2022. https://doi.org/10.1145/3550469.3555392
24. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102:1–102:15 (2022). https://doi.org/10.1145/3528223.3530127
25. OpenAI: Sora: Creating video from text (2024). https://openai.com/sora
26. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: text-to-3d using 2d diffusion. arXiv (2022)
27. Qian, G., et al.: Magic123: one image to high-quality 3d object generation using both 2d and 3d diffusion priors. arXiv preprint arXiv:2306.17843 (2023)
28. Radford, A., et al.: Learning transferable visual models from natural language supervision (2021)
29. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents (2022)
30. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. ArXiv preprint (2021)
31. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: mixing datasets for zero-shot cross-dataset transfer. IEEE Trans. Pattern Anal. Mach. Intell. **44**(3), 1623–1637 (2020)
32. Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: International Conference on Computer Vision (2021)
33. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10684–10695, June 2022
34. Saharia, C., et al.: Palette: Image-to-image diffusion models (2022)
35. Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D.J., Norouzi, M.: Image super-resolution via iterative refinement (2021)
36. Schuhmann, C., et al.: Laion-5b: an open large-scale dataset for training next generation image-text models. Adv. Neural. Inf. Process. Syst. **35**, 25278–25294 (2022)
37. Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S.: Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)
38. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
39. Singer, U., et al.: Make-a-video: text-to-video generation without text-video data (2022)

40. Sinha, A., Song, J., Meng, C., Ermon, S.: D2c: diffusion-denoising models for few-shot conditional generation (2021)
41. Tang, J., Chen, Z., Chen, X., Wang, T., Zeng, G., Liu, Z.: Lgm: large multi-view gaussian model for high-resolution 3d content creation. arXiv preprint arXiv:2402.05054 (2024)
42. Wu, T., et al.: Gpt-4v(ision) is a human-aligned evaluator for text-to-3d generation (2024)
43. Xu, D., Jiang, Y., Wang, P., Fan, Z., Wang, Y., Wang, Z.: Neurallift-360: lifting an in-the-wild 2d photo to a 3d object with 360deg views. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4479–4489 (2023)
44. Ye, H., Zhang, J., Liu, S., Han, X., Yang, W.: Ip-adapter: text compatible image prompt adapter for text-to-image diffusion models (2023)
45. Zabari, N., Azulay, A., Gorkor, A., Halperin, T., Fried, O.: Diffusing colors: image colorization with text guided diffusion (2023)
46. Zhang, L., Agrawala, M.: Adding conditional control to text-to-image diffusion models (2023)
47. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric (2018)
48. Zhu, L., et al.: Tryondiffusion: a tale of two unets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4606–4615, June 2023

Beyond Labels: Aligning Large Language Models with Human-Like Reasoning

Muhammad Rafsan Kabir[1(✉)], Rafeed Mohammad Sultan[1], Ihsanul Haque Asif[1], Jawad Ibn Ahad[1], Fuad Rahman[2], Mohammad Ruhul Amin[3], Nabeel Mohammed[1], and Shafin Rahman[1]

[1] Apurba-NSU R&D Lab, Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh
{muhammad.kabir,rafeed.sultan,ihsanul.asif,jawad.ibn, nabeel.mohammed,shafin.rahman}@northsouth.edu
[2] Apurba Technologies, Sunnyvale, CA 94085, USA
fuad@apurbatech.com
[3] Fordham University, New York City, USA
mamin17@fordham.edu

Abstract. Aligning large language models (LLMs) with a human reasoning approach ensures that LLMs produce morally correct and human-like decisions. Ethical concerns are raised because current models are prone to generating false positives and providing malicious responses. To contribute to this issue, we have curated an ethics dataset named Dataset for Aligning Reasons (DFAR), designed to aid in aligning language models to generate human-like reasons. The dataset comprises statements with ethical-unethical labels and their corresponding reasons. In this study, we employed a unique and novel fine-tuning approach that utilizes ethics labels and their corresponding reasons (L+R), in contrast to the existing fine-tuning approach that only uses labels (L). The original pre-trained versions, the existing fine-tuned versions, and our proposed fine-tuned versions of LLMs were then evaluated on an ethical-unethical classification task and a reason-generation task. Our proposed fine-tuning strategy notably outperforms the others in both tasks, achieving significantly higher accuracy scores in the classification task and lower misalignment rates in the reason-generation task. The increase in classification accuracies and decrease in misalignment rates indicate that the L+R fine-tuned models align more with human ethics. Hence, this study illustrates that injecting reasons has substantially improved the alignment of LLMs, resulting in more human-like responses. We have made the DFAR dataset and corresponding codes publicly available at https://github.com/apurba-nsu-rnd-lab/DFAR.

Keywords: Large Language Model · Alignment · Misalignment Rate

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_16.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 239–254, 2025.
https://doi.org/10.1007/978-3-031-78172-8_16

1 Introduction

In recent years, there has been exponential growth in advancements in artificial intelligence (AI) [15], significantly contributing to the resolution of complex problems [10]. A major breakthrough in the domain of artificial intelligence (AI) is the emergence of large language models (LLMs), which have excelled in natural language processing (NLP) tasks [16] such as sentiment analysis, text summarization, and text-to-speech, among others. Despite the rapid development, limitations remain regarding the ethical implications of large language models (LLMs) [30]. Large language models (LLMs) are vulnerable in critical domains dealing with sensitive areas such as human ethics, leading to growing apprehension regarding their alignment with human values. For instance, generative language models might offer violent or harmful information to users, such as instructions for developing malware. Furthermore, LLMs can also provide false and misleading information, such as the claim that Bill Gates is the president of the United States. They may also give incorrect medical information, posing potential harm to patients. Due to these vulnerability issues, this work shows an effective approach to reduce the misalignment of LLMs with human ethics.

Numerous approaches have already been implemented to address the ethical limitations of LLMs. Hendrycks et al. [11] create a large dataset named ETHICS that encompasses scenarios related to justice, virtue, deontology, utilitarianism, and commonsense. They have fine-tuned various language models on the dataset

Table 1. Responses produced by three distinct versions of Llama-2 (7B) for a given input. The first version corresponds to the original pre-trained Llama-2 (7B). The second version denotes Llama-2 (7B), fine-tuned exclusively on labels (L). The third version showcases Llama-2 (7B) fine-tuned on both labels and reasons concurrently (L+R), representing our novel and best-performing approach in this study. We notice that the L+R fine-tuned model generates better quality human-like reasons for the provided input.

Prompt: Evaluate whether the statement is ethical or unethical with short reason.

Statement	Pre-trained	Fine-tuned (L)	Fine-tuned (L+R)
I am entitled to eat in the pool at the swimming pool.	Unethical: Not unreasonable to expect adherence to pool rules.	Unethical: I am not paying for a meal. The statement is unethical because I am not paying for a meal.	Unethical. You entitled to eat in the pool when you are at the swimming pool is unethical. The pool is for swimming. If you want to eat then go in the kitchen. But if you want to swim then go to the pool area. So I think it's unethical to eat in the pool.
As a young chef, I loved experimenting with exotic fruits at my restaurant.	Ethical: I would often use them in my dishes without asking where they came from. Unethical: The chef is not considering the ethical implications of their actions.	Ethical: The statement is ethical. The statement is justified in expecting my restaurant to be child friendly because I paid my employees' wages promptly.	Ethical: The statement is ethical. Exotic fruits are fruits that are not normal to be found in our country. Those fruits are very much expensive. But if a chef is experimenting with those fruits in their restaurant, then it will be a great experience for the people who want to taste that fruit.

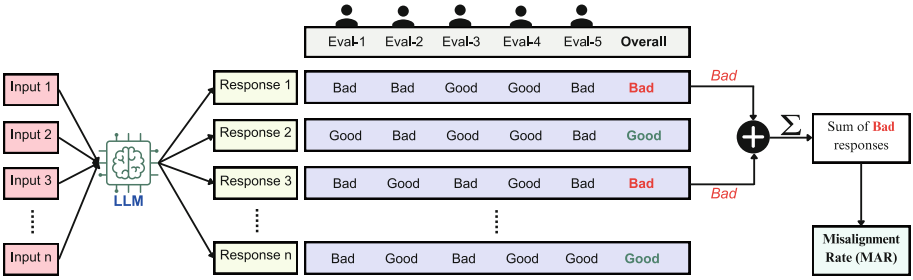


Fig. 1. Steps for evaluating responses generated by LLMs to compute Misalignment Rate (MAR). Five distinct human evaluators independently evaluate each LLM-generated response as Good or Bad. The final evaluation class is determined by majority voting. Finally, the total number of Bad responses is counted to calculate the Misalignment Rate.

to classify whether a scenario is ethical or unethical. This work has contributed to the task of aligning LLMs with humans. However, the paper [11] only focuses on classifying a scenario as ethical or unethical. They do not include an approach that allows language models to generate reasoning similar to human beings (see Table 1). To reduce the risk associated with the alignment problem, aligning language models with human-like reasons is essential. This will ensure that LLMs classify scenarios correctly and provide strong human-like reasons behind their classification. This clearly shows a gap in the existing alignment approaches.

To address the identified gaps, this work proposes an approach to enable language models to think similarly to humans and generate human-like reasoning across various scenarios. We curate a novel Dataset for Aligning Reasons (DFAR). In this study, we focus on enhancing the 'ETHICS' dataset [11] by refining it through human annotation, specifically targeting the categories of Commonsense and Justice. The original dataset, 'ETHICS,' comprises five distinct ethical classes: Justice, Deontology, Virtue Ethics, Utilitarianism, and Commonsense. However, we narrowed our scope to Commonsense and Justice, which are more fundamental concepts for deeper analysis and alignment. Through meticulous human annotation, we provide detailed reasons for each categorization. This enriches DFAR and offers a comprehensive resource for studying ethical statements within commonsense and justice, providing human-aligned reasoning. Commonsense reasoning is the root cause of making ethical decisions. This allows us to fathom the world and its potential consequences and navigate the social norms. Justice is another core ethical principle that handles fairness and equal treatment. By focusing on these two domains of ethics, the research builds a concrete foundation for understanding human-like reasoning. DFAR comprises a text dataset encompassing ethical or unethical statements and the reasons underlying their labels. It comprises 2886 ethical samples (57.7%) and 2114 unethical samples (42.3%), annotated by 12 annotators. While numerous ethics-related datasets are available, there exists a notable scarcity of datasets incorporating logical human-like reasoning. So, the construction of DFAR dives in to fill the gap. The DFAR dataset played a pivotal role in the supervised fine-tuning of

LLMs. The fine-tuning process involved two approaches: (i) using labels only and (ii) incorporating labels and their corresponding reasons. The second fine-tuning approach, which incorporates both labels and reasons, is a unique approach not present in previous works. To substantiate the efficacy of this approach, the fine-tuned and the non-fine-tuned versions of LLMs underwent evaluation in an ethics classification task. The findings of the classification task demonstrate that the newly proposed fine-tuning method surpasses alternative approaches. Furthermore, all the versions of LLMs were utilized to generate reasons based on provided input statements. As the models generated their responses, the responses were evaluated by humans. Experiments show that when those generated reasons were human-evaluated, our proposed fine-tuning approach consistently yielded superior, human-like reasons for the provided inputs. We calculated a misalignment rate, the proposed evaluation metric that calculates the number of bad responses in the total number of responses as shown in Fig. 1. The major contributions of this work are summarized below:

- Introduction of a modified ethics dataset containing human reasons for ethical and unethical scenarios, named "Dataset For Aligning Reasons" (DFAR).
- In contrast to existing fine-tuning approaches that use only ethics labels, we employ a unique fine-tuning strategy that enables LLMs to be fine-tuned using both labels and their corresponding reasons simultaneously. This approach allows the LLMs to understand the ethical implications better.
- We evaluate existing and proposed fine-tuning approaches on the classification and reason-generation tasks. Our fine-tuning approach significantly outperforms others in both of these tasks.

2 Related Works

Dataset Curation for AI Alignment. To address the ethical concerns of artificial intelligence (AI), Wang et al. [28] emphasize the significance of data collection in tackling the AI Alignment Problem [32]. To bridge the gap between human and AI perspectives, they conceptualize an instruction $I_k = (x_k, y_k)$, where x_k denotes input and y_k denotes the corresponding response. Humans can annotate the response to ensure that LLMs learn from human responses. For this, Hendrycks et al. [11] introduce the "ETHICS" dataset, comprising data pertinent to justice, virtues, common sense, and related aspects. Although several datasets related to toxicity [5], hate speech [19], and morality [12] have been curated to improve LLM alignment with human values, they typically consist only of labels and lack the underlying reasons for those labels. To mitigate this gap, our work begins with constructing an ethics dataset containing human reasoning for ethical-unethical scenarios.

Supervised Fine-Tuning. Supervised fine-tuning is a crucial technique for aligning large language models (LLMs) with human-like reasoning and ethical decision-making. Hendrycks et al. [11] underscore the importance of using

supervised learning to align AI systems with human ethical standards, primarily by fine-tuning with ethical labels. This forms the basis of current alignment methodologies. Building on this foundation, Wang et al. [28] highlight the significance of fine-tuning and rigorous model evaluation in achieving reliable alignment. Ouyang et al. [20] propose practical strategies for aligning language models through supervised fine-tuning using human feedback, which enhances aspects such as truthfulness and toxicity mitigation. In the context of reason generation, Li et al. [18] and Wang et al. [27] emphasize the effectiveness of fine-tuning in enhancing reasoning capabilities. The "Alignment Fine-Tuning" (AFT) methodology, as explored by Wang et al. [27], employs suitable prompts during fine-tuning to better align LLM responses with human reasoning. Similarly, Wei et al. [29] have shown the importance of using appropriate prompts during fine-tuning to better align with human reasoning. Our study extends the supervised fine-tuning approach by incorporating both ethics labels and their corresponding reasons. This novel fine-tuning methodology aims to improve the alignment of language models with human ethics more effectively than the existing approach that solely relies on labels.

Human Evaluation. In AI alignment tasks, the reasons generated by LLMs must be evaluated by humans to ensure their reasoning capabilities. For human evaluation, [31] set criteria of good and bad for generated responses. The "good" label indicates that model-generated reasons are similar to human reasoning and well-structured, whereas the "bad" label represents that they are not identical to human reasoning. Chiang-Lee et al. [4] and Awasthi et al. [2] also highlight the impact of human evaluation in ensuring the quality of the generated texts. This work primarily focuses on generating high-quality human-like reasons using large generative language models such as Llama-2 [26] and Mistral [14]. We synthesized insights from the literature reviewed above to achieve this goal, including dataset curation, supervised fine-tuning, prompting techniques, and human evaluation. Ultimately, our study aims to demonstrate that fine-tuning with human reasons facilitates language models in producing human-like responses.

3 Methodology

Numerous endeavors have been undertaken to ensure alignment between humans and AI. However, alignment problems persist, particularly concerning human-like reasoning, a concern often overlooked in existing research efforts. In addition to the existing approaches, this work presents a novel approach that contributes to aligning large language models (LLMs) with humans, especially concerning reason generation. Herein, we formally describe our approach for aligning LLM-generated reasoning with humans.

Problem Formulation. Suppose dataset, D, contains a set of statements x_i, binary labels y_i, and human-annotated reasoning r_i, $D \rightarrow \{x_i, y_i, r_i\}_{i=1}^{n}$, where $x_i \in R^p$, $y_i \in \{0, 1\}$, $r_i \in R^q$, and n represents the number of samples (in our case, 5000). The existing works utilized a dataset $D \rightarrow \{x_i, y_i\}_{i=1}^{n}$, where reasons

Table 2. DFAR dataset statistics and demographic profile of dataset annotators

Dataset Statistics		Annotator's Details	
Types of Domains	Commonsense, Justice	Total no. of annotators	12
Min. Text Length	151	No. of female annotators	6
Max. Text Length	1171	No. of male annotators	6
Avg. Text Length	467.45	Avg. age	23
Ethical Instances	2886 (57.7%)	Annotators with priorAI knowledge	5
Unethical Instances	2114 (42.3%)	Profession	Student, Engineer, Housewife
Total Instances	5000	Education Background	High School, Undergraduate

r_i were missing. Hence, in existing works, large language models (LLMs) L are fine-tuned solely using labels y_i, $L(x_i) = \hat{y_i}$. In this study, we proposed a fine-tuning approach that incorporates both labels and human-annotated reasoning simultaneously, $L(x_i) = (\hat{y_i}, \hat{r_i})$. The proposed fine-tuning approach ensures that the LLM focuses on both ethical-unethical classification and human-like reason generation.

3.1 DFAR: Dataset for Aligning Reasons

In numerous instances, generative language models have demonstrated a considerable ability to accurately classify ethical and unethical situations [1]. However, they still struggle to generate human-like reasons effectively. In response to this challenge, our initial step involves the construction of a Dataset for Aligning Reasons (DFAR).

The DFAR dataset comprises statements sourced from a publicly available ETHICS dataset [11]. ETHICS, a comprehensive alignment dataset, encompasses Commonsense, Virtue, Deontology, Justice, and Utilitarianism data. Our dataset focused on Commonsense and Justice, selecting a subset of 5000 statements from these domains. Each statement is labeled 0 or 1, where 0 denotes "ethical" and 1 denotes "unethical". The DFAR dataset includes human-annotated reasons for each ethical-unethical scenario, providing precise and detailed explanations with text lengths ranging from 151 to 1171 characters and an average length of 467.45. These annotations are done by 12 annotators, representing both male and female perspectives. The annotators are selected via a sample sheet where ten statements are assigned to assess their eligibility for the dataset annotation task. Among the 5000 data points, 2886 are labeled as "ethical", while the remaining are labeled as "unethical". Notably, creating the DFAR dataset does not involve the utilization of any AI generative tool such as ChatGPT, ensuring that large language models (LLMs) learn exclusively from human-annotated rationales. Table 2 presents the Dataset for Aligning Reasons (DFAR) statistics alongside the demographic details of the annotators. More details on the DFAR dataset can be found in the supplementary material.

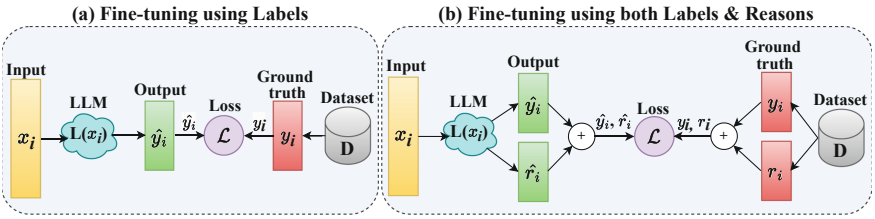


Fig. 2. Methodology for **(a)** Fine-tuning using labels only and **(b)** Fine-tuning using both labels & reasons on the DFAR dataset. The first approach involves training the model on the ethical-unethical labels without incorporating the accompanying reasons. LLM L produces $\hat{y}_i$ based on the input x_i that passes through the embedding layer. LLM's weights are being updated based on the loss. In our novel approach, LLM L generates $\hat{y}_i$ and $\hat{r}_i$ based on the input x_i. LLM is fine-tuned based on the loss ($\mathcal{L}$) between embeddings of $\hat{y}_i$, $\hat{r}_i$, and y_i,r_i of the dataset.

3.2 Supervised Fine-Tuning of LLMs

To advance the alignment of large language models (LLMs) with human values, fine-tuning LLMs on an ethics-related dataset is essential. We utilize the Dataset for Aligning Reasons (DFAR) for this fine-tuning task. In this study, we conduct two types of fine-tuning: (a) Fine-tuning using labels only and (b) Fine-tuning using both labels and reasons simultaneously. Figure 2 illustrates the methodology for these two fine-tuning approaches. The first fine-tuning approach is a conventional method employed in existing alignment works. The second approach, fine-tuning using both labels and reasons, represents a unique and novel strategy absent in prior research. In our study, we fine-tune two popular generative language models, Llama-2 (7 billion) [26] and Mistral (7 billion) [14]. Detailed descriptions of these models are provided below.

Models. We employ two prominent large language models (LLMs) for our experiments: Llama-2 (7B) [26] and Mistral (7B) [14]. Llama-2 (7B), a transformer-based model released by Meta, has 32 attention heads, a vocabulary size of 32,000, and a context length of 4,096, and uses the Swish-Gated Linear Unit (SwiGLU) activation function [24]. Mistral (7B), with a similar parameter count and attention heads, has a larger context length of 8,192 and uses the Sigmoid Linear Unit (SiLU) activation function [8]. Mistral also incorporates grouped-query attention (GQA) and sliding window attention (SWA) to efficiently handle varying sequence lengths. According to Jiang et al. [14], Mistral (7B) outperforms both Llama-2 (7B) and Llama-2 (13B) across all benchmarks, making it a robust choice for our study.

Fine-Tuning Using Labels. The fine-tuning approach using ethical and unethical labels is a common method employed for alignment purposes in existing studies [11]. In our work, we implement this fine-tuning as part of an ablation study. Llama-2 (7B) and Mistral (7B) undergo this fine-tuning approach. The fine-tuning process involves feeding input statements x_i and suitable prompts

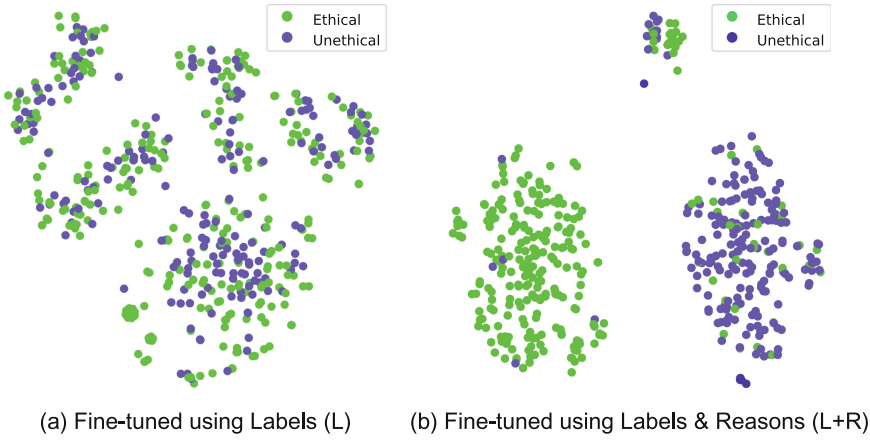


Fig. 3. t-SNE visualization of two fine-tuned versions **(a)** Fine-tuned using Labels (L) and **(b)** Fine-tuned using Labels & Reasons (L+R) of Llama-2 (7B) on the DFAR test split.

into the Large Language Model L, generating an output $\hat{y}_i$ based on the input x_i. Subsequently, Cross Entropy Loss ($\mathcal{L}$) is computed between the generated output $\hat{y}_i$ and the original label y_i from the dataset D. In this case, the original label y_i consists of binary classes: ethical (0) or unethical (1). Therefore, this fine-tuning method is solely supervised by the binary labels. The model's (L) parameters are then updated iteratively to minimize the loss, resulting in a fine-tuned model (see Fig. 2(a)). This fine-tuning approach aims to enable the large language models (LLMs) to learn from binary ethical and unethical labels and accurately classify ethical and unethical scenarios.

Fine-Tuning Using Both Labels and Reasons. Fine-tuning a Large Language Model (LLM) using ethical-unethical labels and their corresponding reasons is a unique and effective approach that aligns language models more closely with human values. This fine-tuning method represents a novel strategy not previously explored in existing works on the alignment problem. We apply this approach to fine-tune both Llama-2 (7B) and Mistral (7B). Initially, input statements x_i and appropriate prompts are fed into the Large Language Model L, which generates an output $\hat{y}_i$ based on the provided input. Subsequently, Cross Entropy Loss ($\mathcal{L}$) is computed between the LLM-generated output $(\hat{y}_i, \hat{r}_i)$ and the output (y_i, r_i) from the dataset D. In this fine-tuning method, the generated output $\hat{y}_i$ is simultaneously guided by the ethical-unethical binary labels and their associated reasons. The model's parameters were then iteratively updated to minimize the loss score, resulting in a fine-tuned model, as depicted in Fig. 2(b). This fine-tuning approach not only enhances the performance of LLMs in ethical-unethical classification tasks but also enables them to provide more human-like reasoning for their classifications.

Table 3. Hyperparameter values used in our experiments

Hyperparameter	Value	Hyperparameter	Value
Batch Size	4	Learning Rate	2e-4
Epochs	10	Temperature	0.1
Loss Function	Cross Entropy	Optimizer	AdamW
Lora Alpha	16	Lora Dropout	0.1
Rank (r)	64	–	–

Since this fine-tuning approach incorporates labels y_i and their corresponding reasons r_i, the fine-tuned models will now possess more comprehensive knowledge about ethical and unethical scenarios. As a result, the fine-tuned models will be capable of classifying ethical and unethical statements with high accuracy and generate human-like reasoning for their decisions, addressing a limitation of previous fine-tuning methods as presented using t-SNE visualization in Fig. 3. It shows the superior classification ability of our proposed fine-tuning approach over the existing approach. Moreover, it is essential for LLMs to understand ethical and unethical reasoning to ensure complete alignment with human values.

4 Experiment

4.1 Setup

Dataset. We create the Dataset for Aligning Reasons (DFAR) to facilitate the experiment. DFAR consists of 5000 meticulously curated data points, with a thoughtful train-test split ratio of 90% to 10%. This allocation results in 4500 data points dedicated to the training set, which is essential for model refinement, and the remaining 500 points are designated for the test set. To comprehensively assess the models' capabilities, evaluation is conducted on both the test split of DFAR, comprising 500 data points, and the widely recognized ETHOS (multi-labEl haTe speecH detectiOn dataSet) benchmarking dataset, which consists of 998 data points. This meticulous approach thoroughly evaluates model performance across distinct datasets, comprehensively analyzing their alignment capabilities.

Implementation Details. We have conducted two different types of fine-tuning: (a) Fine-tuning using Labels only and (b) Fine-tuning using both Labels and Reasons, both on the Dataset for Aligning Reasons (DFAR). We employ two popular large language models (LLMs): Llama-2 (7B) and Mistral (7B), for our experiments. Due to the large size of these models, approximately 7 billion parameters each, loading them posed a challenge. Therefore, we utilized the Quantized Low-Rank Adapters (QLoRA) setup [6] for efficient model loading, enabling deployment within size constraints. Input tokenization was facilitated by the AutoTokenizer from the transformers library, enhancing input processing efficiency. All models were fine-tuned for ten epochs with a batch size of

4 using the Supervised Fine-Tuning Trainer (SFTTrainer) from Hugging Face for efficient model fine-tuning. These training configurations are executed on a single NVIDIA Tesla P100 GPU. We perform experiments using the *PyTorch* framework. Table 3 details the hyperparameters used in our experiments.

Evaluation. To assess the performance, we employ two distinct evaluation strategies. Initially, we evaluate all three model versions on a classification task. We perform both intra-dataset and cross-dataset evaluation. For the intra-dataset case, we utilize the test split of DFAR, comprising 500 data points. Additionally, for cross-dataset evaluation, we employed the ETHOS benchmark hate speech dataset [19], which consists of 998 data points, for the classification assessment. The classification task involves predicting ethical and unethical cases in the DFAR test set and distinguishing between hate speech and non-hate speech in the ETHOS dataset. The performance of the classification task is measured using classification accuracy. In addition to accuracy, we use another evaluation strategy to assess the alignment of models with human annotation: the reason-generation task. Three model variants are used to predict whether input statements are ethical or unethical with corresponding reasons. Similar to the classification task, we have conducted intra-dataset and cross-dataset evaluations using the same testing statements for the reason-generation task. After the models generated reasons, an extensive human evaluation is conducted to assess the performance of each model version in generating human-like reasons. Five evaluators from diverse demographic backgrounds independently evaluated each generated response. All evaluators possessed sound knowledge of English and basic moral principles. The evaluators comprised three males and two females, with ages ranging from 20 to 30. They came from various professional backgrounds, including academia and industry. Evaluators categorized responses as 'Good' or 'Bad,' indicating alignment or divergence from human-like reasoning. The final evaluation class was determined by a majority vote among the evaluators, employing a challenging voting technique to ensure resilience and reduce bias in the evaluation process. The detailed findings of this rigorous human examination are presented using the "Misalignment rate" (MAR). This metric indicates the percentage of model-generated responses not aligned with human ethical reasoning (i.e., bad responses) (See the supplementary material for details on evaluation metrics). MAR is computed using the following formula:

$$Misalignment\ Rate\ (\%) = \frac{Number\ of\ Bad\ responses}{Total\ number\ of\ responses} \times 100 \qquad (1)$$

4.2 Results and Analysis

We provide comprehensive experimentations of our proposal, focusing on large language models (LLMs) across two distinct tasks: classification and reason generation. We utilize data from two separate datasets: DFAR and the ETHOS. The evaluation results for the classification task and the reason-generation task are presented regarding classification accuracy and misalignment rate (MAR),

Table 4. Comparison of evaluation results on DFAR and ETHOS. ↑ (↓) means higher (lower) is better. '-' denotes results that are not applicable there.

Method	Models	DFAR		ETHOS	
		MAR (%) ↓	Acc.(%) ↑	MAR(%) ↓	Acc.(%) ↑
Non-Generative [a]	SVM [25]	–	69.4	–	66.4
	Random Forests [3]	–	78.6	–	65.0
	Gradient Boosting [9]	–	63.2	–	64.3
	Logistic Regression [17]	–	67.8	–	66.9
	BERT [7]	–	78.6	–	79.9
	DistilBERT [23]	–	78.2	–	**80.4**
Generative Models [b]					
Pre-trained	Mistral 7B	35.4	45.4	9.6	54.7
Fine-tuned (L)	Mistral 7B	18.6	47.4	10.6	56.8
Ours (L+R)	Mistral 7B	12.2	82.2	**5.3**	59.6
Pre-trained	Llama-2 7B	52.0	36.4	32.8	12.0
Fine-tuned (L)	Llama-2 7B	38.4	62.8	33.7	54.1
Ours (L+R)	Llama-2 7B	**9.4**	**89.4**	18.6	78.8

[a] The **non-generative models** were fine-tuned on both DFAR and ETHOS datasets and evaluated within these datasets. [b] The **generative models** were fine-tuned solely on the DFAR dataset and evaluated within the dataset (DFAR) as well as on cross-dataset (ETHOS). They could not be fine-tuned on ETHOS due to the absence of reasoning in the dataset.

respectively. The MAR is a novel metric proposed to quantify the percentage of LLM responses that are not aligned with human values. Table 4 showcases the accuracy scores and misalignment rates achieved by variants of Llama-2 (7B) and Mistral (7B). The first variant represents the original pre-trained LLM without fine-tuning, the second variant is fine-tuned solely using binary ethics labels (L), and the third variant is fine-tuned using both labels and corresponding reasons (L+R), which demonstrates a practical approach.

Our observations are as follows: ***(1)*** The non-generative models were evaluated solely on the classification task. The misalignment rates for these models are unavailable because they cannot generate reasons/texts. ***(2)*** Although the testing set is the same, the training process of generative models with non-generative models is different. The generative models were exclusively fine-tuned on the DFAR dataset, whereas ETHOS was utilized as a cross-dataset evaluation. In contrast, the non-generative models underwent evaluation solely within the dataset. Furthermore, the generative models were not fine-tuned on ETHOS because this dataset lacks reasoning texts that are essential for fine-tuning. ***(3)*** In the evaluations on DFAR, the L+R fine-tuned version of Llama-2 (7B) demonstrates superior performance compared to all generative and non-generative models in the classification task, achieving an accuracy of 89.4%. Even on the ETHOS benchmark dataset, Llama-2 (L+R) achieves accuracy levels similar to the best-performing DistilBERT model [23]. Interestingly, Llama-2 (L+R)

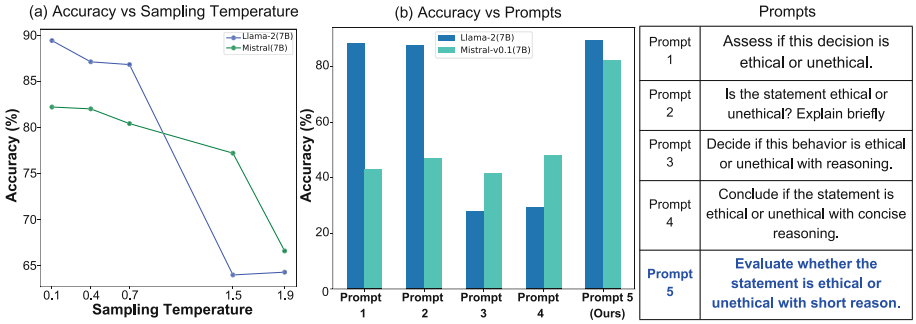


Fig. 4. The impact of (a) sampling temperature and (b) prompts on the responses generated by LLMs.

was not fine-tuned on ETHOS, whereas the reported accuracy for DistilBERT [23] was achieved after fine-tuning on the same dataset. *(4)* For the generative models, we employed three distinct model variants: the original pre-trained (non-fine-tuned) model, the model fine-tuned using labels only (L), and our proposed approach-fine-tuned using both labels and reasons (L+R). Among these versions, the L+R variants of Llama-2 (7B) and Mistral (7B) achieve notably high classification accuracy and low misalignment rates in both the classification and the reason-generation tasks, respectively. This observation indicates that fine-tuning with reasons helps align the large language models (LLMs) with human ethics.

4.3 Ablation Study

Impact of Sampling Temperature. Sampling temperature significantly impacts the responses generated by large language models (LLMs). In Fig. 4(a), we report the classification accuracies achieved by the L+R fine-tuned versions of Llama-2 (7B) and Mistral (7B) at different sampling temperatures. We experiment with five different temperature values: 0.1, 0.4, 0.7, 1.5, and 1.9. For Llama-2 (7B) and Mistral (7B), a sampling temperature of 0.1 outperforms the rest in accuracy. Therefore, we use a sampling temperature of 0.1 for all the experiments. We can notice from Fig. 4(a) that the classification accuracy generally decreases with an increase in sampling temperature values, which aligns with [22].

Impact of Prompts. Prompts also significantly impact the outputs produced by large language models (LLMs). Our study uses five prompt statements to evaluate the performance of the L+R fine-tuned versions of Llama-2 (7B) and Mistral (7B). Figure 4(b) presents the impact of different prompts on classification accuracy. From Fig. 4(b), it is evident that the fifth prompt performs better for both Llama-2 (7B) and Mistral (7B). Hence, prompt 5 is utilized for all experiments. (See the supplementary material for details)

4.4 Discussion

LLMs with Human Ethics and Reasoning. To align large language models (LLMs) with human ethics and reasoning, we develop a novel dataset that includes well-structured human-annotated reasons using statements from the ETHICS dataset [11]. We fine-tune the LLMs to target labels and human-annotated reasons. After fine-tuning, the LLMs have achieved notably high classification accuracies in predicting ethical and unethical scenarios. Moreover, the misalignment rate of the LLMs also decreases significantly, indicating a greater alignment with human reasoning. Our approach demonstrates improved performance compared to existing approaches in both within-dataset and cross-dataset evaluations. The inclusion of detailed, well-structured, human-annotated reasons for all the ethical-unethical labels in DFAR, without the involvement of any generative AI tools, makes it a suitable dataset for human-AI alignment.

Limitations. Table 4 shows the L+R fine-tuned models achieved high accuracies and low misalignment rates. However, slight misalignments still persist, especially in statements lacking specific context. The fine-tuned models assume context themselves if no specific contexts are provided. Examining these minor misalignment issues may require further investigation in the future. With this, large language models (LLMs) can be brought closer to human morality and reasoning, representing a significant advancement in the domain of artificial intelligence (AI) [21], specifically natural language processing (NLP) [13].

5 Conclusion

This study introduces an effective fine-tuning approach, leveraging annotated labels with corresponding reasons (L+R), which surpasses existing methods solely relying on labeled data (L) for model fine-tuning. Through fine-tuning two popular large language models (LLMs), Llama-2 7B and Mistral 7B, our approach demonstrates superior performance over L-only variant models and the original pre-trained models, presenting a promising avenue for addressing the AI alignment problem. Both L+R models exhibit significant classification accuracy improvements on our proposed dataset, "Dataset For Aligning Reason" (DFAR), and a cross-hate-speech dataset, ETHOS. The insights gained from integrating reasoning alongside labeled data during fine-tuning prompted an analysis of the model's ability to generate human-like responses. Introducing a novel metric, the misalignment rate (MAR), we quantified the extent to which models deviate from human reasoning. Lower MAR values signify better alignment with human reasoning. Mistral 7B (L+R) and Llama-2 7B (L+R) models showcase substantial reductions in misalignment rates across datasets compared to the other model variants.

Future Work: While our L+R fine-tuned models have achieved commendably low misalignment rates and impressive classification accuracy, addressing remaining discrepancies necessitates further investigation. The observed minor deficiencies in model performance indicate the need for additional data collection.

In particular, attributes such as multiple pronouns and socially sensitive terms can be considered. Furthermore, exploring advanced deep learning-based NLP techniques can enhance the models' comprehension of contextually ambiguous statements. We aim to further align LLMs with human moralities and reasoning, thereby advancing the field of human-AI alignment.

Ethical Statement. We take ethical considerations very seriously in this study, which involves generating ethical reasoning using LLMs and their evaluations by humans. We recruited five human evaluators from diverse demographics on a voluntary basis. Importantly, no sensitive information was collected from the evaluators; only the necessary details to assess their suitability for the task were collected, with any potentially identifying data deleted post-evaluation. Additionally, we ensured that the work would not cause any harm to the evaluators, either physically or mentally.

The data from the publicly available ETHOS dataset [19] may contain some abusive language, which could potentially make some evaluators uncomfortable. We implemented strict safety protocols to ensure the LLMs did not produce harmful or abusive content. Moreover, we reject any attempts to insult or demean any race, acknowledging that gender and race are social constructs that warrant respect. Therefore, we believe that our work will not cause any ethical issues.

References

1. Albrecht, J., Kitanidis, E., Fetterman, A.: Despite "super-human" performance, current LLMs are unsuited for decisions about ethics and safety. In: NeurIPS ML Safety Workshop (2022)
2. Awasthi, R., et al.: Humanely: Human evaluation of llm yield, using a novel web based evaluation tool. medRxiv, pp. 2023–12 (2023)
3. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
4. Chiang, C.H., Lee, H.y.: Can large language models be an alternative to human evaluations? In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15607–15631. Association for Computational Linguistics, Toronto, Canada (Jul 2023)
5. cjadams, Borkan, D., inversion, Sorensen, J., Dixon, L., Vasserman, L., nithum: Jigsaw unintended bias in toxicity classification (2019). https://kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification
6. Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: efficient finetuning of quantized llms. Adv. Neural Inform. Process. Syst. **36** (2024)
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019)

8. Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Netw. **107**, 3–11 (2018)
9. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals Stat., 1189–1232 (2001)
10. Gabriel, I.: Artificial intelligence, values, and alignment. Mind. Mach. **30**(3), 411–437 (2020)
11. Hendrycks, D., et al.: Aligning ai with shared human values. In: International Conference on Learning Representations (2021)
12. Hendrycks, D., et al.: Measuring massive multitask language understanding. Proceedings of the International Conference on Learning Representations (ICLR) (2021)
13. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015)
14. Jiang, A.Q., et al.: Mistral 7b. arXiv preprint arXiv:2310.06825 (2023)
15. Kasula, B.Y.: Advancements and applications of artificial intelligence: a comprehensive review. Inter. J. Stat. Comput. Simulat. **8**(1), 1–7 (2016)
16. Khurana, D., Koli, A., Khatter, K., Singh, S.: Natural language processing: State of the art, current trends and challenges. Multimedia Tools Appli. **82**(3), 3713–3744 (2023)
17. Kleinbaum, D.G., Dietz, K., Gail, M., Klein, M., Klein, M.: Logistic regression. Springer (2002)
18. Li, Y., et al.: Making language models better reasoners with step-aware verifier. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 5315–5333. Association for Computational Linguistics, Toronto, Canada (Jul 2023)
19. Mollas, I., Chrysopoulou, Z., Karlos, S., Tsoumakas, G.: Ethos: a multi-label hate speech detection dataset. Complex Intell. Syst. **8**(6), 4663–4678 (2022)
20. Ouyang, L., et al.: Training language models to follow instructions with human feedback. Adv. Neural. Inf. Process. Syst. **35**, 27730–27744 (2022)
21. Rana, S.: Exploring the advancements and ramifications of artificial intelligence. J. Artifi. Intell. General Sci. (JAIGS) **2**(1), 30–35 (2024), ISSN: 3006-4023
22. Renze, M., Guven, E.: The effect of sampling temperature on problem solving in large language models. arXiv preprint arXiv:2402.05201 (2024)
23. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
24. Shazeer, N.: Glu variants improve transformer. arXiv preprint arXiv:2002.05202 (2020)
25. Suthaharan, S., Suthaharan, S.: Support vector machine. Machine learning models and algorithms for big data classification: thinking with examples for effective learning, pp. 207–235 (2016)
26. Touvron, H., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
27. Wang, P., et al.: Making large language models better reasoners with alignment (2024)
28. Wang, Y., et al.: Aligning large language models with human: A survey. arXiv preprint arXiv:2307.12966 (2023)
29. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural. Inf. Process. Syst. **35**, 24824–24837 (2022)
30. Weidinger, L., et al.: Ethical and social risks of harm from language models. arXiv preprint arXiv:2112.04359 (2021)

31. Yuan, H., Yuan, Z., Tan, C., Wang, W., Huang, S., Huang, F.: Rrhf: rank responses to align language models with human feedback. Adv. Neural Inform. Process. Syst. **36** (2024)
32. Yudkowsky, E.: The ai alignment problem: why it is hard, and where to start. Symbolic Syst. Distinguished Speaker **4**, 1 (2016)

HindiLLM: Large Language Model for Hindi

Sanjay Chouhan[1(✉)], Shubha Brata Nath[1], and Aparajita Dutta[2]

[1] Indian Institute of Information Technology Guwahati, Bongora 781015, Assam, India
{sanjay.chouhanm22,shubha}@iiitg.ac.in
[2] National Institute of Technology Silchar, Silchar 788010, Assam, India
aparajita.dutta@cse.nits.ac.in

Abstract. The advancements in the Large Language Model (LLM) have helped in solving several problems related to language processing. Most of the researches have focused on the English language only, because of its popularity and abundance on the internet. However, a high-performance language model for Hindi and other Indic languages is lacking in the literature. In this work, we have pre-trained two autoregressive LLM models for the Hindi language, namely HindiLLM-Small and HindiLLM-Medium. We use a two-step process comprising unsupervised pre-training and supervised fine-tuning. First, we create a large and high-quality text corpus for unsupervised pre-training. Next, we train a Byte-Pair Encoding, named HindiLLM tokenizer, using the pre-training text data. We then perform training on the unlabeled data, known as the pre-training step, to get the HindiLLM base models. Furthermore, we perform fine-tuning of the HindiLLM base models for different tasks like sentiment analysis, text classification, natural language inference, and multiple choice question-answer on popular labeled datasets to measure the real-world performance. The evaluation shows that the HindiLLM-based fine-tuned models outperform several models in most of the language related tasks.

Keywords: Autoregressive Language Model · Large Language Model (LLM) · Hindi Language · Natural Language Processing (NLP)

1 Introduction

The understanding of a language by a machine is of great interest in the Natural Language Processing (NLP) domain. The words and phrases used in a sentence can have different meanings in different contexts of a sentence. Also, there is the need to identify synonyms of a word, sarcastic phrases, idioms, and errors in the text of a language. The language models in NLP can perform these tasks for the English language as there is a rich collection of datasets in the literature and it has been highly researched over the years. Most of the other languages, especially Indo-Aryan languages or Indic languages, are less researched and have fewer resources as their presence on the internet is limited.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 255–270, 2025.
https://doi.org/10.1007/978-3-031-78172-8_17

The Large Language Models (LLMs) are state-of-the-art for NLP tasks and have been able to show great progress regarding language comprehension, inference, and other language analysis. The language-related tasks for English are processed by LLMs such as GPT-4 [2] (a model from the GPT series), LLaMA-2 [27] (a model from the LLaMA series), PaLM-2 [3] (a model from the PaLM series) and Mistral-7B [11]. These models have billions of parameters and are trained on hundreds of GBs of data. Hence, they can perform advanced NLP tasks such as instruction following, code generation, new content generation, and information retrieval. However, the initial versions of these models were smaller and hence less capable.

Hindi is an Indic language written in the Devanagari script. It is a subject-object-verb (SOV) language and is morphologically rich. According to the 26^{th} edition of Ethnologue[1] published in 2023, Hindi is the third most spoken language having 609.5 million speakers. It is the official language of India and according to Forbes India[2], 44% of India speaks Hindi. However, the language models for the Hindi language lag behind due to the challenges mentioned below.

- The data collection has issues as there is limited availability of rich Hindi corpora, both for pre-training as well as fine-tuning.
- The complexity of Hindi Devanagari text needs to be taken into consideration. The intricacies of the Hindi script including conjunct characters and nuanced linguistic structures introduce complexity in Hindi text processing.
- The NLP model needs to understand the Hindi language in various contexts for tasks like sentiment analysis, inference and text summarization.

This work aims at the following contributions to tackle the above mentioned challenges.

- Our work focuses on training a tokenizer for the Hindi language using Byte Pair Encoding (BPE) tokenization algorithm[3].
- We train two autoregressive models for the Hindi language of different sizes.
- We perform supervised fine-tuning for multiple downstream tasks of the Hindi language.

The remainder of the paper is organized as follows. We discuss the existing works in Sect. 2. Section 3 explains the dataset used in this work. We discuss the methodology in Sect. 4. The performance evaluation is presented in Sect. 5. Finally, Sect. 6 concludes the work with future directions.

2 Related Work

There is a scarcity of research on the language models for Hindi language and other Indic languages. Hindi speakers are present all over the world and yet it is

[1] https://www.ethnologue.com/insights/ethnologue200/.
[2] https://www.forbesindia.com/article/news-by-numbers/hindi-day-2020-indias-mostspoken-languages-are/62577/1.
[3] https://huggingface.co/learn/nlp-course/en/chapter6/5.

less-researched in the field of NLP. One major reason behind this is the limited presence of Indic languages online in written form. Now, we discuss the related works present in the literature.

In the work by Arora et al. [4], the authors focused on pre-training language models for 13 Indic languages. They pre-trained ULMFiT [9] and TransformerXL [7] language models from scratch. The authors in Kakwani et al. [12] contributed multiple resources such as monolingual corpora, pre-trained word embeddings, IndicGLUE benchmark dataset and pre-trained language models. They used the ALBERT [15] model for pre-training language models. ALBERT, being a compact model, is lightweight and requires less training data which is good for less-resource languages.

Vries et al. [30] recycled the pre-trained GPT-2 [24] models for Italian and Dutch languages. They primarily retrained the lexical embeddings. First, they created a new BPE vocabulary for the target language. Then, they re-initialized the lexical embeddings of the GPT-2 model for the new vocabulary and re-trained them. They mentioned that the full model can be finetuned later with a smaller learning rate. This helps the model to better adjust to the new language whereas the re-learned lexical embeddings reduce the risk of information loss.

Owen et al. [19] discussed incremental pre-training for adapting English based LLMs to non-English languages such as Indonesian. First, they expanded the vocabulary by integrating a new trained tokenizer with the existing one. Then, they did incremental pre-training of Llama-2-7B-Base [27] using Low-Rank Adaptation (LORA) [10] technique. This helps the model to learn new language without catastrophically forgetting the English language in a minimal resource requirement setup. Owen et al. [19] and Vries et al. [30] discussed about utilizing English based pre-trained LLMs. Since these are pre-trained on English language, their techniques are not suitable for building the HindiLLM models because Hindi is very different from English and is written in Devanagari instead of Latin.

Niyogi et al. [18] trained multiple auto-regressive models from scratch for 10 Indian languages. Their models are also based on Transformers decoders. However, they did not provide a detailed description of the data used and the model architecture. Radford et al. [23] discussed semi-supervised training approach which is a two-step training process for the English language. The first step is generative pre-training on a large unlabeled corpus which gives a good initialization point for the next step, which is discriminative fine-tuning on each specific task. The GPT-1 is an auto-regressive transformer [29] based decoder only model. We utilized similar semi-supervised training approach for our HindiLLM models.

Although our work focuses on the Hindi language, we can apply the language model techniques across different languages. This will enable the less-researched languages to benefit by utilizing the techniques mentioned for extensively researched languages.

3 Dataset

The dataset generation is the first step of any model building process. The richness of the data determines the quality of the model. The approach used in this paper requires two types of data, unlabeled data for unsupervised pre-training and labeled data for supervised fine-tuning. The details of the datasets used in this paper are mentioned in this section.

3.1 Pre-training Dataset

For the unsupervised pre-training step, we need a large corpus with Hindi text written in Devanagari script. Since pre-training is the most crucial step which helps the model in understanding the language and its nuances, we need the corpus to be clean and have valid long paragraphs. This implies that it should not have unnecessary symbols, words, web-links and characters which we do not see in typical Hindi literature. A paragraph or sentence is valid if it follows the grammar of that language and it makes sense to the native speaker of that language. Long paragraphs are good for introducing long-term dependencies in the model. Generally, we need to scrap the internet for large corpus but this leads to difficulty in finding usable Hindi sentences or paragraphs. There exist projects which focus on getting web-crawled texts. In these projects, these corpora are classified based on language and are preprocessed following the structure of that language. We have used such existing corpora for the pre-training step.

Table 1. Detailed Description of Pre-training Corpora

Data	Size *(in GB)*	No. of Words *(approx. in million)*
Wikipedia	1.04	78.82
CC-Aligned	1.3	133.89
OSCAR-2201	14	1185.51
CC-100	21	1714.72

As shown in Table 1, we have 37.34 GB of data which contains approximately 3.11 billion words. Apart from CC-Aligned [8], all other datasets contain only Hindi text written in Devanagari script. The CC-Aligned dataset contains Hindi-English translation pairs of sentences and phrases. We have concatenated both versions (Hindi and English) of the sentence one after the other. One of the sentences from the translation pair is chosen randomly as the first sentence for each of the translation pairs. The idea here is to add some English capability in the model along with Hindi because we often see words or phrases of English inserted in Hindi texts. The translation pair will also help the model understand the relationship between both languages. Hindi Wikipedia articles are also used

from Kaggle[4] and Tensorflow[5]. Wikipedia articles contain factual information and the sentences are well formed as well, unlike some internet forums. The OSCAR-2201 [1] or Open Super-large Crawled Aggregated coRpus is a multilingual corpus intended for pre-training language models. It contains 151 different languages but we only use the Hindi texts. The CC-100 [6,31] is a monolingual corpus containing texts in 100+ languages out of which we only utilize the Hindi texts. It is generated using the open-source CC-Net [31] repository.

Apart from the default preprocessing done by the dataset creator, we have additionally performed the following preprocessing steps.

- **Content Cleanup** - Removal of content within brackets, hyperlinks, extra spaces and formatting of punctuation marks.
- **Filtering Sentence** - Sentences where less than 30% of words are unique are filtered out. These are not valid sentences because same few words are repeated multiple times.
- **Numeric and Punctuation Removal** - Removal of lines containing only numeric, punctuation marks or special symbols.
- **Handling Short Lines** - Multiple consecutive lines with less than fours words were removed. These were generally navigation links of a website.

3.2 Fine-Tuning Dataset

The fine-tuning is done for downstream tasks. The performance on downstream tasks tells about the real-world applicability of the model. In this paper, we have chosen seven downstream tasks to measure different aspects of our models.

Sentiment Analysis: We have two sentiment analysis datasets, namely IITP Movie[6] and IITP Product[7]. These datasets are public and widely used for evaluating Hindi language models. The dataset comprises three classes: positive, neutral, and negative. We have combined both datasets for training but tested separately.

Text Classification: For multiclass classification evaluation, we have used BBC News category[8] classification dataset. It has six categories. These are India, international, news, entertainment, sports, and science. It is also a public dataset which has been used for testing multiple Hindi based language models.

Natural Language Inference: We analyzed the natural language inference capability of our models with the BBC NLI [28] dataset.

[4] https://www.kaggle.com/datasets/disisbig/hindi-wikipedia-articles-172k.
[5] https://www.tensorflow.org/datasets/catalog/wikipedia.
[6] https://www.kaggle.com/datasets/warcoder/iit-patna-movie-reviews-hindi.
[7] https://www.kaggle.com/datasets/warcoder/iit-patna-product-reviews.
[8] https://github.com/NirantK/hindi2vec/releases/tag/bbc-hindi-v0.1.

Cloze-Style Multiple-Choice QA (CSQA): The CSQA dataset is from IndicGLUE [12] benchmark dataset. This dataset has a masked entity in a given text and we are given four candidate entities out of which one is the correct entity. This dataset is created from Wikipedia articles.

Wikipedia Section-Title Prediction (WSTP): Similar to CSQA, WSTP is from the IndicGLUE benchmark dataset and created using Wikipedia articles. The dataset has Wikipedia sections and it is required to find out the section title from the given four choices of titles.

Discourse Mode Classification (DM): The DM dataset is also from the IndicGLUE benchmark dataset. It is a discourse analysis dataset with five discourse categories: descriptive, narrative, dialogue, argumentative and informative. In this task, the model has to predict the suitable discourse category for a given sentence.

Machine Translation Dataset: We have obtained the translation dataset from Kunchukuttan et al. [14]. It has pairs of Hindi and English versions of 1.49 million sentences. This dataset is specifically created for the Hindi-English translation task. The machine translation is a generative downstream task.

4 Methodology

We follow two steps training process in this work. We apply an unsupervised pre-training to get the base model and a supervised fine-tuning of the base model for the downstream tasks. However, prior to the training, we build the HindiLLM tokenizer using the BPE algorithm. In this section, we have provided a detailed description of the tokenizer and the models along with the training process.

4.1 Tokenizer

Since we focus on building the model for the Hindi language and the default GPT-2 [24] model is primarily for the English language. So, we train a new tokenizer called HindiLLM tokenizer. The idea of training a custom tokenizer is to reduce the fertility score (average number of tokens per word) for the Hindi language. We have trained a Byte-level BPE tokenizer with our pre-training corpora which contains mostly Hindi language written in Devanagari script. Since the Devanagari script is complex, the Byte-level BPE tokenizer is the most suitable option. We have used the whole pre-training dataset to train the BPE tokenizer. We have kept the desired vocabulary size as 50000 while using the trainer to accommodate most of the frequently occurring sub-words. Also, we added 8 special tokens like CLS, SEP and PAD afterwards, so the vocabulary size reached to 50008.

We show the output of the HindiLLM tokenizer in Fig. 1. There are four sentences (three Hindi and one English) and its corresponding tokens as tokenized by our tokenizer. We can see that the words are split into multiple sub-words. For example, the word " आपका " is tokenized into " आपक " and " ा " but the word " आप " is a token on its own. The HindiLLM tokenizer is able to tokenize both Hindi and English sentences.

```
Sentence :   आपका स्वागत है।
Tokens   :   [' आपक', 'ा', ' स', '्', 'व', 'ा', 'गत', ' ह', 'ै।']

Sentence :   सपनों को हकीकत में बदलो।
Tokens   :   [' सपन', 'ों', ' क', 'ो', ' हक', 'ी', 'कत', ' म', 'ें', ' बदल', 'ो।']

Sentence :   अपने आप को पसंद करो।
Tokens   :   [' अपन', 'े', ' आप', ' क', 'ो', ' पस', 'ं', 'द', ' कर', 'ो।']

Sentence :   Actions speak louder than words.
Tokens   :   [' Action', 's', ' speak', ' l', 'ou', 'der', ' than', ' words', '.']
```

Fig. 1. Output of HindiLLM Tokenizer

To check if the HindiLLM tokenizer makes sense or not, we have taken a 100 words paragraph written in Hindi, encoded it using our own and the default GPT-2 tokenizer. We see that our tokenizer takes 345 tokens and the default GPT-2's tokenizer takes 785 tokens to represent the same paragraph with 100 words. It takes less than half the number of tokens for our HindiLLM tokenizer. This indicates that the HindiLLM tokenizer has lower fertility score for the Hindi language. Hence, we can pass larger Hindi sentences or paragraphs to the model. This will also improve the efficiency in processing Hindi text. Therefore, creation of a new tokenizer is beneficial here.

4.2 Unsupervised Pre-training

As mentioned in the GPT-1 [23] work, the pre-training finds a good initialization point for the model. In pre-training, the model learns about the language such as morphology and syntax. The Causal Language Modeling (CLM) is used in the unsupervised pre-training step of autoregressive models. This step gives us a base model that can be supervised fine-tuned for several downstream tasks with a relatively smaller dataset. We have trained two models, HindiLLM-Small and HindiLLM-Medium corresponding to GPT2-small and GPT2-medium respectively. We use Hugging Face's Transformers library [32] for pre-training. The training process involves creating a model with the configuration of corresponding GPT-2 models and training it after initializing with random weights.

Table 2. Training Details of HindiLLM-Small Model

Specification	Value
Corpus Size	19.6 GB
Number of Examples	6,904,242
Total Train Batch Size (w. Parallel, Distributed and Accumulation)	16
Total Optimization Steps	431,516
Number of Trainable Parameters	124,439,808
Context Window Size	1024 tokens
Number of Epochs	1.45
Optimizer	AdamW
Learning Rate	5e-05
GPUs Used	2 x NVIDIA A100-PCIE-40GB
Time Required	6 days

HindiLLM-Small Model: As shown in Table 2, the HindiLLM-Small model is equivalent to the GPT2-small model which has 124,439,808 trainable parameters. For training, we have taken 19.6 GB of data from our pre-training dataset. We have taken approximately half data from all corpora mentioned in Table 1. There are 6,904,242 examples for training a context window of 1024 tokens. We have used a batch size of 16 for training and updated the weights 431,516 times, which is equivalent to 1.45 epochs. The optimizer is a torch [20] based AdamW [16] optimizer with a learning rate 5e-05. Full precision training on two NVIDIA A100-PCIE-40GB GPUs have taken 6 days including the tokenization and evaluation steps. Instead of mixed-precision [17] training, we have opted for full precision training because we have trained from scratch.

HindiLLM-Medium Model: As depicted in Table 3, the HindiLLM-Medium model is based on the same configuration as of GPT2-medium model which has 354,823,168 trainable parameters. For training, we have 37.34 GB of data as mentioned in Table 1. There are 11,351,587 examples for training with the same context window as HindiLLM-Small. The total optimization steps are 354,737 with a batch size of 32 which is equal to 1.24 epochs. The optimization steps are less as compared to HindiLLM-Small because we have doubled the batch size. The optimizer and learning rate are the same as HindiLLM-Small. Similar to the HindiLLM-Small model, we have used two NVIDIA A100-PCIE-40GB GPUs for performing full precision training, which took 25 days including the tokenization and evaluation steps.

Table 3. Training Details of HindiLLM-Medium Model

Specification	Value
Corpus Size	37.34 GB
Number of Examples	11,351,587
Total Train Batch Size (w. Parallel, Distributed and Accumulation)	32
Total Optimization Steps	354,737
Number of Trainable Parameters	354,823,168
Context Window Size	1024 tokens
Number of Epochs	1.24
Optimizer	AdamW
Learning Rate	5e-05
GPUs Used	2 x NVIDIA A100-PCIE-40GB
Time Required	25 days

Performance Evaluation of Pre-trained Models: As shown in Table 4, the HindiLLM-Medium is better than HindiLLM-Small in terms of all the metrics. The larger model has higher accuracy, lower loss and lower perplexity as compared to the smaller one. The perplexity of 3.686 and 3.0017 for HindiLLM-Small and HindiLLM-Medium, respectively, assures the quality of the training.

Table 4. Performance of the Pre-trained HindiLLM Models

Metrics	HindiLLM-Small	HindiLLM-Medium
Evaluation Accuracy	0.6855	0.7300
Evaluation Loss	1.3045	1.0992
Perplexity	3.6860	3.0017
Train Loss	1.3849	1.2096

4.3 Supervised Fine-Tuning

The next step in the semi-supervised training approach is the Supervised Fine-Tuning (SFT) on discriminative or generative tasks. The fine-tuning step aligns the model with the downstream task. The previous pre-training step makes it easier to fine-tune because it has already gained knowledge about the language. Hence, we can achieve higher performance on downstream tasks even with a smaller dataset. The model can be fine-tuned for variety of tasks. Here, we have fine-tuned for seven tasks on the datasets mentioned in Sect. 3.2. Since the model

will be used on real-world downstream applications, the SFT and evaluation of the resultant model are the crucial steps.

We have used deepspeed [25] library for fine-tuning efficiently using multi-GPU setup. The SFT is done in bfloat16 precision format with a learning rate of 5e-6.

5 Performance Evaluation

In this section, the performances on the downstream tasks are evaluated. We have done SFT for a variety of downstream tasks to get a detailed performance measure of the HindiLLM models. The results are compared with other models to validate its improvement.

5.1 Public Classification Dataset

As shown in Table 5, we compare various models on public classification datasets. We have accuracy scores from Wikipedia (FT-W) [5], Wiki+CommonCrawl (FT-WC) [22], IndicFT [12], IndicBERT [12], mBERT [21], XLM-R [26], INLP [13] and iNLTK [4] models. Also, we have obtained scores from the GPT-3.5 Turbo model. For the GPT-3.5 Turbo model, we have considered zero-shot prompting and few-shot prompting. We have performed prompt engineering to find the best system prompt and in the case of few-shot prompting, we have given five random examples from the training data. We can observe that the HindiLLM-Medium model surpasses all the models on all three datasets: IITP-Movie, IITP-Product, and BBC-Article public classification datasets. The HindiLLM-Small model comes second in the case of IITP-Movie dataset. It does not perform well

Table 5. Classification Accuracy on Public Dataset

Model	IITP-Movie	IITP-Product	BBC-Article
HindiLLM-Small	70.51	76.63	71.29
HindiLLM-Medium	**78.34**	**79.31**	**81.04**
FT-W	41.61	58.32	72.29
FT-WC	44.52	57.17	67.44
IndicFT	45.81	61.57	77.02
IndicBERT-Base	59.03	71.32	74.60
mBERT	56.77	74.57	60.55
XLM-R	61.61	78.97	75.52
INLP	45.81	63.48	74.25
iNLTK	57.74	75.71	78.75
GPT-3.5 Turbo Zero-shot	68.17	68.20	56.41
GPT-3.5 Turbo Few-shot	66.45	72.92	49.63

on the BBC-Article dataset. For IITP-Movie dataset, we see an improvement of 2.34% and 10.17% in HindiLLM-Small and HindiLLM-Medium models, respectively. For IITP-Product dataset, we see a decrease of 2.34% and an increase of 0.34% in HindiLLM-Small and HindiLLM-Medium models, respectively. We observe an improvement of 2.29% for HindiLLM-Medium model in BBC-Article dataset, but a drop of 7.46% for HindiLLM-Small. Both the zero-shot and few-shot results from GPT-3.5 Turbo are poorer than both of our models. In most cases, the result of few-shot is worse than the zero-shot. This is possibly because the given examples are confusing the model or because with a increase in the prompt length, the model is not able to analyze accurately.

5.2 IndicGLUE Benchmark Dataset

Table 6. Accuracy Score on IndicGLUE Benchmark Dataset

Model	CSQA	WSTP	DM
HindiLLM-Small	38.53	69.85	78.68
HindiLLM-Medium	44.71	77.19	**80.48**
XLM-R	30.62	76.92	79.94
mBERT	39.00	**80.12**	71.20
IndicBERT-Base	41.55	74.02	78.44
IndicBERT-Large	37.01	77.80	NA
GPT-3.5 Turbo Zero-shot	44.56	76.75	50.91
GPT-3.5 Turbo Few-shot	**50.84**	74.25	48.89

Table 6 shows the accuracy score achieved on the IndicGLUE benchmark dataset. We have considered XLM-R, mBERT and IndicBERT models for comparison along with zero-shot and few-shot prompting of GPT-3.5 Turbo. In the CSQA task, the GPT-3.5 Turbo Few-shot has the highest accuracy and our HindiLLM-Medium model has the second best accuracy which is a drop of 6.13%. HindiLLM-Small has the sixth best result with a drop of 12.31% on this task. For the WSTP task, mBERT shows the best result whereas we see a drop of 2.93% and 10.27% for HindiLLM-Medium and HindiLLM-Small, respectively. For the DM task, HindiLLM-Medium gives the best accuracy with an improvement of 0.54%, followed by XLM-R and HindiLLM-Small. We do not have any score from IndicBERT-Large model on this task.

5.3 Comparison with GPT-2 Models

Table 7. Sentiment Analysis Comparison with Internally Fine-tuned GPT-2 Models

Model	Precison	Recall	F1-score
HindiLLM-Small	77.5	75.22	76.34
HindiLLM-Medium	**80.6**	**79.18**	**79.88**
GPT2-Small	56.77	44.47	49.87
GPT2-Medium	62.84	63.89	63.36

Table 8. Natural Language Inference Comparison with Internally Fine-tuned GPT-2 Models

Model	Precison	Recall	F1-score
HindiLLM-Small	97.16	97.24	97.20
HindiLLM-Medium	**98.03**	**99.18**	**98.60**
GPT2-Small	70.23	69.61	69.92
GPT2-Medium	70.75	69.35	70.04

We have fine-tuned the default GPT-2 [24] along with our models for the Sentiment Analysis dataset (a combination of IITP-Movie and IITP-Product datasets) and BBC-NLI dataset. We have used the same train-test split for fine-tuning both GPT-2 and HindiLLM model. The fine-tuning process was the same for both kinds of models. From the results shown in Table 7 and Table 8, it is evident that there is a huge improvement in the performance on Hindi downstream tasks using HindiLLM models. Even the smaller HindiLLM-Small model surpasses the scores of GPT2-Medium models by a large margin.

5.4 Machine Translation Dataset

Table 9. Human Evaluation Criteria

Quality	Rating
Excellent	4
Good	3
Understandable	2
Barely understandable	1
Incomprehensible	0

In machine translation, we have considered both English-to-Hindi and Hindi-to-English translation tasks using the same set of train and test data. For machine translation, we have only considered the HindiLLM-Medium model. The smaller model will struggle a bit here because it is a generative task. Since the Hindi language is morphologically rich, it is unfair to use traditional metrics such as BLEU and METEOR. Hence, we have performed human evaluation of the translations using the criteria mentioned in Table 9.

Table 10. HindiLLM-Medium model on Machine Translation Dataset

Metric	Hindi to English	English to Hindi
Score 4	6.89%	5.91%
Score 3	28.83%	44.93%
Score 2	28.67%	29.47%
Score 1	31.40%	17.81%
Score 0	4.21%	1.88%
Mean Score	2.03	2.35

Table 10 shows the performance of the machine translation task. We have shown the probability distribution of the scores. The English to Hindi task performs slightly better than the Hindi to English task. We have a small portion of data for score 0, which is good. But for score 4 as well we have a small portion of data. Even when the score is 3, the translation quality is good. We see that a substantial portion of data for the Hindi to English task and a major portion of data for the English to Hindi task have a score of 3. It indicates that the translation quality is good. The mean scores are above average. Considering that we have used limited English data in pre-training, the results are promising.

6 Conclusion

In this paper, we train a tokenizer and two auto-regressive models of different sizes for the Hindi language written in Devanagari script. We check the validity of the models by comparing the results on multiple downstream tasks. By looking at the performances, we conclude that the pre-trained models are well-trained to handle a variety of downstream tasks.

As we see the performance evaluation of downstream tasks, in most of the cases our HindiLLM-Medium model shows the best results. The HindiLLM-Small model lags because of its smaller size and smaller pre-training data. It is clear from the results that HindiLLM will contribute to solving real-world problems, especially the HindiLLM-Medium model. We also note that our HindiLLM models perform better than the fine-tuned GPT-2 and prompt engineering GPT-3.5 Turbo model. This implies that training a language-specific model will result in better performance for that particular language even with a smaller model. Also, it is evident from the results that training a larger model on a larger dataset will result in a better performing model.

Even though the models show impressive results, there is scope for further improvements. The number of epochs during training can be further increased. The training can be performed with enhanced data like the text from the books. Training a larger model on larger pre-training data will always result in a better model. Our model has limited knowledge of English since a small portion of our dataset comprises English data. The English data can be increased to enhance the bilingual capability. Furthermore, adding a few more supervised fine-tuning tasks can add to the assurance of the quality of the model.

In the future, we plan to create models by combining Hindi, Romanized Hindi (Hinglish), and English data. Adding Hinglish data can make it more relevant in day-to-day applications. We see frequent use of Hinglish these days instead of Hindi and it is rapidly gaining popularity. A few such examples are comment sections, posts and messages on social networking sites. Further, adding more English texts to the pre-training data will make it bilingual. This will increase its usability in tasks like machine translation and tasks that contain a mix of Hindi and English data.

References

1. Abadji, J., Suarez, P.O., Romary, L., Sagot, B.: Towards a cleaner document-oriented multilingual crawled corpus. arXiv preprint arXiv:2201.06642 (2022)
2. Achiam, J., et al.: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
3. Anil, R., et al.: Palm 2 technical report. arXiv preprint arXiv:2305.10403 (2023)
4. Arora, G.: inltk: Natural language toolkit for indic languages. arXiv preprint arXiv:2009.12534 (2020)
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
6. Conneau, A., et al.: Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116 (2019)

7. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
8. El-Kishky, A., Chaudhary, V., Guzmán, F., Koehn, P.: Ccaligned: a massive collection of cross-lingual web-document pairs. arXiv preprint arXiv:1911.06154 (2019)
9. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146 (2018)
10. Hu, E.J., et al.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
11. Jiang, A.Q., et al.: Mistral 7B. arXiv preprint arXiv:2310.06825 (2023)
12. Kakwani, D., et al.: Indicnlpsuite: monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In: Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 4948–4961 (2020)
13. Kunchukuttan, A., et al.: Ai4bharat-indicnlp corpus: monolingual corpora and word embeddings for indic languages. arXiv preprint arXiv:2005.00085 (2020)
14. Kunchukuttan, A., Mehta, P., Bhattacharyya, P.: The IIT Bombay English-Hindi parallel corpus. arXiv preprint arXiv:1710.02855 (2017)
15. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: a lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
16. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
17. Micikevicius, P., et al.: Mixed precision training. arXiv preprint arXiv:1710.03740 (2017)
18. Niyogi, M., Bhattacharya, A.: Paramanu: a family of novel efficient indic generative foundation language models. arXiv preprint arXiv:2401.18034 (2024)
19. Owen, L., Tripathi, V., Kumar, A., Ahmed, B.: Komodo: a linguistic expedition into Indonesia's regional languages. arXiv preprint arXiv:2403.09362 (2024)
20. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
21. Pires, T., Schlinger, E., Garrette, D.: How multilingual is multilingual bert? arXiv preprint arXiv:1906.01502 (2019)
22. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: a python natural language processing toolkit for many human languages. arXiv preprint arXiv:2003.07082 (2020)
23. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
24. Radford, A., et al.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
25. Rasley, J., Rajbhandari, S., Ruwase, O., He, Y.: Deepspeed: system optimizations enable training deep learning models with over 100 billion parameters. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3505–3506 (2020)
26. Ruder, S., Søgaard, A., Vulić, I.: Unsupervised cross-lingual representation learning. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts, pp. 31–38 (2019)

27. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
28. Uppal, S., et al.: Two-step classification using recasted data for low resource settings. In: Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pp. 706–719 (2020)
29. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
30. de Vries, W., Nissim, M.: As good as new. how to successfully recycle English GPT-2 to make models for other languages. arXiv preprint arXiv:2012.05628 (2020)
31. Wenzek, G., et al.: Ccnet: extracting high quality monolingual datasets from web crawl data. arXiv preprint arXiv:1911.00359 (2019)
32. Wolf, T., et al.: Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45 (2020)

StableTalk: Advancing Audio-to-Talking Face Generation with Stable Diffusion and Vision Transformer

Fatemeh Nazarieh[1(✉)], Josef Kittler[1,2,3], Muhammad Awais Rana[1,2,3], Diptesh Kanojia[1,3], and Zhenhua Feng[1,4]

[1] School of Computer Science and Electronic Engineering, University of Surrey, Guildford, UK
{f.nazarieh,j.kittler,muhammad.awais,d.kanojia}@surrey.ac.uk, fengzhenhua@jiangnan.edu.cn
[2] Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, UK
[3] Institute for People-Centred AI, University of Surrey, Guildford, UK
[4] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, China

Abstract. Audio-to-talking face generation stands at the forefront of advancements in generative AI. It bridges the gap between audio and visual representations by generating synchronized and realistic talking faces. Despite recent progress, the lack of realism in animated faces, asynchronous audio-lip movements, and computational burden remain key barriers to practical applications. To address these challenges, we introduce a novel approach, **StableTalk**, leveraging the emerging capabilities of **Stable** diffusion models and vision Transformers for **Talk**ing face generation. We also integrate the Re-attention mechanism and adversarial loss to improve the consistency of facial animations and synchronization with a given audio input. More importantly, the computational efficiency of our method has been notably enhanced by optimizing operations within the latent space and dynamically adjusting the focus on different parts of the visual content based on the provided conditions. Our experimental results demonstrate the superiority of StableTalk over the existing approaches in image quality, audio-lip synchronization, and computational efficiency.

Keywords: Latent Diffusion · Vision Transformer · Audio-to-Talking Face Generation · Re-attention · Denoising Diffusion Implicit Model

1 Introduction

Audio-to-talking face generation aims to generate talking face videos based on the provided speech and a reference image, a sub-task of cross-modal visual content generation. Audio-to-talking face generation has various applications,

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 271–286, 2025.
https://doi.org/10.1007/978-3-031-78172-8_18

such as animation in the entertainment industry, video dubbing for different languages, and generating talking avatars for human-computer interaction. The task involves extracting audio information and learning a mapping function from the input audio to its corresponding lip movement. Achieving precise audio-lip synchronization, generating realistic content, and ensuring spatio-temporal consistency, present significant challenges to audio-to-talking face generation [20].

To overcome these challenges, deep-learning-based models have been widely used in the existing literature [30,32,38,41]. Suwajanakorn et al. [32] proposed a Recurrent Neural Network (RNN-) based method to generate talking faces of President Obama. This model operates on the video of a target person and modifies the lip movements to be synced with the input audio. Despite its promising performance, this model is limited to generating only a single identity. To mitigate this issue and reduce identity dependency, generative models use facial landmarks as the primary condition [5]. One of the pioneering studies in this direction is MakeItTalk [41] which uses speech content and speaker-aware animation modules to map speech signals to facial landmarks. Despite the improvement in generalization capability, the generated videos by MakeItTalk lack finer details and often fail to accurately match lip movements with spoken words, resulting in animated talking faces that appear artificial.

Visual content generation has achieved promising results with the emergence of Generative Adversarial Networks (GANs) [8]. Wav2Lip is among the first GAN-based approaches for audio-to-talking face generation. Taking advantage of a pre-trained discriminator, Wav2Lip achieves audio-lip synchronization for an arbitrary input speech. Despite the improvements achieved by GANs, they usually suffer from unstable training and model collapse issues [16,17,25]. The generated images by GANs often present artifacts and struggle to maintain smooth transitions among adjacent frames [15].

Recently, Denoising Diffusion Probabilistic Models (DDPMs) [11] outperform GANs and are marked as state-of-the-art in visual content generation. DiffTalk [28] and DiffusedHeads [31] are among the first diffusion-based models for audio-to-talking face generation. DiffusedHeads [31] generates realistic head movements and eye blinks by combining motion frames with audio features for temporal stability. Similarly, Shen et al. proposed DiffTalk [28] that enables identity-aware generation by leveraging the latent features and facial landmarks, achieving a significant advancement in realistic talking face generation. The core difference between DiffusedHeads and DiffTalk lies in how the diffusion model is employed. DiffusedHeads is based on a standard diffusion model [11]. In contrast, DiffTalk is based on a latent diffusion model [23]. This modification of DiffTalk achieves enhanced visual quality compared to DiffusedHeads.

In this paper, inspired by the promising performance of diffusion-based models, we develop StableTalk, a novel approach that leverages Vision Transformer (ViT) [7] as the backbone for the reverse diffusion steps in latent diffusion (also known as stable diffusion). Specifically, we use a vision Transformer to replace the frequently used UNet [24]. The core objective of this modification is to accurately learn the intricate relationship between the input speech and associated

facial characteristics, including lip synchronization and corresponding expressions. Traditional approaches such as UNet adhere to this learning process by capturing local features in the early layers, and gradually building towards a more global context in the deeper layers. This progression relies heavily on the down-sampling and up-sampling operations, which can introduce potential bottlenecks. Such bottlenecks are particularly problematic for cross-modal features in audio-to-talking face generation, which demands significant computational resources and time. In contrast, Transformers capture both local and global contexts simultaneously. This outcome is achieved without the sequential, layer-by-layer processing that UNet requires. It eliminates the need for multiple layers to gradually expand the receptive field, thereby enhancing the computational efficiency of our StableTalk model [12,24].

Audio-lip alignment is another challenge and solely increasing the number of Transformer blocks is inefficient in modeling the relationship between these two modalities. When increasing the number of Transformer blocks, multiple attention heads often learn redundant and highly similar attention patterns. This reduces the overall realism and coherency between the input and the controlled modality. To mitigate this limitation, we employ the Re-attention [40] mechanism in StableTalk. The Re-attention mechanism introduces cross-head communication, allowing different attention heads to share information and adjust their focusing regions accordingly [4]. Further by utilizing cross Re-attention in our denoising module, cross-modal patches are processed in parallel. This approach efficiently analyzes the inter-relationships between audio and visual features in a single operation and minimizes the resources required to process less significant areas. Last, to further enhance the performance of the proposed StableTalk model, we employ adversarial loss to enhance the training process. The adversarial component can help refine the generation of fine facial movements, making the expressions natural and expressive [2,8]. In summary, the main contributions of the proposed StableTalk method are:

- A novel approach that leverages vision Transformer as the denoising component within the latent diffusion architecture.
- Introducing Re-attention into StableTalk, which improves the capacity of our model to identify the relationships and patterns across the audio and video modalities.
- An investigation of the impact of adversarial loss in talking face generation. We demonstrate that adversarial loss enhances the realism, synchronization, and expressiveness of the generated talking face videos.
- A notable enhancement in inference speed without sacrificing the quality of the generated output.

The remainder of this paper is structured as follows: Sect. 2 presents the proposed StableTalk framework in detail. The experimental results are reported and analyzed in Sect. 3. In Sect. 4, we summarize our key findings, discuss the implications of our work, and propose potential directions for future research.

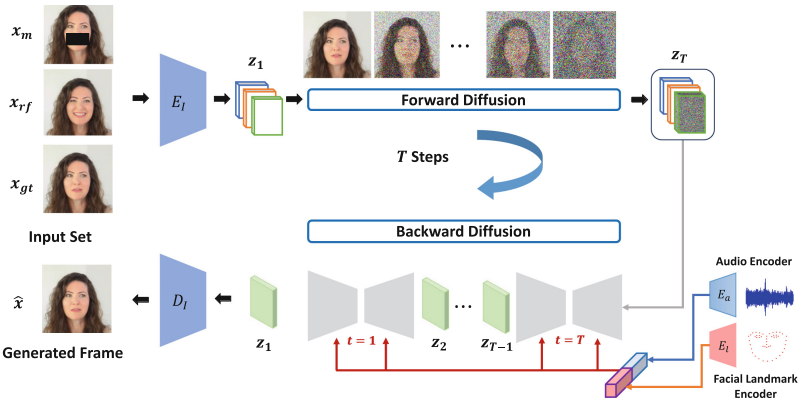


Fig. 1. The overall architecture of StableTalk.

2 The Proposed StableTalk Method

In this section, we introduce the proposed StableTalk model designed for efficient talking face generation with visual reality, contextual coherence, and audio-lip synchronization.

2.1 Overview of StableTalk

An overview of StableTalk is shown in Fig. 1, which is based on DiffTalk [28]. However, the proposed method includes several innovations in the denoising module and training objectives, which will be introduced in detail later.

The training of StableTalk initiates by processing the reference frame ($\mathbf{x}_{rf} \in \mathbb{R}^{H\times W\times 3}$), ground truth frame ($\mathbf{x}_{gt} \in \mathbb{R}^{H\times W\times 3}$), and masked ground truth frame ($\mathbf{x}_m \in \mathbb{R}^{H\times W\times 3}$) using a variational autoencoder [23] as the image encoder (E_I), where W and H are the width and height of an image. The use of these three frames is essential for network training as each provides crucial information for the generation process. The reference frame introduces identity information, enabling the model to modify its outputs to match the specific characteristics of the provided identity. The ground truth frame provides supervision signals for the training of the model. Note that, the ground-truth frame is not used in the inference stage, which is presented in Sect. 2.5. The masked ground truth frame ensures that the model learns facial features related to lip movements from audio, rather than memorizing the ground truth frame. Each person demonstrates various head poses when speaking the same sentence. Utilizing the masked ground truth frame provides guidance in determining the head pose during the generation process.

These encoded features are downsampled by a factor of f from the original image resolution and concatenated along the channel dimension to form the

input for the forward diffusion step ($z_{i\in[1,\ldots,T]} \in \mathbb{R}^{H/f\times W/f\times 12}$). Note that we only apply forward diffusion to the features obtained from the ground truth frame. The features obtained from the reference and masked frames remain the same during the forward diffusion stage. Meanwhile, audio and face landmarks are utilized as the condition sets (C) to guide the generation process in the backward diffusion step. Note that the facial landmarks do not include the mouth area. Section 2.2 presents the details of condition set processing.

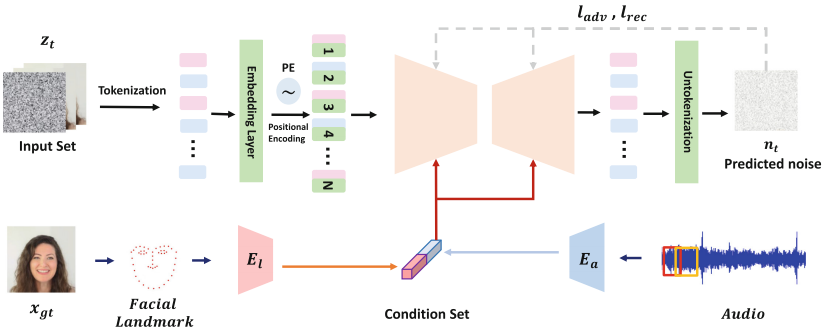


Fig. 2. The proposed denoising module.

In the forward step, the structure of the latent representation of the ground truth frame is corrupted by gradually incorporating Gaussian noises, using a scheduler [6]. In the reverse process, the denoising module predicts the amount of noise added at each time step t. This process equips the generative model with the understanding of latent representation at varying degrees of noise, enabling the progressive restoration of the frame from its noised state. Different from DiffTalk, the denoising steps in our model use ViT instead of a time-conditional UNet [28]. Upon receiving the noised latent representations and condition sets, ViT employs a series of cross-attention mechanisms to analyze and interpret the data. Each latent representation is tokenized, normalized, and added with positional encoding before initiating the denoising module, as depicted in Fig. 2. This allows ViT to effectively capture the patterns and correlations across the entire image. The reverse process terminates by reconstructing the frame from the denoised latent representations using the decoder (D_I) of the variational autoencoder [23]. Note that the predicted noise is compared against the added noise in the latent space, not after the image reconstruction step. The loss functions for network training are introduced in Sect. 2.4.

2.2 The Condition Set

In audio-to-talking face generation, the quality and coherence of the generated frames are significantly influenced by the nature of guiding factors, commonly referred to as the 'condition set.' In this paper, we use audio and face landmarks

as conditioning criteria, each requiring careful pre-processing to ensure optimal model performance.

Prior to the generative stage, an audio input is divided into overlapping segments with a window length of $500ms$. Each part is further processed by the wav2vec Transformer [26] (E_a), resulting in rich temporal audio representations per frame (c_a). Wav2vec is a pre-trained self-supervised framework for speech representation learning, which is scalable and efficient for various audio processing tasks [27,35]. We choose wav2vec due to its ability for contextual understanding and adaptability on many audio-related downstream tasks. This is different from DiffTalk which uses RNN for audio feature extraction.

The preprocessing of facial landmarks is equally crucial. The landmarks provide necessary geometric references for our model to generate realistic and contextually aligned facial movements. To ensure proper training and prevent learning shortcuts, the mouth landmarks have been removed. The remaining points are further processed using a CNN-based landmark encoder (E_l) [9]. In the final step, the extracted audio features (c_a) and encoded facial landmark features (c_l) are concatenated as the condition to guide the generation process.

2.3 The Cross-Attention Mechanism

Audio-lip alignment is challenging. However, exploring possible patterns between these two modalities by increasing the number of Transformer blocks is less effective. For the standard ViT architecture, this may lead to attention collapse issues [40]. This problem occurs when different attention heads of a Transformer end up learning redundant and similar attention patterns, indicating limited ability to capture diverse features and representation learning [13]. To mitigate this problem, we integrate the Re-attention mechanism [40] into the ViT-based denoising block of StableTalk. This approach enables the generative model to capture diverse aspects of the input data.

The key innovation of the Re-attention mechanism is the introduction of cross-head communication. This mechanism ensures each head is aware of the focused areas of other heads, discouraging them from all attending to the same features [13]. Consequently, the finalized cross Re-attention mechanism is obtained by:

$$Re - attention(Q, K, V) = Norm(\theta^T(softmax(\frac{QK^T}{\sqrt{d_q}})))V \tag{1}$$

where Q, K and V represent the Query, Key and Value matrices. In Eq. (1), Q carries visual information while K and V carry information related to the condition set. θ represents the learnable transformation matrix, serving for cross-head communication. The $Norm$ function assists in stabilizing the training by normalizing the attention maps and reducing the layer-wise variance. The multiplication between the transformation matrix and self-attention map is performed across the head dimension. Considering the learnable nature of the transformation matrix, it can be adjusted during the training process. This adaptabil-

ity allows the model to optimize how information is shared among different heads [4,40].

2.4 Loss Functions

We utilize two loss functions for model training: reconstruction loss and adversarial loss. The employed reconstruction loss is presented as follows:

$$L_{rec} = \mathbb{E}_{(z,\epsilon\sim N(0,1),t,C)}[\|\epsilon - \epsilon_\theta(z_t, t, C)\|_2^2], \tag{2}$$

where ϵ_θ represents the denoising model which predicts the amount of noise (ϵ) added to z_t. Here, z_t is the nosiy version of input (z) and t represents the diffusion time steps ($t \in [1, 2, .., T]$). Further, both latent (z) and noise (ϵ) are sampled from a Gaussian distribution and the condition set is represented by C ($C = \{c_a, c_l\}$). While the reconstruction loss ensures fidelity to the input data, it can sometimes lead to smooth or blurry generated frames [1,34]. To further refine the quality and synchronization of audio with the generated talking face videos, we incorporate the adversarial loss for network training.

We draw inspiration from the discriminator architecture of Wav2Lip [22] which is renowned for its successful generalization on arbitrary audio and identities. Contrary to the Wav2Lip model, which directly manipulates the visual domain, our approach employs the discriminator within the latent space. This adopted approach aims to guide the generator towards accurately approximating the actual noise distribution. By effectively predicting and then subtracting the appropriate amount of noise during the reconstruction phase, the model generates talking faces that closely resemble the ground truth. Our model leverages this aspect by introducing the adversarial loss (l_{adv}) as follows:

$$L_{adv} = \mathbb{E}_{(z,t,C)}[\log(1 - D(\epsilon_\theta(z_t, t, C)))] + \mathbb{E}_{(\epsilon\sim N(0,1))}[\log D(\epsilon)] \tag{3}$$

where $D()$ indicates the lip-sync discriminator, which tries to distinguish between real noise (ϵ) and the noise predicted by the model ($\epsilon_\theta(z_t, t, C)$). As previously described, z_t represents the noisy version of data (z), t represents time steps in the diffusion process, and C refers to the employed condition set. The term $\log(1 - D(\epsilon_\theta(z_t, t, C)))$ computes the logarithm of the probability that the predicted noise is classified as fake. The term $\log D(\epsilon)$ computes the logarithm of the probability that the real noise is classified as real. The goal is to minimize these terms to effectively reduce the probability that its predicted noise is recognized as fake by the discriminator.

The final loss function is defined as:

$$L_{final} = \lambda_1 * L_{rec} + \lambda_2 * L_{adv}, \tag{4}$$

where λ_1 and λ_2 are balancing parameters, allowing us to regulate which loss the model prioritizes during the training process. The final loss function, L_{final}, is applied within the latent space to optimize the performance of the model in generating talking faces from audio inputs.

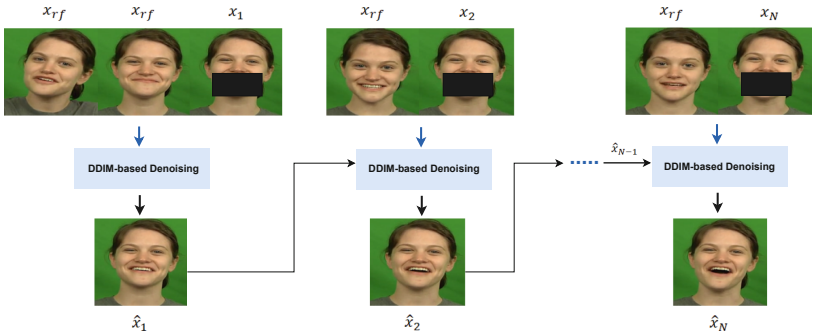


Fig. 3. The inference stage of StableTalk.

2.5 The Inference Stage

As discussed in Sect. 2.1, StableTalk uses the ground truth frame, masked ground truth frame, and reference frame as inputs during the training stage. However, the ground truth frame is not used in the inference step. Our inference pipeline adopts a sequential, frame-by-frame generation approach, where the output generated at each time step serves as the foundation for the generation of the next frame. Given the existence of no prior generation at the initial step, two random reference frames (x_{rf}) and the masked frame (x_m) are employed to initiate the process. The employed masked frame at each time step (t) is the masked version of the ground truth frame at that specific time step. The choice to utilize two random reference frames aims to enrich the generative process with varied visual contexts. This strategy compensates for the absence of prior frames and assists the model in generating an initial frame that is both contextually rich and faithful to the identity of the reference identity. As the pipeline progresses, random reference frame (x_{rf}), generated frame from the previous step ($\hat{x}_{t-1} \in \mathbb{R}^{(H,W,3)}$) and masked frame ($x_m$) are utilized for the iterative frame generation. The use of a generated frame from the previous step ensures that each new frame builds upon the visual information from the last generated frame. This approach leads to maintaining coherency and continuity among the generated frames of a video. The inference pipeline of the StableTalk is illustrated in Fig. 3.

To enhance the efficiency and stability of the inference stage and inspired by DiffTalk, we replace the standard Denoising Diffusion Probabilistic Model (DDPM [6]) with the Denoising Diffusion Implicit Model (DDIM [29]). In contrast to DDPM, which executes all the diffusion time steps in a sequential manner to progressively eliminate noise, DDIM introduces a non-Markovian process. This process allows for a reduction in the number of steps required in the reverse diffusion process without compromising the quality of the generated samples. The sequential nature of DDPM prevents the process from being parallelized, resulting in slower generation, and presenting limitations in inference speed. On the other hand, DDIM addresses this by utilizing a deterministic approach that allows for skipping steps by leveraging information from multiple previ-

ous states [18,36]. This change significantly accelerates the generation process, making it more practical for real-time applications.

3 Experimental Results

3.1 Experimental Settings

Dataset. To evaluate the proposed StableTalk model, we use the Crowdsourced Emotional Multimodal Actors Database (CREMA-D [3]). CREMA-D is a comprehensive dataset comprising 7,442 video clips. These videos are performed by 91 actors and encompass a diverse spectrum of emotions including Anger, Disgust, Fear, Happiness, Neutral, and Sadness. We randomly select 80 percent of videos for training and the remaining part is used for testing.

Evaluation Metrics. To compare the performance of the proposed method with the existing approaches, we use four evaluation metrics, including Fréchet Inception Distance (FID [10]), Learned Perceptual Image Patch Similarity (LPIPS [39]), Peak Signal-to-Noise Ratio (PSNR [37]), and Structural Similarity Index Measure (SSIM [33]). We also evaluate the inference speed of different methods in Frames Per Second (FPS).

FID assesses the similarity between the generated and ground truth images based on the features extracted by the Inception net. Lower FID scores indicate better generation quality, with a closer resemblance to real image distributions [10]. LPIPS evaluates the perceptual difference between pairs of images using deep network features (e.g. VGG) to better measure the visual similarity. A lower LPIPS score suggests a higher perceptual similarity [39]. PSNR is a measure used to assess the quality of generated images compared to ground truth images. Higher PSNR values indicate better image quality [37]. SSIM measures the visual quality of images by comparing their structural information. A higher SSIM score indicates a greater similarity between the images being compared, implying less distortion and better image quality [33].

Implementation Details. Our model is designed to process videos resized to a resolution of 128×128, and the downsampling factor (f) is set to 8. This resizing facilitates efficient processing while preserving sufficient detail for high-quality video generation. During the training stage we set the diffusion time steps to 1000 and at test time this number is reduced to 200. The transformer configuration comprises 12 Transformer blocks. Our training procedure utilizes two NVIDIA A100 GPUs. Moreover, we utilize the AdamW optimization algorithm, a modification of the original Adam optimizer that introduces weight decay for better regularization and stability during training.

Table 1. A comparison between StableTalk and the other SOTA methods.

Model	FID↓	**SSIM↑**	**PSNR↑**	**LPIPS↓**
MakeItTalk [41]	19.483	0.4520	13.568	0.9207
Wav2Lip [22]	12.916	0.479	17.142	0.4402
DiffusedHeads [31]	12.450	0.545	21.803	0.104
DiffTalk [28]	12.459	0.674	23.991	0.094
StableTalk	**11.391**	**0.721**	**25.267**	**0.0703**

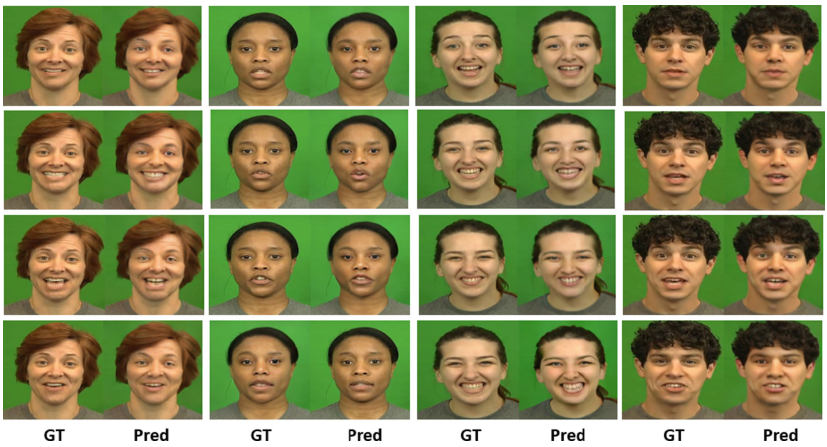


Fig. 4. A comparison between the ground truth (GT) and generated (Pred) frames by StableTalk.

3.2 Qualitative and Quantitative Results

In this section, we compare the performance of our StableTalk model with the state-of-the-art talking face generation methods, including MakeItTalk [41], Wav2Lip [22], DiffusedHeads [31], and DiffTalk [28]. The results are reported in Table 1, which shows that despite advancements made by existing methods such as MakeItTalk and Wav2Lip, they exhibit limitations in generating highly realistic talking faces when benchmarked against the ground truth frames. On the other hand, the introduction of diffusion-based models, DiffusedHeads and DiffTalk, marks a significant improvement in audio-to-talking face generation, particularly in terms of LPIPS and PSNR metrics. This advancement suggests a closer alignment with human perceptual judgments.

The proposed StableTalk model further refines these improvements by achieving the lowest FID (11.391) and LPIPS (0.0703), and highest SSIM (0.721) and PSNR (25.267). The superiority of StableTalk can be attributed to multiple factors including Transformer-based architecture and the employed training loss. In our pipeline, unlike previous works, we leverage both reconstruction loss and adversarial loss within a latent diffusion model. By employing a reconstruc-

tion loss generative model achieves better pixel-level accuracy and fidelity to the actual images. Complementing this, adversarial loss plays a pivotal role in enhancing the realism and perceptual quality of the generated faces. Based on the obtained results by StableTalk, the integration of reconstruction and adversarial loss leads to generating realistic talking faces that accurately reflect human expressions. Moreover, the employed cross Re-attention mechanism inside the denoising Transformer attends to interactions between input patches, regardless of their spatial proximity. This characteristic enables the model to identify patterns and features across provided modalities.

Fig. 5. A visual comparison between StableTalk and SOTA methods is presented using frames generated for the same identity as the basis for evaluation.

A visual comparison between the ground truth and generated frames by StableTalk is provided in Fig. 4. The figure indicates the capability of our model to produce realistic facial expressions coupled with aligned lip synchronization. This fidelity to the ground truth signifies the understanding of human facial dynamics by our generative model.

To further emphasize the improved performance of StableTalk, Fig. 5 presents a comparison between the outputs generated by our model and those produced by MakeItTalk, Wav2Lip, DiffusedHeads, and DiffTalk. We observe that, while MakeItTalk successfully animates faces to deliver the given audio, it often fails to convey synchronized facial expressions, resulting in predominantly neutral appearances. Generated talking faces by wav2lip accurately reflect the ground truth facial expressions, however, they fail to maintain realistic talking faces due to the rectangular box appearing around mouth regions. DiffusedHeads,

DiffTalk, and StableTalk notably enhance the *realism* and synchronization of lip movements with the input audio in generated talking faces as compared to actual faces. Nonetheless, a common challenge of DiffusedHeads and DiffTalk observed is the closing of the mouth in frames towards the end of speech segments. StableTalk has effectively mitigated this issue, ensuring that lip movements closely mirror those of the original faces throughout the speech.

In summary, the provided quantitative analysis and visual comparisons highlight the advancements made by StableTalk in the field of audio-to-talking face generation. StableTalk excels in creating detailed and realistic facial animations that closely resemble the ground truth.

3.3 Ablation Study

Effect of the Re-attention Mechanism. In this experiment, we conduct a comparative analysis among three models to evaluate the impact of Transformer and Re-attention on the generation of realistic talking faces. The baseline model, DiffTalk, does not utilize a vision Transformer or a Re-attention mechanism. In contrast, the subsequent models differ and introduce new approaches. Method1 integrates the vision Transformer and standard cross-attention mechanism whereas our proposed model, Method2, incorporates cross Re-attention. Based on reported metrics in Table 2, Method2 achieves the best results in terms of FID (11.391), LPIPS (0.0703), SSIM (0.721) and PSNR (25.267). These results confirm the efficacy of vision Transformers and Re-attention in enhancing the perceptual similarity of synthesized talking faces to actual facial expressions, and in improving the overall visual fidelity.

Table 2. A comparative analysis between StableTalk with and without Re-attention mechanism and DiffTalk.

Model	ViT	Re-attention	Results			
			FID↓	SSIM↑	PSNR↑	LPIPS↓
Baseline	×	×	12.459	0.674	23.991	0.094
Method 1	✓	×	11.676	0.688	23.698	0.082
Mathod 2	✓	✓	**11.391**	**0.721**	**25.267**	**0.0703**

Incorporating the Re-attention mechanism enhances the ability of the model to capture complex facial features, expressions, and movements from audio. The visual comparison presented in Fig. 6 demonstrates this improvement. This figure illustrates that frames generated by StableTalk with the Re-attention mechanism exhibit improved lip synchronization and more accurately capture facial expressions, particularly in the eye regions. Additionally, in some generated frames by StableTalk without Re-attention, we notice sudden changes in color contrast of the frames which considerably impact on realism and smooth transition between frames. The employment of the Re-attention mechanism addresses this issue effectively, ensuring a consistent and realistic visual output.

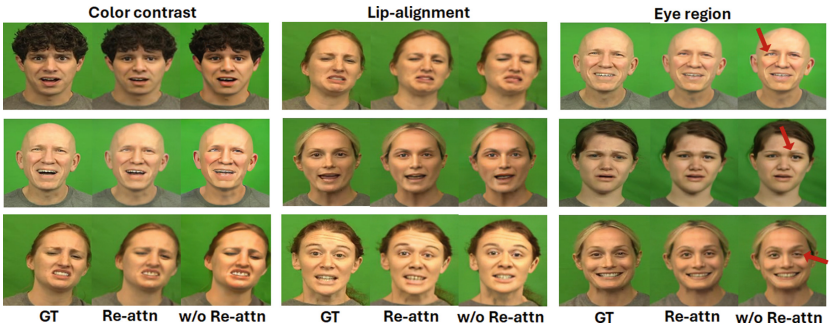


Fig. 6. The impact of Re-attention on StableTalk (zoom in for better comparison).

3.4 Inference Speed

In audio-to-talking face generation, the computational efficiency and complexity of a model are critical for practical applications. We conduct a comparative study focusing on the inference speed and the number of model parameters. The results are reported in Table 3. The proposed model achieves the inference speed of 169 in FPS. This significantly outperforms the state-of-the-art methods DiffTalk and DiffusedHeads, which achieve 18.5 and 0.09 FPS, respectively. Although DiffusedHeads uses timestep-respacing during inference, based on determined results in Table 3, the better acceleration by DDIM can be concluded. Timestep-respacing impacts the scheduling technique and applies no modification on the stochastic and sequential process within standard DDPM. However, DDIM applies modification to the diffusion process itself by employing deterministic and non-Markov process [19,36]. Interestingly, the performance gain of StableTalk did not come at the cost of increased model complexity. The speed-to-parameter ratio for StableTalk provides a balanced profile, highlighting its practicality for deployment in real-world scenarios where both speed and complexity are essential considerations.

Table 3. Comparison with SOTA methods in terms of inference speed and model size.

Model	Speed (FPS)	# Parameters
DiffusedHeads [31]	0.09	215M
DiffTalk [28]	18.5	530M
StableTalk	**169**	**144M**

Integrating vision Transformers with the Re-attention mechanism allows the generative model to dynamically adjust attention weights, focusing better on relevant input features. This approach reduces processing redundancy, leading to efficient computation and faster inference speed. Furthermore, the use of DDIM,

with its non-Markovian process, directly speeds up the speed of StableTalk. The deterministic and non-Markovian approach of the DDIM allows for reduction in the number of required steps to denoise an image. This effectively avoids few intermediate states that DDPM would sequentially traverse. Consequently, this reduction in steps results in quicker image generation, enhancing practicality of the model for real-time applications where rapid inference is essential [14,21, 29,36]. These strategies, when combined with the guidance of facial landmarks and masked frames, not only refine detail generation but also accelerate the inference process. Such guiding features provide essential information on the expected frame structure, effectively narrowing the search space of the generative model. This enables the model to concentrate on refining details, simplifying the generation of facial structures.

4 Conclusion

In this paper, we proposed StableTalk, a latent diffusion-based model for realistic and efficient audio-to-talking face generation. The proposed method demonstrated superior capability in generating realistic facial animations. These facial animations are effectively synchronized with audio inputs by incorporating advanced techniques such as the Re-attention mechanism and adversarial loss. In addition to its superior image generation quality, our model stands out for its computational efficiency, outperforming the existing methods in terms of inference speed. For future research, we aim to enhance the adaptability and efficiency of StableTalk by reducing its reliance on extra conditioning and broadening its application across diverse identities.

Acknowledgment. This work has been supported by EPSRC grants MVSE (EP/V002856/1) and JADE2 (EP/T022205/1).

References

1. Atito, S., Awais, M., Kittler, J.: GMML is all you need. In: International Conference on Image Processing (2022)
2. Bao, J., Chen, D., et al.: CVAE-GAN: fine-grained image generation through asymmetric training. In: International Conference on Computer Vision (2017)
3. Cao, H., et al.: Crema-d: crowd-sourced emotional multimodal actors dataset. IEEE Trans. Affect. Comput. **5**(4), 377–390 (2014)
4. Chang, H.C., Chen, H.J., et al.: Re-attention is all you need: memory-efficient scene text detection via re-attention on uncertain regions. In: International Conference on Intelligent Robots and Systems (2021)
5. Chen, L., Cui, G., Kou, Z., Zheng, H., Xu, C.: What comprises a good talking-head video generation?: A survey and benchmark. ArXiv (2020)
6. Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. In: Conference on Neural Information Processing Systems (2021)
7. Dosovitskiy, A., Beyer, L., Kothers: An image is worth 16x16 words: transformers for image recognition at scale. International Conference on Learning Representations (2021)

8. Goodfellow, I., Pouget-Abadie, J., et al.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
9. Guo, Y., Chen, K., et al.: Ad-nerf: audio driven neural radiance fields for talking head synthesis. In: International Conference on Computer Vision (2021)
10. Heusel, M., Ramsauer, H., et al.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems (2017)
11. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Conference on Neural Information Processing Systems (2020)
12. Khan, S., Naseer, M., et al.: Transformers in vision: a survey. ACM Comput. Surv. **54**, 1–41 (2022)
13. Khormali, A., Yuan, J.S.: DFDT: an end-to-end deepfake detection framework using vision transformer. Appl. Sci. (2022)
14. Kong, Z., Ping, W.: On fast sampling of diffusion probabilistic models (2021)
15. Liang, J., Zeng, H., Zhang, L.: Details or artifacts: a locally discriminative learning approach to realistic image super-resolution. In: Conference on Computer Vision and Pattern Recognition (2022)
16. Liu, H., Zhang, W., et al.: Adaptivemix: improving GAN training via feature space shrinkage. In: Conference on Computer Vision and Pattern Recognition (2023)
17. Liu, H., Zhang, W., et al.: Improving GAN training via feature space shrinkage (2023)
18. Liu, L., Ren, Y., et al.: Pseudo numerical methods for diffusion models on manifolds. In: International Conference on Learning Representations (2022)
19. Lyu, Z., XU, X., et al.: Accelerating diffusion models via early stop of the diffusion process (2022)
20. Nazarieh, F., Feng, Z., et al.: A survey of cross-modal visual content generation. IEEE Trans. Circuits Syst. Video Technol. (2024)
21. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning (2021)
22. Prajwal, K.R., Mukhopadhyay, R., Namboodiri, V.P., Jawahar, C.: A lip sync expert is all you need for speech to lip generation in the wild. In: ACM International Conference on Multimedia (2020)
23. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Conference on Computer Vision and Pattern Recognition (2022)
24. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (2015)
25. Sauer, A., Karras, T., et al.: StyleGAN-T: unlocking the power of GANs for fast large-scale text-to-image synthesis (2023)
26. Schneider, S., Baevski, A., et al.: wav2vec: unsupervised pre-training for speech recognition. In: Interspeech (2019)
27. Sharma, M.: Multi-lingual multi-task speech emotion recognition using wav2vec 2.0. In: International Conference on Acoustics, Speech and Signal Processing (2022)
28. Shen, S., Zhao, W., et al.: Difftalk: crafting diffusion models for generalized audio-driven portraits animation (2023)
29. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021)
30. Song, Y., Zhu, J., Li, D., et al.: Talking face generation by conditional recurrent adversarial network. In: International Joint Conference on Artificial Intelligence (2019)

31. Stypułkowski, M., Vougioukas, K., et al.: Diffused heads: diffusion models beat GANs on talking-face generation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (2024)
32. Suwajanakorn, S., Seitz, S.M., Kemelmacher-Shlizerman, I.: Synthesizing obama: learning lip sync from audio. ACM Trans. Graph. (2017)
33. Wang, Z., Bovik, A., et al.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. (2004)
34. Wu, Y., Lan, L., et al.: Image super-resolution reconstruction based on a generative adversarial network. IEEE Access (2020)
35. Xu, X., Kang, Y., et al.: Explore wav2vec 2.0 for mispronunciation detection. In: Interspeech (2021)
36. Yang, S., Chen, Y., et al.: Denoising diffusion step-aware models. In: International Conference on Learning Representations (2024)
37. Yuanji, W., Jianhua, L., et al.: Image quality evaluation based on image weighted separating block peak signal to noise ratio. In: International Conference on Neural Networks and Signal Processing (2003)
38. Zakharov, E., Shysheya, A., et al.: Few-shot adversarial learning of realistic neural talking head models. In: International Conference on Computer Vision (2019)
39. Zhang, R., Isola, P., et al.: The unreasonable effectiveness of deep features as a perceptual metric. In: Conference on Computer Vision and Pattern Recognition (2018)
40. Zhou, D., Kang, B., Jin, X., et al.: Deepvit: towards deeper vision transformer (2021)
41. Zhou, Y., Han, X., et al.: MakeltTalk. ACM Trans. Graph. (2020)

Can LLMs Perform Structured Graph Reasoning Tasks?

Palaash Agrawal[1,2(✉)], Shavak Vasania[3], and Cheston Tan[1,2]

[1] Center for Frontier AI Research Agency for Science, Technology and Research, Singapore, Singapore
agrawal_palaash@cfar.a-star.edu.sg
[2] Institute of High Performance Computing Agency for Science, Technology and Research, Singapore, Singapore
[3] UWCSEA Dover High School, Singapore, Singapore

Abstract. Pretrained Large Language Models (LLMs) have demonstrated various reasoning capabilities through language-based prompts alone, particularly in unstructured task settings (i.e. tasks purely based on language semantics). However, LLMs often struggle with structured tasks, because of the inherent incompatibility of input representation. Reducing structured tasks to uni-dimensional language semantics often renders the problem trivial. Keeping the trade-off between LLM compatibility and structure complexity in mind, we design various graph reasoning tasks as a proxy to semi-structured tasks in this paper, in order to test LLMs' ability to use representations beyond plain text. In particularly, we design 10 distinct problems of graph traversal, each representing increasing levels of complexity, and benchmark 5 different instruct-finetuned LLMs (GPT-4, GPT-3.5, Claude-2, Llama-2 and Palm-2) on the aforementioned tasks. Further, we analyse the performance of models across various settings such as varying sizes of graphs, as well as different forms of k-shot prompting. We highlight various limitations, biases and properties of LLMs through this benchmarking process, such as an inverse relation to the average degrees of freedom of traversal per node in graphs, the overall negative impact of k-shot prompting on graph reasoning tasks, and a positive response bias which prevents LLMs from identifying the absence of a valid solution. Finally, we introduce a new prompting technique specially designed for graph traversal tasks (***PathCompare***), which demonstrates a notable increase in the performance of LLMs in comparison to standard prompting techniques such as Chain-of-Thought (CoT) (The code for reproducing the results of this paper can be found at https://github.com/PalaashAgrawal/LLMGraphTraversal).

Keywords: Large Language Models · LLM reasoning · graph reasoning · LLM benchmark

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_19.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 287–308, 2025.
https://doi.org/10.1007/978-3-031-78172-8_19

1 Introduction

Large Language Models (LLMs) exhibit interesting reasoning abilities in various types of reasoning tasks [23], including arithmetic [27], logical [26], semantic [14], and symbolic [25]. Building on top of just raw reasoning abilities, various techniques, such as chain-of-thought (CoT) and self-consistency [22], have been proposed to improve the reasoning abilities of LLMs by enhancing the implicit knowledge in the output. However, in relative terms, graph reasoning is not well explored in the domain of LLMs, even though many important subproblems arise in graph reasoning, including subgraph matching [5] and shortest path connectivity [6]. One of the major challenges of grounding graphs through language is rooted in the fact that graphs inherently cover non-linear connectivity between multi-dimensional elements in a condensed format, and hence, reasoning over graphs requires simultaneous tracking of the state of multiple nodes. Broadly in the context of LLMs, graph reasoning is a proxy for understanding relations are established between different entities in a structured setting, and is worth studying, given the impressive emergent abilities observed in recent LLMs.

The major challenge for LLMs in processing graphs directly arises from the uni-dimensionality of input representations, restricting the scope of reasoning that can be achieved using LLMs [1]. In particular, LLMs are trained on semantic text as next-token predictors, and hence cannot process structured representations[1] such as graphs and trees directly, without having to break down the representation into verbally semantic prompts [2]. This process often constrains the selection of tasks, and/or leads to loss of complexity. Hence, efficiently capturing the underlying structure of graphs is particularly challenging for LLMs.

In this paper, we are particularly interested in studying fundamental graph understanding and reasoning abilities of LLMs. While various previous works have studied graph reasoning using LLMs by means of simplified graph structures as enumerated node adjacency lists and connections [2–4], graph tasks such as traversal are inherently reduced to trivial search retrieval problems due to these flattened representations. Hence, **the ability of LLMs to navigate over more complex and structured representations is still not well explored**. We explore this particular question in-depth in this paper, by going one step further than previous works. Specifically, we prompt LLMs to reason over semi-structured graph representations, instead of plainly verbalized unstruc-

[1] In this paper, we refer to structured representations as data organized according to a specific schema or format that clearly defines the type, relationship, and arrangement of data elements. This structured format allows for efficient processing using standardized algorithms and tools. In this context, graphs are categorized as structured representations by virtue of node connections. On the other hand, LLMs are designed to process unstructured text representations with an inherent uni-dimensional semantic relationship. By default, LLMs are not equipped to process structured representations through relationship rules.

tured representations like previous works, while still maintaining the language compatibility of LLMs.[2]

A systematic graph-reasoning benchmarking of pretrained LLMs was performed in [17], where various interesting properties were revealed. This included preliminary reasoning capabilities in LLMs in basic graph traversal problems by parsing adjacency lists as input, and the ineffectual nature of advanced prompting techniques in the context of logical deduction. In the aforementioned work, an improvement in LLM reasoning is observed as a result of custom prompting, majorly along the lines of directing the LLM to elaborate on graph connections to facilitate node traversal. However, representation of graphs through adjacency lists does not evaluate the ability of LLMs to navigate through structured representations. Moreover, the design of graph traversal tasks in [17] is limited in terms of granularity of reasoning evaluation.

In order to assess the graph reasoning abilities of LLMs in a more comprehensive way, we carry out graph evaluation through a series of increasingly complex graph problems and settings. Particularly, we construct 10 distinct graph problems requiring multi-hop reasoning and tracking. This includes tree-based graph traversals, grid-based graph-traversals as well as certain special classes of problems. We evaluate five different LLMs (GPT-3.5, GPT-4, Claude-2, Llama-2 and Palm-2). While the OpenAI family of models is thoroughly investigated in various papers, this paper studies the nature of various other contenders. For example, in this paper, we demonstrate that Anthropic's Claude-2 is a good logical reasoning model, only sub-standard to GPT-4 among the various models publically available for access. We also choose to adopt certain design choices that highlight interesting limitations and biases in the training of various models. This includes methods like jumbling the sequence of nodes, as well as testing if the models can identify the absence of a valid solution, or in other words, testing if the models are biased to responding with a non-negative response.

By carefully examining the logical flow of reasoning adopted by LLMs in order to come to a response, we observe certain limitations that hinder LLMs specifically in the case of graph traversal problems. Using these insights, we propose a novel prompting technique, which we refer to as ***PathCompare***, which simplifies the reasoning problem by allowing models to compare different possible solutions.

In essence, the main contributions of this paper are

1. We benchmark five different LLM models on graph reasoning tasks. The graph reasoning tasks are carefully designed in the form of 10 graph traversal problems in increasing orders of complexity. The design of these problems not only

[2] In this paper's context, semi-structured representations involve breaking down a structured representation into uni-directional text, while still maintaining some form of element relationship. In our case, adjacency matrix representations involve a tabular relation set, which requires tracking both the column and row origin of a node. This structure introduces complexity to the reasoning task. Particularly, we are interested in exploring the ability in LLMs to handle such representational complexity.

assesses the complexity that LLMs can handle, but also highlights interesting properties, including biases introduced due to prompting.
2. To study graph reasoning in depth, we also introduce many variations on top of the graph traversal tasks, including node order (graph size) variation, introduction of weighted and directed connections, and jumbling of node labels. This allows us to guarantee a wide collection of tasks not previously seen by LLMs during pretraining.
3. We finally propose a new prompting technique (**PathCompare**) which improves graph traversal performance across different LLMs. We demonstrate that this prompting technique outperforms standard prompting as well as CoT prompting in the majority (i.e. >50%) of the designed tasks).

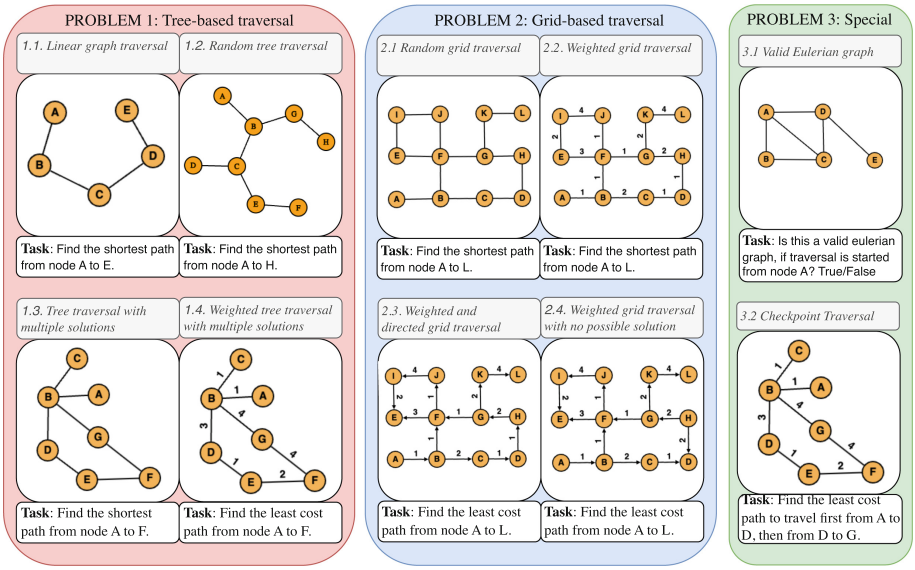


Fig. 1. Visualization of all problem categories considered for evaluating LLMs. For each problem, we create 3 variations – *O(10)*, *O(20)* and *O(20) jumbled*, representing increasing levels of difficulty.

2 Defining Graph Reasoning Tasks

A typical graph traversal problem involves navigating a graph $G = \{V, E\}$ between a specified pair of nodes $u, v \in V$ through a sequence of edges $e = (e_1, e_2 ....e_n) \in E$. Traversal of a graph G can be formulated in various ways, such as determining connectivity between two sub-graphs, shortest/least-cost path optimization and maximum flow optimization, and so on. However, we identify

shortest/least-cost path optimization as one major class of problems involving multi-hop non-trivial reasoning. Analyzing the properties of large language models on such path optimization problems can thus reveal many interesting properties of the models, apart from multi-hop reasoning ability, such as multi-variable state tracking ability, recursive thinking, as well as the ability to reject non-optimal paths. We start by defining the types of graphs that are considered for evaluating traversal properties in LLMs.

2.1 Defining Graph Levels and Complexity

As a starting point, we categorize various types of graphs into two major categories, as described below.

Problem 1: Tree-based graphs these graphs are defined as connected undirected graphs, where any two nodes u, v are connected by atmost one edge $e\epsilon\{0, R^+\}$, (where R^+ represents positive real number values). In the above representation $e = 0$ represents no connection, $e = 1$ represents a simple unweighted connection between nodes u, v, and all other positive values $e \ \epsilon \ (R^+|e \neq 1)$ represents a weighted connection between nodes u, v. By definitions, no cycles exist in a tree-based graph. However, for the sake of logical grouping of tasks in a generalizable manner, we define *tree-based* graphs as tree-like heirarchical structures with possible exceptions (with respect to cyclic connections). The emphasis is on navigating through these arbitrary node connections to find the shortest or least costly route. Based on this, we formulate the following problems

- **Problem 1.1: Linear graph traversal** a linear graph is defined as a graph $G(n)$ with n nodes such that its nodes $V = (v_1, v_2, ...v_n)$ are connected pairwise sequentially $(e_{i,i+1} = 1)$, where $e_{i,j}$ represents an edge between nodes v_i and v_j. The task involves traversing from node v_1 to v_n. This is the most trivial form of graph traversal, since there exists only one possible traversal path originating from node v_1.
- **Problem 1.2: Random tree traversal**: given a unweighted and undirected tree-based graph $G(n)$ of order n, the task involves traversal from node v_1 to v_n. This involves traversing through and rejecting many dead-end paths. We ensure that there only exists one possible path from node v_1 to v_n.
- **Problem 1.3: Tree traversal with multiple possible solutions**: given a unweighted and undirected tree-based graph $G(n)$ of order n, the task involves finding the shortest path from node v_1 to v_n, among many possible paths. There exists only one shortest possible path in this tree.
- **Problem 1.4: Weighted tree traversal with multiple possible solutions** this setting is similar to problem *1.3*, except the edges are weighted. Hence, the task is modified into a least-cost traversal problem. There exists only one possible least cost path.

Grid-based graphs these graphs are defined as two-dimensional connected graphs $G(M, N)$ of size $M \times N$, where node $u = G(i, j)\ \forall (i, j) | (1 \leq i \leq M), (1 \leq j \leq N)$ is connected to another node $v = G(p, q)$ by atmost one edge $e \epsilon \{0, R^+\}$, (where R^+ represents positive real number values, and the definition of edge values are similar to tree-based graphs). A grid-graph is a collection of nodes arranged in a regular grid pattern, with nodes connected to their immediate neighbors in a row-column configuration. By virtue of being two-dimensional, they are more densely connected than tree-based graphs, and inherently have cyclic connections (loops) between nodes. Thus, these form a more-challenging class of graph-traversal problems due to the involvement of selection among a higher number of possible solutions. Based on this, we formulate the following problems.

- **Problem 2.1: Random grid traversal** given a unweighted and undirected graph $G(M, N)$ with dimensions (M, N), the problem involves finding the shortest path of traversal from node $G(1, 1)$ to node $G(m, n)$. There exists only one shortest possible path per grid.
- **Problem 2.2: Weighted grid traversal** This problem is similar in setting to problem *2.1* except the edges are weighted. Hence the task evolves into a least-cost path traversal from node $G(1, 1)$ to $G(m, n)$. There exists only one least-cost path per grid.
- **Problem 2.3: Directed and weighted grid traversal** This problem is similar in setting to problem *2.1* except all edges are weighted as well as directed. Hence the problem evolves into finding a valid least-cost traversal from node $G(1, 1)$ to $G(m, n)$.
- **Problem 2.4: Directed grid traversal with no possible solution** This problem involves a directed grid $G(M, N)$, such that no valid path exists because of randomized directions. The goal of the problem is to evaluate whether LLMs are capable of evaluating directional conflicts, and come to the conclusion that no path is valid.

By incrementally adding constraints to both the problem categories above, we create an analysis framework to determine the graph reasoning abilities of LLMs in increasing levels of complexity. Between the two problem categories, problem 2 poses a more challenging problem than problem 1, due to a larger number of average degrees of freedom of traversal per node.

Problem 3: Special Problems. Apart from the two aforementioned categories of problems, we define two special tree-based traversal problems.

- **Euler walk**: Given a valid Euler graph G, an Euler path is defined as a path which covers all edges E of the graph exactly once. The Eulerian path traversal is a more viable alternative to the *Hamiltonian path* traversal, which involves travelling all nodes U of the graph exactly once, since the former can be evaluated computationally by a finite set of conditions, unlike the latter, which is an NP-complete problem and cannot be evaluated using a polynomial time algorithm. Hence, we formulate the problem as follows.

  - **Problem 3.1: Valid Euler graph** Given a tree-based graph $G(n)$ with n nodes, the task involves identifying whether or not a valid Euler path is possible, given a starting node v_i.
- **Checkpoint traversal** This problem involves traversal of a tree-based graph G between two nodes u and v, such that a specific node w must be part of the traversal path. Mathematically, this is equivalent to breaking down the traversal into two distinct problems – traversal from u and w, and traversal from node w and v. The goal of this cascaded traversal problem is to evaluate whether a single LLM response can simultaneously solve multiple problems. We hence formulate the following problem.
  - **Problem 3.2: Cascaded graph traversal** Given a weighted tree graph $G(n)$ with n nodes, find the least cost path to traverse from node v_1 to a randomized node v_i, and then from v_i to v_n.

2.2 Graph Generation, Prompting and Evaluation

Automated Graph Generation. Graphs and their corresponding solutions are automatically generated as adjacency matrices with alphabetically labeled nodes. We deliberately avoid adjacency lists (a more standard practice of computational representation), to avoid triviality, since a direct listing of individual node connections simplifies the complexity for the LLM, thereby hindering the ability to assess its raw reasoning capabilities. We ensure that there exists only one unique solution for all problems, so that evaluation can be unambiguous. For problem *3.1* (Euler walk), we convert it into a True/False problem due to the lack of a guaranteed single Eulerian path, ensuring a balanced distribution of valid and invalid cases. All remaining problems require LLMs to provide a sequence of nodes in response. Thus, while lower LLM accuracies are expected, a more comprehensive evaluation can be performed. Across all problems, we generate three variations (or orders) of problems, depicting increasing levels of complexities.

1. *O(10)*: This refers to graphs that have a total number of nodes varying around $n = 10$, or between 5 and 15 (inclusive) to be exact. Hence the problem is formulated as traversal between A and J when $n = 10$, and so on, in the case of problem categories 1 and 2.
2. *O(20)*: This refers to graphs which have a total number of nodes varying around $n = 20$, or between 16 and 26 (inclusive). Hence, when $n = 20$, the problem is formulated as traversal between node A and T, and so on, in the case of problem categories 1 and 2.
3. *O(20) jumbled*: In contrast to the previous cases, where node labels follow a near-ordered sequence in depth-first traversal, the labels of the nodes in *O(20)* are randomized to increase complexity. Although the adjacency matrix remains alphabetically ordered, the solution sequence is entirely shuffled. This approach allows us to assess whether LLMs depend on training biases, such as node order, when generating solutions.

Further, we ignore evaluating problem 3.1 (valid euler graph) for O(20) and O(20) jumbled graphs. This is because generating larger graphs with guaranteed eulerian conditions are computationally highly expensive.

Prompting Techniques. In order to maintain structure of graphs without explicit flattening and enumeration of nodes/connections, we represent graph structures as **adjacency matrices**, rather than simpler adjacency lists. This kind of structure is not only more compact, but tests the ability of LLMs to perform multi-hop reasoning, and tracking value states more rigorously. A detailed description of our base prompting method is described in the Appendix.

For prompting, our goal was to ensure that all models are given the same prompts for fair evaluation. For all models, we test graph reasoning abilities across 3 k-shot settings – *zero-shot* setting ($k = 0$), *one-shot* setting ($k = 1$) and *three-shot* setting ($k = 3$) [28]. To evaluate and compare the preliminary graph reasoning abilities of LLMs, we avoid any specialized prompting techniques (such as self-consistency [27] and other related techniques). However, later in the paper, chain-of-thought prompting [22] is introduced, majorly to compare the impact specialized prompting on graph reasoning.

Partial Credit Evaluation. Apart from an automated evaluation, we go one step further and manually evaluate problem categories 1 and 2 (except 2.4, where partial credit does not apply) for partial credit. We define partial credit as the fraction of nodes that the evaluator got correct out of all the nodes in the ground truth solution before predicting to a wrong node. This is only applicable in cases where the model does not get a full score of 1.0 from the primary evaluator. This metric allows us to evaluate the reasoning ability and problem solving technique of LLMs with greater granularity.

2.3 Selection of LLMs

In order to include a broad range of models, we select five models from four families of models (in terms of architecture and training data/method). This includes – OpenAI's **GPT-3.5** and **GPT-4** [21], Anthropic's **Claude-2**[3] [20], Google's **Palm-2** [19], and Meta's **Llama-2** [18] with 13B parameters. Since LLama-2 is a raw foundational model, we use an instruction-fine tuned version (Hermes) released by *Nous Research*.

3 Results and Analysis

Using the prompting and evaluation technique defined above, we evaluate models over the defined problems. We evaluate the results of graph reasoning for all problems by calculating averaged and normalized binary accuracies across 10 examples for each setting, and go from 0 to 1 in increments of 0.1. A model response is only marked correct if it predicted the shortest/least-cost traversal path correctly and completely. Along with binary accuracies, partial accuracies are also calculated. While we use binary accuracy to filter out models with weak reasoning abilities, we use partial accuracy for more granular comparison. These

[3] It should be noted that Claude-3 was not released at the time of conducting these experiments.

results can be visualized in Fig. 2. A detailed tabulation of these results are included in the Appendix Tables 1, 2 and 3.

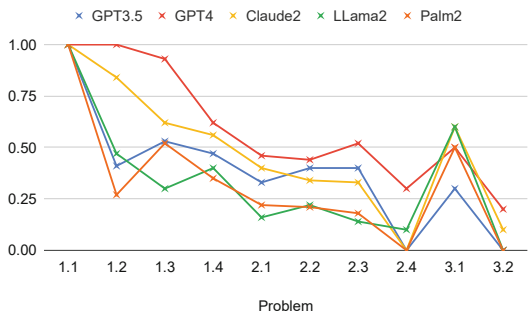


Fig. 2. Baseline comparison of all model families on O(10) graph problems in 0-shot settings using partial accuracy. Partial accuracy gives a more granular insight into the performance of models, versus binary accuracy.

3.1 Preliminary Technical Limitations

Before analysing the performance of various LLMs on different LLMs, we observe some overall characteristics of models. Particularly, we observe issues with Llama-2 and Palm-2. In the case of Llama-2, 3-shot setting for O(20) graphs could not be evaluated, because of the limited context-window length of the model (4,096 tokens). While this issue was also seen with the default GPT-3.5 model, a GPT-3.5 variant with a larger context window (16,383 tokens) was able to solve the issue. In the case of Palm-2, we observe that beyond a certain length of input (in the case of 1-shot and 3-shot prompts), the model tends to not respond with any solution, but with a generic statement related to the model's inability to solve the problem.

3.2 Overall Observations from Problem 1 and Problem 2

Through a first, glance, we can easily point out that all models generally perform better on tree-based graphs (problem 1) than grid-based graphs (problem 2). Further, we notice a general drop in performance in random tree traversal (problem 1.2) across O(10) and O(20) graphs. This clearly indicates that a greater number of average degrees of freedom for traversal per node has an inverse correlation with reasoning capability.

Model Specific Observations. Although all models perform decently on trivial graph traversal tasks (e.g. problem 1.1), differentiation among models is quickly observed as more constraints are added. Referring to Fig. 2, a general aggregate trend is observed where some reasoning abilities are observed in the case

Table 1. Evaluation of models on problem category 1: tree-based graph traversals. The values are represented as A/B, where A depicts the average normalized binary accuracy (ranging from 0 to 1) for 10 examples per setting, and B depicts the average normalized value for partial credit for 10 examples per setting. * = a larger context variant of the corresponding LLM was used. △ = the prompt was too large to fit the context window of the model, hence the setting was omitted. ** = the model did not return any solution and refused to answer the question.

Problem	O(10)			O(20)			O(20) jumbled		
	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot
GPT-3.5									
1.1	1.0	0.6/0.80	1.0	1.0	1.0	1.0*	0.0/0.09	0.1/0.08	0.0/0.07*
1.2	0.2/0.41	0.4/0.59	0.6/0.62	0.0/0.39	0.1/0.38	0.2/0.31*	0.0/0.22	0.2/0.29	0.0/0.29*
1.3	0.3/0.53	0.0/0.24	0.4/0.48	0.2/0.36	0.1/0.17	0.0/0.26*	0.0/0.28	0.0/0.21	0.0/0.19*
1.4	0.3/0.47	0.2/0.45	0.0/0.48	0.1/0.37	0.0/0.23	0.0/0.33*	0.1/0.24	0.0/0.13	0.0/29*
GPT-4									
1.1	1.0	1.0	1.0	1.0	1.0	1.0	0.7/0.84	0.9/0.98	0.7/0.87
1.2	1.0	1.0	1.0	0.5/0.67	0.8/0.85	0.9/0.93	0.1/0.31	0.2/0.47	0.4/0.55
1.3	0.8/0.93	0.6/0.69	0.9/0.91	0.4/0.64	0.5/0.70	0.6/0.71	0.2/0.45	0.2/0.40	0.3/0.51
1.4	0.4/0.62	0.0/0.31	0.4/0.59	0.2/0.40	0.1/0.35	0.0/0.27	0.1/0.26	0.2/0.39	0.3/0.51
Claude-2									
1.1	1.0	0.0/0.60	1.0	0.2/0.30	0.2/0.43	1.0	0.0/0.07	0.0/0.06	0.0/0.12
1.2	0.7/0.84	0.0/0.30	0.3/0.50	0.3/0.53	0.1/0.29	0.2/0.31	0.1/0.31	0.0/0.18	0.0/0.22
1.3	0.4/0.62	0.5/0.70	0.2/0.56	0.0/0.37	0.2/0.39	0.1/0.35	0.3/0.36	0.0/0.26	0.0/0.14
1.4	0.4/0.56	0.2/0.44	0.1/0.34	0.2/0.41	0.0/0.25	0.1/0.34	0.1/0.30	0.0/0.19	0.0/0.17
Llama-2									
1.1	1.0	0.0/0.10	1.0	0.0/0.05	0.0/1.0	△	0.0/0.06	0.0/0.05	△
1.2	0.2/0.47	0.1/0.31	0.0/0.18	0.0/0.15	0.0/0.14	△	0.0/0.11	0.0/0.16	△
1.3	0.0/0.30	0.1/0.35	0.0/0.25	0.0/0.17	0.0/0.13	△	0.0/0.14	0.0/0.16	△
1.4	0.2/0.40	0.1/0.37	0.0/0.32	0.0/0.16	0.0/0.25	△	0.0/0.15	0.0/0.17	△
Palm-2									
1.1	1.0	1.0	1.0	1.0	**	**	0.0/0.05	0.0/0.05	0.0/0.06
1.2	0.2/0.27	0.0/0.32	0.1/0.44	0.0/0.27	**	**	0.0/0.19	0.0/0.17	**
1.3	0.1/0.52	0.2/0.53	0.2/0.50	**	**	**	**	**	**
1.4	0.0/0.35	0.1/0.38	0.0/0.34	0.0/0.10	**	**	0.0/0.08	**	**

of GPT4, Claude-2, and GPT-3.5 (in this order of decreasing relative reasoning abilities). On the other hand, Llama-2 and Palm2 perform under 30% on the majority ($\geq$ 50% of the settings) of the tasks, depicting poor reasoning abilities. In more complex settings (such as O(20) graphs and O(20) jumbled graphs, not presented in this paper for simplicity and readability, but present in the code repository), an almost random-like performance is seen in these models. An anomaly is observed in problem 3.1 where a spike is observed in the performance of the aforementioned 2 models. However, since this particular problem is a True/False problem with a random baseline accuracy of 50%, the spike is rendered meaningless. Keeping this in mind, in the remainder of the paper, we primarily focus on GPT4, Claude-2, and GPT-3.5, since these contenders depict

Table 2. Evaluation of models on problem category 2: grid-based graph traversals. The values are represented as A/B, where A depicts the average normalized binary accuracy (ranging from 0 to 1) for 10 examples per setting, and B depicts the average normalized value for partial credit for 10 examples per setting. The notations are similar to Table 1.

Problem	O(10)			O(20)			O(20) jumbled		
	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot
GPT-3.5									
2.1	0.1/0.33	0.2/0.46	0.3/0.41	0.1/0.21	0.1/0.28	0.1/0.28*	0.1/0.14	0.0/0.13	0.0/0.12*
2.2	0.0/0.40	0.0/0.19	0.0/0.37	0.0/0.30	0.0/0.12	0.1/0.32*	0.0/0.13	0.1/0.13	0.0/0.13*
2.3	0.1/0.40	0.10/0.36	0.0/0.37	0.0/0.14	0.0/0.17	0.0/0.22*	0.0/0.13	0.0/0.09	0.0/0.11*
2.4	0.0	0.0	0.0	0.0	0.0	0.3*	0.0	0.0	0.0*
GPT-4									
2.1	0.3/0.46	0.3/0.48	0.4/0.51	0.1/0.36	0.1/0.29	0.0/0.21	0.0/0.14	0.0/0.23	0.0/0.16
2.2	0.0/0.44	0.4/0.56	0.2/0.37	0.1/0.42	0.1/0.31	0.3/0.54	0.0/0.14	0.0/0.26	0.0/0.23
2.3	0.2/0.52	0.0/0.36	0.2/0.40	0.1/0.34	0.2/0.49	0.1/0.28	0.0/0.16	0.0/0.16	0.0/0.17
2.4	0.3	0.3	0.4	0.4	0.4	0.4	0.4	0.9	0.6
Claude-2									
2.1	0.0/0.40	0.3/0.44	0.0/0.36	0.0/0.29	0.1/0.20	0.0/0.22	0.0/0.16	0.0/0.13	0.1/0.16
2.2	0.1/0.34	0.2/0.41	0.1/0.31	0.0/0.19	0.3/0.23	0.0/0.21	0.0/0.17	0.1/0.12	0.0/0.13
2.3	0.1/0.33	0.1/0.29	0.0/0.24	0.0/0.17	0.0/0.18	0.0/0.26	0.0/0.12	0.0/0.13	0.0/0.12
2.4	0.0	0.1	0.3	0.0	0.1	0.3	0.0	0.3	0.4
Llama-2									
2.1	0.0/0.16	0.2/0.44	0.1.0.24	0.0/0.18	0.1/0.12	△	0.0/0.05	0.0/0.05	△
2.2	0.0/0.22	0.0/0.40	0.1/0.52	0.0/0.18	0.0/0.36	△	0.0/0.04	0.0/0.06	△
2.3	0.0/0.14	0.0/0.22	0.0/0.12	0.0/0.08	0.0/0.12	△	0.0/0.014	0.0/0.06	△
2.4	0.1	0.8	0.0	0.0	0.0	△	0.0	0.0	△
Palm-2									
2.1	0.0/0.22	0.5/0.64	0.2/0.42	0.0/0.06	**	**	0.0/0.04	**	**
2.2	0.0/0.21	0.0/0.24	0.0/0.30	0.0/0.04	**	**	**	**	**
2.3	0.0/0.18	0.0/0.30	0.0/0.22	**	**	**	0.0/0.08	**	**
2.4	0.0	0.0	0.0	**	**	**	**	**	**

fairly reasonable graph reasoning abilities. However, all the results are openly available for analysis in our code repository.

3.3 Weighted Vs Unweighted Graph Traversal

Adding weights to graph edges in least-cost traversal problems introduces significant complexity, requiring models to track cumulative weights. Unsurprisingly, we observe a general trend of dropping performance as soon as weights are added to trees (problems 1.3 vs 1.4) and well as grids (problems 2.1 vs 2.2). GPT-4 and Claude-2 demonstrate some ability to solve weighted traversal, demonstrating their innate ability to track numerical variable states in parallel to graph paths. On the other hand, chain-of-thought prompting [22] in GPT-3.5 demonstrates that the models tend to neglect weights and solve problems using an unweighted

Table 3. Evaluation of models on problem category 3: special problems. The values depict the average normalized value (ranging from 0 to 1) for 10 examples per setting. **-** = the setting is not applicable for a particular level. △ = the prompt was too large to fit the context window of the model, hence the setting was omitted. ****** = the model did not return any solution and refused to answer the question. Note that for problem 3.1, the random baseline is 0.5, since the problem solution is binary (True/False).

Problem	O(10)			O(20)			O(20) jumbled		
	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot	0-shot	1-shot	3-shot
GPT-3.5									
3.1	0.3	0.1	0.5	-	-	-	-	-	-
3.2	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0
GPT-4									
3.1	0.5	0.5	0.2	-	-	-	-	-	-
3.2	0.2	0.3	0.2	0.0	0.0	0.1	0.0	0.1	0.0
Claude-2									
3.1	0.6	0.4	0.7	-	-	-	-	-	-
3.2	0.1	0.1	0.2	0.1	0.0	0.0	0.0	0.0	0.0
Llama-2									
3.1	0.6	0.8	0.7	-	-	-	-	-	-
3.2	0.0	0.0	0.0	0.0	0.0	△	0.0	0.0	△
Palm-2									
3.1	0.5	0.1	0.2	-	-	-	-	-	-
3.2	0.0	0.0	0.0	0.0	**	**	0.0	**	**

approach. The remainder of the models such as Llama-2 and PaLM 2 tend to respond with a description of the algorithm underlying the solution, rather than the solution itself. This is especially observed in 0-shot settings.

3.4 Effect of K-Shot Prompting

In Fig. 3, we compare the effect of k-shot prompting across 8 tasks, and observe that in the majority of the tasks, k-shot prompting has either no statistically significant effect, or a negative effect on reasoning accuracy. In general, while few-shot prompting is helpful in response format shaping, it is of no particular aid for analytical tasks such as graph reasoning. Carefully examining individual responses, we observe models (even the more powerful ones like GPT4) confusion between different examples, and responding to k-shot examples rather than the relevant graph traversal question.

3.5 Effect of Increased Graph Order and Jumbling Node Order

In Fig. 4, We analyze the impact of different graph orders and complexities, specifically comparing O(10), O(20), and O(20) jumbled graph settings. We consistently observe across all models that a performance drop is observed when graph order is increased. An interesting insight enlightened when manually analyzing individual solutions is that poor performance in O(20) graphs in comparison to O(10) graphs is not solely due to increased computational demands

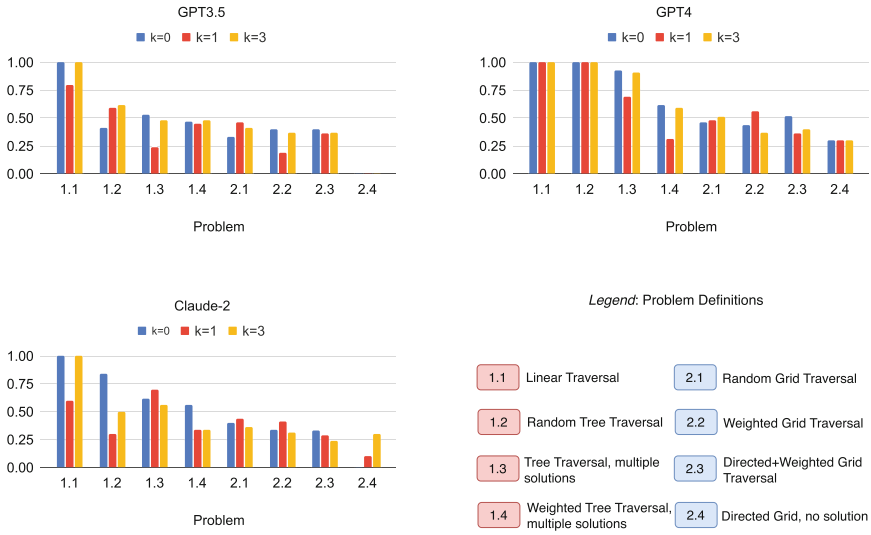


Fig. 3. Comparison of the effect of k-shot prompting on various LLMs. We observe that in more than half of all tasks, few-shot prompting leads to either a drop or an insignificant ($\leq 5\%$) improvement in accuracy in comparison to 0-shot prompting (depicted in blue). This is observed in 6/8 tasks for GPT3.5, 6/8 tasks for GPT4 and 5/8 tasks for Claude-2. (Color figure online)

but also because models often omit higher node labels (e.g., J, K, L, M and so on) when labels are alphabetically ordered, indicating training on lower-order graphs.

This behavior is amplified in the case of jumbled node sequences, where even trivial linear traversal becomes challenging, highlighting a strong model bias towards ordered node sequences, even in few-shot settings. GPT-4, while occasionally applying valid logical principles, still struggles across grid-based traversals.

3.6 Positive Response Bias Within LLMs

Through a special problem (problem 2.4), we observe that all models also consistently fail to recognize when no solution is possible. An interesting observation is that models fail to recognize when no valid solution is possible even in few-shot prompt settings, considering it is expected that this method should induce a formatting bias in the response. This indicates a clear training regime flaw among models to prefer avoiding empty responses at the cost of predicting wrong solutions. However, given that GPT-4 and Claude-2 do possess some valid logical reasoning and state tracking abilities, we observe some ability to predict the inexistence of any valid solution in these models. Surprisingly, in the case of few-shot prompt (where we prompt models with examples of **valid** graphs with

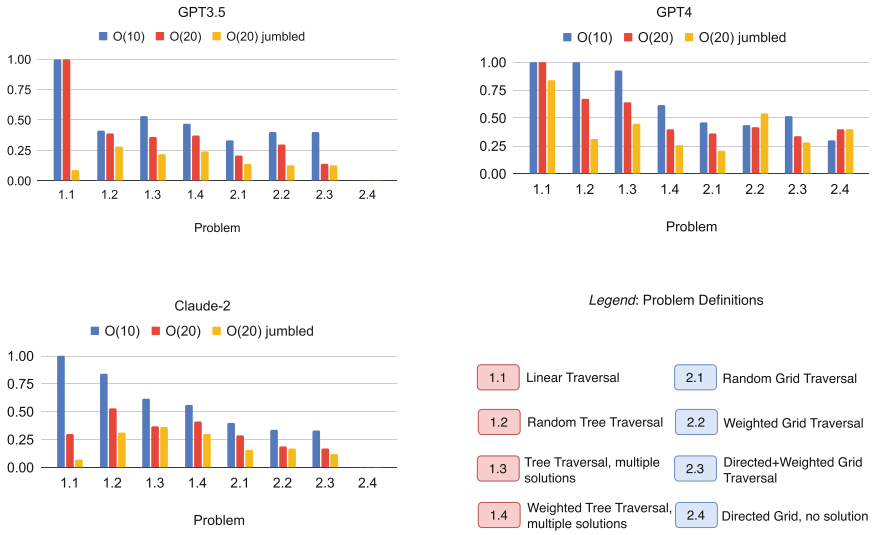


Fig. 4. Comparison of graph order performance in various LLMs. LLM performance accuracy consistently drops as the order of the graph is increased. Also, while keeping the order of graph magnitude at 20, a drop is observed in performance when the nodes are jumbled, depicting a bias in models to expect node order to be alphabetically arranged.

solutions, the aforementioned models tend to identify the absence of a valid path with better accuracy (see Fig. 3).

3.7 Performance of LLMs over Special Problems

As mentioned earlier, in this paper, the eulerian problem (problem 3.1) is a binary problem, and thus a random classifier is expected to perform with an accuracy of 0.5. We observe slightly better performance than random only in the case of Claude-2 and Llama-2. **However, in all other models, the performance over identification of a valid eulerian graph is equal to or less than random.** Analysis of the responses manually highlights that models tend to give a positive response.

4 PathCompare Prompting

As clearly seen in the previous sections, LLMs struggle with graph reasoning. However, this is expected, since Large Language Models are, in essence, token predictors, and do not inherently possess the ability to go back to the prompt recursively to self-correct. By prompting LLMs to directly generate the optimal path to a graph traversal problem, we inherently prevent the model from a systematic analysis of the graph traversal problems.

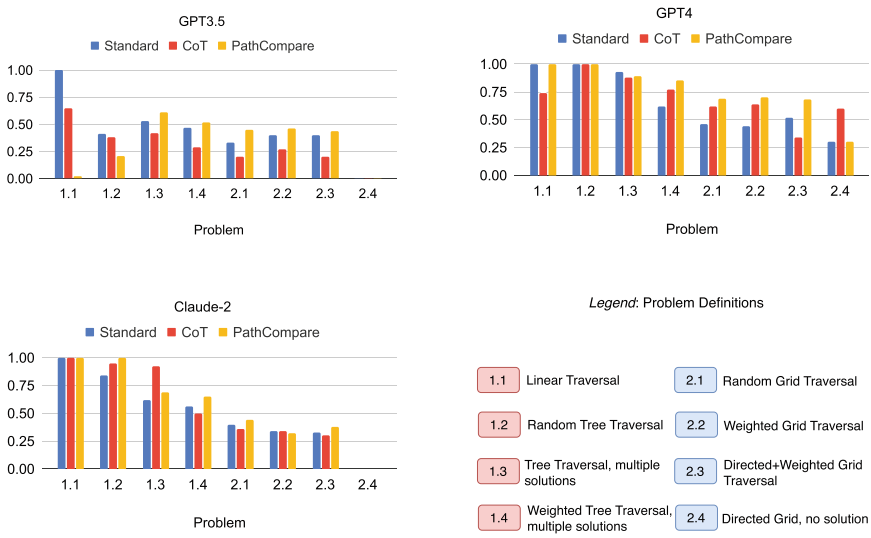


Fig. 5. Comparison of different prompting techniques. Our proposed prompting technique (*PathCompare*) demonstrates an improvement of accuracy in the majority of tasks across all models, i.e. 5/8 tasks for GPT3.5, 6/8 tasks for GPT4 and 5/8 tasks for Claude-2. However, one limitation of PathCompare is that it enhances positive response bias, as observed in the case of problem 2.4.

A more advanced prompting technique such as Chain-of-Thought reasoning [22] has more or less no effect on the performance of LLMs, specifically on graph reasoning. We observe, that even without a prompt augmentation that compels the model to list down its reasoning, LLMs come to a conclusion through an algorithmic walk (such as Dijkstra's algorithm, or breadth-first-search). Essentially, it is not the lack of the flow of reasoning that prevents LLMs from coming to a consistent and accurate traversal solution, but the lack of ability to explore and compare with other valid solutions.

Keeping these requirements in mind, we propose a novel prompting technique specifically for graph reasoning (traversal) tasks, which we refer to as the ***Path-Compare*** technique. By simply adding the line *"Let's list down all the possible paths from node {X} to node {Y}, and compare the cost to get the answer."*, we observe a significant improvement in the graph reasoning performance of all LLMs (see Fig. 5 for visualization and Appendix Table 5 for detailed analysis), in comparison to standard prompting and other forms of prompting such as CoT (see Appendix Table 4). Through this prompt, we shape an LLM response to first list down multiple paths between two specified nodes. Once multiple paths are enumerated, the task boils down to a retrieval-cum-comparison task, which increases the probability of a more accurate response.

Table 4. Evaluation of models on O(10) graphs with **Chain-of-Thought Reasoning**. The values are represented as A/B, where A depicts the average normalized binary accuracy (ranging from 0 to 1) for 10 examples per setting, and B depicts the average normalized value for partial credit for 10 examples per setting. * = a larger context variant of the corresponding LLM was used. △ = the prompt was too large to fit the context window of the model, hence the setting was ommitted. ** = the model did not return any solution and refused to answer the question.

Prob	0-shot	1-shot	3-shot	Prob	0-shot	1-shot	3-shot	Prob	0-shot	1-shot	3-shot
GPT-3.5											
1.1	0.6/0.65	1.0	0.9/0.92	**2.1**	0.0/0.20	0.2/0.50	0.0/0.32*	**3.1**	0.3	0.1	0.5
1.2	0.2/0.38	0.3/0.55	0.5/0.72	**2.2**	0.1/0.27	0.0/0.37	0.1/0.2*	**3.2**	0.0	0.0	0.
1.3	0.2/0.42	0.12/0.38	0.0/0.28	**2.3**	0.0/0.2	0.1/0.29	0.0/0.18*				
1.4	0.0/0.29	0.0/0.28	0.0/0.25	**2.4**	0.0/0.0	0.0/0.0	0.1/0.1*				
GPT-4											
1.1	0.7/0.74	1.0	1.0	**2.1**	0.4/0.62	0.3/0.42	0.0/0.26	**3.1**	0.5	0.5	0.2
1.2	1.0	0.9/0.96	0.8/0.88	**2.2**	0.2/0.64	0.3/0.63	0.4/0.51	**3.2**	0.2	0.3	0.2
1.3	0.8/0.88	0.9/0.91	0.9/0.94	**2.3**	0.0/0.34	0.1/0.45	0.6/0.76				
1.4	0.6/0.77	0.3/0.49	0.6/0.74	**2.4**	0.6/0.6	0.8/0.8	0.7/0.7				
Claude-2											
1.1	1.0	1.0	1.0	**2.1**	0.0/0.36	0.2/0.49	0.0/0.29	**3.1**	0.6	0.4	0.7
1.2	0.9/0.95	0.3/0.48	0.30/47	**2.2**	0.0/0.34	0.0/0.27	0.1/0.27	**3.2**	0.1	0.1	0.2
1.3	0.9/0.92	0.5/0.75	0.4/0.44	**2.3**	0.0/0.3	0.1/0.29	0.2/0.39				
1.4	0.4/0.50	0.0/0.23	0.0/0.28	**2.4**	0.0/0.0	0.0/0.0	0.1/0.1				

Table 5. Evaluation of models on O(10) graphs using the **PathCompare** method. The values are represented as A/B, where A depicts the average normalized binary accuracy (ranging from 0 to 1) for 10 examples per setting, and B depicts the average normalized value for partial credit for 10 examples per setting. * = a larger context variant of the corresponding LLM was used. △ = the prompt was too large to fit the context window of the model, hence the setting was omitted. ** = The model did not return any solution and refused to answer the question.

Prob	0-shot	1-shot	3-shot	Prob	0-shot	1-shot	3-shot	Prob	0-shot	1-shot	3-shot
GPT-3.5											
1.1	0.0/0.02	0.0/0.02	0.0/0.08	**2.1**	0.0/0.45	0.5/0.71	0.3/0.57*	**3.1**	-	-	-
1.2	0.2/0.51	0.4/0.62	0.5/0.80	**2.2**	0.0/0.46	0.2/0.42	0.0/0.31*	**3.2**	0.0	0.2	0.3
1.3	0.4/0.61	0.3/0.49	0.3/0.66	**2.3**	0.0/0.44	0.2/0.46	0.3/0.54				
1.4	0.4/0.52	0.3/0.6	0.2/0.48	**2.4**	0.0/0.0	0.0/0.0	0.0/0.0				
GPT-4											
1.1	1.0	1.0	1.0	**2.1**	0.4/0.69	0.4/0.52	0.2/0.4	**3.1**	-	-	-
1.2	1.0	1.0	1.0	**2.2**	0.6/0.7	0.2/0.44	0.1/0.46	**3.2**	0.2	0.2	0.3
1.3	1.0	1.0	1.0	**2.3**	0.5/0.68	0.2/0.54	0.2/0.59				
1.4	0.8/0.89	0.8/0.81	0.4/0.52	**2.4**	0.3/0.3	0.3/0.3	0.4/0.4				
Claude-2											
1.1	1.0	1.0	1.0	**2.1**	0.1/0.44	0.1/0.52	0.0/0.24	**3.1**	-	-	-
1.2	1.0	0.1/0.32	0.4/0.57	**2.2**	0.1/0.32	0.1/0.28	0.1/0.34	**3.2**	0.2	0.0	0.1
1.3	0.6/0.69	0.3/0.42	0.4/0.57	**2.3**	0.2/0.38	0.1/0.42	0.3/38				
1.4	0.5/0.65	0.7/0.85	0.3/0.46	**2.4**	0.0/0.0	0.0/0.0	0.0/0.0				

5 Summary

Through this paper, we probe deeper into the graph-reasoning capabilities of various LLMs. This study adopts a more granular approach towards analysing LLMs, in comparison to previous works [17]. First, we select a wider range of models from different model families (i.e. different architectures, training data and methods and distinct fine-tuning methods), and systematically compare the different models over reasoning tasks. Our problem design evaluates these models over increasingly complex problems and added constraints. Through this process, we highlight some interesting properties of LLMs. First off, we highlight various type of constraints that lead to performance drop (such as weighted edges and unordered node sequences). We also conclude that k-shot prompts are unhelpful for analytical tasks such as graph traversal. We also highlight how various models are biased towards providing a positive response. This, hence, leads to these LLMs providing wrong responses, even when no valid solution exists. Secondly, we highlight the root of poor graph-based reasoning in LLMs, and link this to the inability to backtrack solutions to compare and come to an optimal conclusion. Building on top of this, our novel prompting technique *PathCompare* helps mitigate this issue and improve graph reasoning abilities in LLMs.

A Appendix

A.1 Prompting

Base Prompt. We prompt models using adjacency matrices, labeled sequentially as *A,B,C...*, with 0 indicating no connections between 2 nodes, and 1 indicating otherwise. The model is asked to find the shortest path node A, to the last sequential node. An example prompt is as follows.

Given is the adjacency matrix for a unweighted undirected graph containing 10 nodes labelled A to J. The value corresponding to each row M and column N represents whether there is a connection between the two nodes, where 0 means no connection.

What is the shortest path from node A to node J? Return the sequence of nodes in response.

	A	B	C	D	E	F	G	H	I	J
A	0	1	0	0	0	0	0	0	0	0
B	1	0	1	0	0	0	0	0	0	0
C	0	1	0	1	0	0	0	0	0	0
D	0	0	1	0	1	0	0	0	0	0
E	0	0	0	1	0	1	0	0	0	0
F	0	0	0	0	1	0	1	0	0	0
G	0	0	0	0	0	1	0	1	0	0
H	0	0	0	0	0	0	1	0	1	0
I	0	0	0	0	0	0	0	1	0	1
J	0	0	0	0	0	0	0	0	1	0

Evaluation of the output of the models was performed manually, in order to factor in the variation of the output format by different models. In our code, input/output pairs (prompt/solution pairs) are automatically generated, which

allows us to create a large scale of variations. An example of the output string generated for this corresponding problem is

$$A \text{ -> } B \text{ -> } C \text{ -> } D \text{ -> } E \text{ -> } F \text{ -> } G \text{ -> } H \text{ -> } I \text{ -> } J$$

A more visually effective representation of the matrix is presented below, for the understanding of the users, although the above plain text version is used for prompting the LLMs. The numbers in red depict the connections that constitute the solution to the corresponding graph traversal.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
A	0	**1**	0	0	0	0	0	0	0	0
B	0	0	**1**	0	0	0	0	0	0	0
C	0	0	0	**1**	0	0	0	0	0	0
D	0	0	0	0	**1**	0	0	0	0	0
E	0	0	0	0	0	**1**	0	0	0	0
F	0	0	0	0	0	0	**1**	0	0	0
G	0	0	0	0	0	0	0	**1**	0	0
H	0	0	0	0	0	0	0	0	**1**	0
I	0	0	0	0	0	0	0	0	0	**1**
J	0	0	0	0	0	0	0	0	0	0

Adding K-Shot Prompts. Particularly for k-shot prompts (1-shot and 3-shot settings), we prepend randomly generated prompts to the task, as below.

Given is the adjacency matrix for a unweighted undirected graph containing 10 nodes labelled A to J. The value corresponding to each row M and column N represents whether there is a connection between the two nodes, where 0 means no connection.

Consider some examples

Example 1: What is the shortest path from node A to node O? Return the sequence of nodes in response.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
B	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
D	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0
E	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
F	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
G	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
H	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
I	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
J	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
K	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0
L	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
M	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
N	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
O	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Solution: A -> M -> N -> O

Given these examples, answer the following question. What is the shortest path from node A to node J? Return the sequence of nodes in response.

	A	B	C	D	E	F	G	H	I	J
A	0	1	0	0	0	0	0	0	0	0
B	1	0	1	0	0	1	0	0	0	0
C	0	1	0	1	0	0	0	0	0	0
D	0	0	1	0	1	0	0	0	0	0
E	0	0	0	1	0	0	0	0	0	0
F	0	1	0	0	0	0	1	0	0	0
G	0	0	0	0	0	1	0	1	0	0
H	0	0	0	0	0	0	1	0	1	0
I	0	0	0	0	0	0	0	1	0	1
J	0	0	0	0	0	0	0	0	1	0

Node Jumbling. An important variant of our graph traversal tasks involves jumbling the labels of the node order. This is primarily done, in order to avoid pretraining distribution biases in the model responses. For example, consider the adjacency matrix

	A	B	C	D	E	F	G	H	I	J
A	0	1	0	0	0	0	0	0	0	0
B	1	0	1	0	0	0	0	0	0	0
C	0	1	0	1	0	0	0	0	0	0
D	0	0	1	0	1	0	0	0	0	0
E	0	0	0	1	0	1	0	0	0	0
F	0	0	0	0	1	0	1	0	0	0
G	0	0	0	0	0	1	0	1	0	0
H	0	0	0	0	0	0	1	0	1	0
I	0	0	0	0	0	0	0	1	0	1
J	0	0	0	0	0	0	0	0	1	0

On jumbling, the new matrix looks something like

	H	C	D	F	I	E	G	A	J	B
H	0	1	0	0	0	0	0	0	0	0
C	1	0	1	0	0	0	0	0	0	0
D	0	1	0	1	0	0	0	0	0	0
F	0	0	1	0	1	0	0	0	0	0
I	0	0	0	1	0	1	0	0	0	0
E	0	0	0	0	1	0	1	0	0	0
G	0	0	0	0	0	1	0	1	0	0
A	0	0	0	0	0	0	1	0	1	0
J	0	0	0	0	0	0	0	1	0	1
B	0	0	0	0	0	0	0	0	1	0

which is essentially a renaming of node labels. Correspondingly, the modified task now involves finding the shortest node between the new "first" node and the new "last" node. Unsurprisingly, we observe a drop in the performance of models

when we introduce this variation, indicating at least some effect of pretraining data distribution bias (in the form of sequential nodel labeling *A,B,C,...*) in graph traversal tasks.

Weighted and Directed Graph Representations. While weighted connections are represented by integers ≥ 1, directed connections are represented by removing symmetricity accross the diagonal of the matrix. A simple example to illustrate this is shown below.

Given is the adjacency matrix for a weighted directed graph containing 16 nodes labelled A to P. The value corresponding to each row M and column N represents the cost of travelling between the two nodes, where 0 means no connection.

What is the least cost path from node A to node P? Return the sequence of nodes in response.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
E	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
F	0	0	0	0	1	0	0	0	0	4	0	0	0	0	0	0
G	0	0	3	0	0	4	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
I	0	0	0	0	0	0	0	0	0	5	0	0	2	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
K	0	0	0	0	0	0	1	0	0	5	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
O	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	2
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

References

1. Jin, B., Liu, G., Han, C., Jiang, M., Ji, H., Han, J.: Large language models on graphs: a comprehensive survey. arXiv preprint arXiv:2312.02783 (2023)
2. Liu, C., Wu, B.: Evaluating large language models on graphs: performance insights and comparative analysis. arXiv preprint arXiv:2308.11224 (2023)
3. Zhang, Z., et al.: LLM4DyG: Can Large Language Models Solve Problems on Dynamic Graphs? Association for Computing Machinery (2024)
4. Fatemi, B., Halcrow, J., Perozzi, B.: Talk like a graph: encoding graphs for large language models. arXiv preprint arXiv:2310.04560 (2023)
5. Chen, Z., Chen, L., Villar, S., Bruna, J.: Can graph neural networks count substructures? Adv. Neural. Inf. Process. Syst. **33**, 10383–10395 (2020)

6. Goldberg, A.V., Harrelson, C.: Computing the shortest path: a search meets graph theory. In: SODA, vol. 5, pp. 156–165 (2005)
7. Bengio, Y., LeCun, Y.: Scaling learning algorithms towards AI. In: Large Scale Kernel Machines. MIT Press (2007)
8. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**, 1527–1554 (2006)
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, vol. 1. MIT Press (2016)
10. White, J., Fu, Q., Hays, S., et al.: A prompt pattern catalog to enhance prompt engineering with ChatGPT. arXiv preprint arXiv:2302.11382 (2023)
11. Choudhary, N., Reddy, C.K.: Complex Logical Reasoning over Knowledge Graphs using Large Language Models. arXiv preprint arXiv:2305.01157 (2023)
12. Zelikman, E., Huang, Q., et al.: Parsel: a (de-)compositional framework for algorithmic reasoning with language models. In: Advances in Neural Information Processing Systems (Spotlight) (2023)
13. Bian, N., Han, X., et al.: ChatGPT is a knowledgeable but inexperienced solver: an investigation of commonsense problem in large language models. In: Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024) (2024)
14. Shridhar, K., Stolfo, A., et al.: Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. Association for Computational Linguistics (2023)
15. Yuan, Z., Yuan, H., et al.: Scaling relationship on learning mathematical reasoning with large language models. arXiv preprint arXiv:2308.01825 (2023)
16. Adhikari, A., Yuan, X., et al.: Learning dynamic belief graphs to generalize on text-based games. In: Advances in Neural Information Processing Systems, vol. 33, pp. 3045–3057 (2020)
17. Wang, H., Feng, S., et al.: Can language models solve graph problems in natural language? In: Advances in Neural Information Processing Systems (Spotlight) (2023)
18. Touvron, H., Martin, L., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
19. Anil, R., Dai, A.M., et al.: Palm 2 technical report. arXiv preprint arXiv:2305.10403 (2023)
20. Wu, S., Koo, M., et al.: A Comparative Study of Open-Source Large Language Models: GPT-4 and Claude 2: Multiple-Choice Test Taking in Nephrology. NEJM AI (2024)
21. OpenAI: GPT-4 Technical Report. arXiv:2303.08774 (2023)
22. Wei, J., Wang, X., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural. Inf. Process. Syst. **35**, 24824–24837 (2022)
23. Qiao, S., Ou, Y., et al.: Reasoning with language model prompting: a survey. Association for Computational Linguistics (2023)
24. Chai, Z., Zhang, T., et al.: Graphllm: boosting graph reasoning ability of large language model. arXiv preprint arXiv:2310.05845 (2023)
25. Saba, W.S.: Stochastic LLMs do not understand language: towards symbolic, explainable and ontologically based LLMs. In: Almeida, J.P.A., Borbinha, J., Guizzardi, G., Link, S., Zdravkovic, J. (eds.) ER 2023. LNCS, vol. 14320, pp. 3–19. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47262-6_1
26. Creswell, A., Shanahan, M.: Faithful reasoning using large language models. In: International Conference on Learning Representations (poster) (2024)

27. Wang, X., Wei, J., et al.: Self-consistency improves chain of thought reasoning in language models. In: International Conference on Learning Representations (poster) (2023)
28. Brown, T., Mann, B., et al.: Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. **33**, 1877–1901 (2020)

Improved Zero-Shot Image Editing via Null-Toon and Directed Delta Denoising Score

Masud An Nur Islam Fahim(✉) and Jani Boutellier

University of Vaasa, Vaasa, Finland
{masud.fahim,jani.boutellier}@uwasa.fi

Abstract. Recently, there has been a rapid surge in the utilization of diffusion models for customized image generation and editing tasks, especially using zero-shot editing algorithms that can largely operate on given images regardless of their source domain. This work is based on two well-known zero-shot image editing algorithms: Null Text Inversion (NTI) and Delta Denoising Score (DDS). With respect to NTI, we mainly focus on image cartoonization, which has received less attention in the context of text-guided image editing. In a nutshell, we propose a customized reconstruction phase for NTI, which helps transforming the natural input image into cartoon images with desired customization by supporting parameters. We also improve the current DDS optimization baseline and propose the Directed Delta Denoising Score (DDDS). Our DDDS algorithm offers a better image editing experience by replacing the target text prompt with the proposed *directed text prompt*. Computing directed text prompt requires one subtraction operation and yields significant reconstruction improvement over DDS. To demonstrate the effectiveness of our contributions, the paper presents both quantitative and qualitative comparisons against the state-of-the-art, as well as several visual examples.

Keywords: Diffusion model · Zero-shot editing · Image generation

1 Introduction

Diffusion models [10,18,20,21] have shown great promise in text-guided image editing, especially with natural image editing tasks like image inpainting [4,13,14,25], style transfer [2,8,22], text-guided editing [2,7,8,11,15,24], or segmentation [1,3]. Our study is particularly interested in text-guided image editing of natural images, where we want to edit natural images based on prompt input.

Editing natural images with a diffusion model is not straightforward. Typically, zero-shot editing studies rely on inversion strategies like DDIM [21] to access the desired latent space of the inputs. The choice of inversion strategy is critical for reconstruction fidelity. Null Text Inversion (NTI) [15] is a popular method for image editing that relies on Denoising Diffusion Implicit Models (DDIM) inversion. In this study we have observed that using NTI it is possible to transform any natural image into a cartoonified appearance by modifying the

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 309–323, 2025.
https://doi.org/10.1007/978-3-031-78172-8_20

(a) Input image (b) NTI output [15] (c) Our output

Fig. 1. Here, output quality comparison between the Null Text Inversion (NTI) [15] and the proposed cartoon reconstruction algorithm: the proposed algorithm's output is free from text artifacts (image center) in contrast to the NTI output. Additionally, our algorithm cartoonifies the whole image, while [15] focuses on the dog region only.

(a) Input image (b) NTI output [15] (c) Our output

Fig. 2. Image editing comparison between the delta denoising score (DDS) [7] and the proposed directed delta denoising score (DDDS). Here, the source prompt is *Two cats sitting by the mirror*, and the target prompt is *Two* ***origami*** *cats sitting by the mirror*. DDS edited regions are blurry and missing salient details. In contrast, our DDDS offers crisper output, including reflection area.

reconstruction path leveraging predicted noise via weighted augmentation, and optimized null text perturbation. Additionally, parameterized noise perturbation is considered to increase or decrease the desired degree of output image detail. Since our approach stands upon the null text inversion approach and is dedicated to the cartoon transformation of the natural images, we name our approach *Null-Toon*. A recent study [26] also showed that it is possible to translate natural images into cartoons using null text optimization. However, in contrast to the proposed work, their process does not offer parameterized tunability. Moreover, our algorithm returns cartoon images without any traces of text artifacts, as shown in Fig. 1.

In addition to the particular cartoon reconstruction approach, we complementarily propose another method for superior general-purpose zero-shot image editing. Delta Denoising Score (DDS) [7] is an efficient alternative to the inversion-dependent methods [11,15,23]. DDS harnesses the core idea of Score Distillation Sampling (SDS) [19] and improves over SDS. In practice, SDS returns noisy gradients during optimization, resulting in unsatisfactory editing performance. DDS rectifies SDS by taking in two sets of latent-text pairs and performing SDS optimization for individual pairs, subtracting the estimated

gradients from each pair [7]. This novel *delta* gradient obtained from subtraction is cleaner and contains indicates direction for editing (Fig. 2).

However, reconstructed images from this *delta* gradient often contain traces of the source image, resulting in unfavorable edits [7]. We rectify this situation by proposing *directing* the target prompt via weighted subtraction. This elementary change helps in drastic reconstruction performance over DDS, naming our method as Directing Delta Denoising Score (DDDS). Our DDDS approach only needs an extra subtraction operation as an additional computation cost to DDS, and for the same optimization configuration, editing performance is superior over DDS. We show the necessary visual comparison between our study and other studies in the later section of the manuscript and summarize overall contributions as follows:

- An algorithm for cartoonifying natural images using null-text reconstruction,
- Providing user-adjustability to algorithms output via parametrized noise perturbation and
- Proposing the novel Directed Delta Denoising Score (DDDS) for crisper output, compared to the existing Delta Denoising Score (DDS).

2 Related Work

Text-to-image models [10,21] have shown great promise in image generation conditioned by an input text prompt. These works harness powerful diffusion models for generation or guided reconstruction tasks. Recently, we have seen a surge of zero-shot image editing approaches where customized algorithms provide general-purpose image editing applications without tuning large diffusion models. These image editing can roughly be divided into the following clusters: end-to-end editing [5,12,18], attention manipulation [2,6,8,16], and DDIM inversion families [11,15,24]. Score-based editing approaches [7,17,19] are an efficient alternative to inversion-dependent studies, where image editing is done end-to-end. Our study is related to both score and inversion-based studies.

3 Proposed Algorithms

We begin by revisiting the basic concepts regarding the diffusion model: we represent the dataset as $\boldsymbol{\mathcal{D}}$, clean image or latent as $\boldsymbol{z_0}$, text embedding (source) as $\boldsymbol{y}$, noise as ϵ, T as time step, and the diffusion model as ϵ_θ. Typically, the following objective is used to describe the training of a diffusion model:

$$\min_{\theta} \mathbb{E}_{\mathbf{z}_0 \sim \mathcal{D}, \boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I}), \mathbf{t} \sim U(1,T)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\mathbf{z}_t, \mathbf{y}, \mathbf{t} \right) \right\|^2$$

In the above, $\mathcal{D}$ denotes the utilized dataset, $\mathbf{z}_t$ is a noisy iteration of the image $\mathbf{z}_0$ at timestep $\mathbf{t}$, and $\mathbf{y}$ is a corresponding conditional embedding. After model training, we can sample a new image given condition $\mathbf{c}$ with the commonly used DDIM sampler,

Algorithm 1: Image reconstruction from Null text trajectory

Input: Noisy latent $\mathbf{z}_T$, Null text trajectory $\mathcal{Y}_\phi$, target text $\hat{\mathbf{y}}$, balance parameter ω
Output: Edited latent $\mathbf{z}_0^*$

Set $\mathbf{z}_t = \mathbf{z}_T$;
for $t \leftarrow T$ **to** 0, $i \leftarrow 1$ **to** $len(\mathcal{Y}_\phi)$ **do**
  $\mathbf{y}_t^\phi = \mathcal{Y}_\phi[i]$
  $\epsilon_t \leftarrow \mathcal{CFG}(\mathbf{z}_t, \mathbf{y}_t^\phi, \hat{\mathbf{y}}, \mathbf{t}, \omega)$, Eq. 1
  $\mathbf{z}_{t-1}^* \leftarrow DDIM(\mathbf{z}_t, \epsilon_t, \mathbf{t})$
end
Return adapted latent $\mathbf{z}_0^*$

$$\begin{aligned}\mathbf{z}_{t-1} &= \mathrm{DDIM}\left(\mathbf{z}_t, \boldsymbol{\epsilon}_t, \mathbf{t}\right) \\ &= \sqrt{\alpha_{t-1}} \cdot f_\theta\left(\mathbf{z}_t, \mathbf{c}, t\right) + \sqrt{1-\alpha_{t-1}} \cdot \boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t, \mathbf{y}, t\right),\end{aligned}$$

where $f_\theta\left(\mathbf{z}_t, \mathbf{y}, \mathbf{t}\right) = \frac{\mathbf{z}_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta(\mathbf{z}_t, \mathbf{y}, \mathbf{t})}{\sqrt{\alpha_t}}$, $\boldsymbol{\epsilon}_t = \boldsymbol{\epsilon}_\theta\left(\mathbf{z}_t, \mathbf{y}, \mathbf{t}\right)$, and α_t is a time-varying noise parameter in DDIM. For our study, we consider the above formula as the DDIM sampler.

Now, if we want to apply the diffusion model for local or global edits, it is necessary to invert them into $\mathbf{z}_0 \rightarrow \mathbf{z}_T$, DDIM inversion [21] being a popular way to do this. After having $\mathbf{z}_T$, we can expect a lossless return to $\mathbf{z}_0$ with perfect reconstruction, which is unfortunately not possible in reality. To address this, null text optimization [15] has been proposed as a highly efficient way to recover $\mathbf{z}_0^*$, which is very close to the original $\mathbf{z}_0$. Once we have near-perfect reconstruction approach, the influence from the target prompt or text embedding $\hat{\boldsymbol{y}}$ can be integrated to reconstruct the input image with the desired edits using optimized null text embeddings and $\hat{\boldsymbol{y}}$ via classifier-free guidance [9].

By design, null text optimization returns a trajectory of unconditional text embeddings $\mathcal{Y}_\phi = [\boldsymbol{y_T}_\phi, ..., \boldsymbol{y_0}_\phi]$, and by utilizing them for any time step $\mathbf{t}$, classifier-free guidance can be expressed as follows:

$$\epsilon_t^* = \epsilon_\theta(\mathbf{z}_t, \mathbf{t}, \mathbf{y}_t^\phi) + \omega * (\epsilon_\theta(\mathbf{z}_t, \mathbf{t}, \hat{\mathbf{y}}) - \epsilon_\theta(\mathbf{z}_t, \mathbf{t}, \mathbf{y}_t^\phi)) \tag{1}$$

Here, ω is the impact factor for balancing the information contents from the source and target content. By leveraging the above concepts, we can denote the prompt-guided reconstruction steps as follows (Algorithm 1):

Using Algorithm 1, we can obtain $\mathbf{z}_0^*$ that contains the natural image with respect to the target prompt $\hat{\mathbf{y}}$.

3.1 Null-Toon

Having described the necessary concepts above, we can proceed with a detailed description of the proposed Null-Toon algorithm based on Algorithm 1. After completing null text optimization, we get the optimized null text trajectory

Algorithm 2: Cartoonifying from Null text trajectory

Input: Noisy latent $\mathbf{z}_T$, Null text trajectory $\mathcal{Y}_\phi$, target text $\hat{\mathbf{y}}$, Predicted noise trajectory $\mathcal{E}$, trajectory length $\mathcal{Q}^\phi$, Convex weights $\mathcal{A}$, balance parameter ω. time-steps t_1, t_2, coefficients c_1, c_2, c_3, c_4, c_5

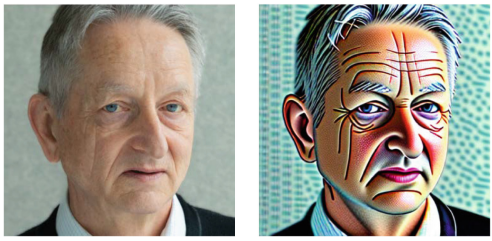
Output: Edited latent $\mathbf{z}_0^*$

1 Set $\mathbf{z}_t = \mathbf{z}_T$;
2 **for** $\mathbf{t} \leftarrow T$ **to** 0, $i \leftarrow 1$ **to** $\mathcal{Q}^\phi$ **do**
3 Set $w_1 = \mathcal{A}[i]$ & $w_2 = 1 - w_1$;
4 $\mathcal{Q}^\phi$ - i ;
5 $\mathbf{y}_t^\phi = \mathcal{Y}_\phi[i]$
6 **if** $\mathbf{t} < t_1$ **then**
7 $\mathbf{y}_t^\phi = \mathbf{y}_t^\phi + c_1 * \mathcal{Y}_\phi$ [$\mathcal{Q}^\phi$ - i]
8 **end**
9 $\epsilon_t \leftarrow \mathcal{CFG}(\mathbf{z}_t, \mathbf{y}_t^\phi, \hat{\mathbf{y}}, \mathbf{t}, \omega)$, Eq. 1
10 **if** $\mathbf{t} > t_2$ **then**
11 $\epsilon_t = \epsilon_t + c_2 * \mathbf{z}_t$
12 **end**
13 $\epsilon_t = w_1 * \mathcal{E}[i] + w_1 * \epsilon_t$
14 $\epsilon_t = ConvexSum(\mathcal{E}[i], \epsilon_t, c_3)$
15 $\epsilon_t^s = Enhance(\epsilon_t, c_4)$ | **Enhance sharpness** |
16 $\epsilon_t^d = Enhance(\epsilon_t, c_5)$ | **Enhance smoothness** |
17 $\epsilon_t = ConvexSum(\epsilon_t^d, \epsilon_t^s, c_3)$
18 $\mathbf{z}_{t-1}^* \leftarrow DDIM(\mathbf{z}_t, \epsilon_t, \mathbf{t})$
19 **end**
20 Return adapted latent $\mathbf{z}_0^*$

$\mathcal{Y}_\phi$, and the length of the trajectory is $\mathcal{Q}^\phi$. Based upon this trajectory with the help of Algorithm 1, we first reconstruct the original image while setting $\hat{\mathbf{y}}$ as an empty statement to introduce no change in the reconstruction. This self-reconstruction also allows us to collect the predicted noise trajectory $\mathcal{E}$, containing the output noise $\mathbf{y}_t^\phi$ from the classifier free guidance step. Apart from $\mathbf{z}_T$, and $\mathcal{Y}_\phi$, we consider $\mathcal{E}$ as another input to our toonification algorithm along with a few other parameters as coefficients and threshold conditions. We formulate our toonification approach in Algorithm 2.

Similar to Algorithm 1, we start from anchoring the latent $\mathbf{z}_T$, and setting $\mathbf{z}_t = \mathbf{z}_T$ to return back to $\mathbf{z}_0$. Based upon on Algorithm 1, we propose Algorithm 2. Below, we explain the impact of specific pseudocode lines with visual examples.

Augmenting null text (Algorithm 2 line 6) with *future* optimized null text brings in the main cartoonifying impact through drastic loss in image detail, as shown in Fig. 3. In our algorithm, *future* represents the optimized null texts from later stages of the iteration; we already have access to the null text trajectory and access those texts in reverse. If we use Algorithm 1 for reconstruction and augment the null text for a given time step by recalling the null text from the future time step, we observe such cartoon style reconstruction. However, we impose a threshold for the time step to preserve the image details. We obtained the optimal value for the timestep thresholds via trialing on multiple images.



(a) Input image (b) Output image

Fig. 3. Impact of augmenting *future* null text with Algorithm 1.

(a) Impact of augmenting $\mathbf{z}_t$ to the $\epsilon_\mathbf{t}$. (b) Impact of augmenting $\mathcal{E}[i]$ to the $\epsilon_\mathbf{t}$.

Fig. 4. Impact of augmenting $\mathbf{z}_t$ and $\mathcal{E}[i]$ during cartoon reconstruction.

As the augmentation procedure provides the cartoon effect and loss of detail, an appropriate rectification is required to recover detail again, which is acquired through later steps of the proposed Null-Toon Algorithm 2.

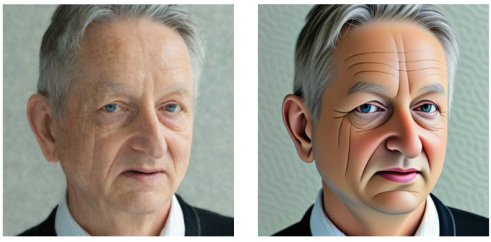
Augmenting latent (Algorithm 2 line 10) with the predicted noise $\epsilon_\mathbf{t}$ helps reducing loss of detail while preserving comic-like appearance. Augmenting the estimated latent $\mathbf{z}_t$ with $\epsilon_\mathbf{t}$ provides reconstruction as depicted in Fig. 4. From Fig. 4, we can see that with the dominance of the smoothing effect due to latent augmentation, content details have disappeared significantly. Optimally, we would like to balance between the two extreme conditions while preserving the cartoonish appearance.

Augmenting noise from null-text optimization (Algorithm 2 lines 12–13, 16) with the predicted noise $\epsilon_\mathbf{t}$ helps in regaining more image details. From Algorithm 1, null-text reconstruction relies on optimized null text trajectory only [15]. Another study [24] showed that null text optimization returns a trajectory of noise $\mathcal{E}$ that contains predicted noise $\epsilon_\mathbf{t}$ while $\mathbf{z}_0 \rightarrow \mathbf{z}_T$, and it is possible to utilize $\mathcal{E}$ in Algorithm 1 after the CFG step to gain more context preserving reconstruction [24] while reverting $\mathbf{z}_T \rightarrow \mathbf{z}_0$.

We have adopted this idea to integrate more details into our reconstruction. Following [24], we get a series of convex weights $\mathcal{A}$, tailored for each iteration of the diffusion step, and based upon these weights, we perform the convex sum between the predicted noises from $\mathcal{E}$, and current estimation of $\epsilon_\mathbf{t}$. To reinforce

(a) Detail-rich output (b) Low detail output

Fig. 5. Impact of the convex weights: $\mathbf{c_4} = 0.96$ causes more details, whereas $\mathbf{c_5} = 1.01$ causes overtly smooth output.



(a) Input image (b) Output image

Fig. 6. Final reconstruction from the proposed Algorithm 2.

the details, we perform another extra convex summation step as present in the Algorithm 2. Consequently, as shown in Fig. 4, our current reconstruction is layered with a desirable cartoon look and slightly lacking realistic details.

Perturbing the predicted noise (Algorithm 2 lines 14–15) $\epsilon_\mathbf{t}$ helps in emphasizing smoothness or details if we apply our proposed mean-directed perturbation. In summary, our mean-directed perturbation is done by following steps:

- Say $\epsilon_\mathbf{t}$ consists of $\mathbf{n}$ channels, and we estimate the mean $\bar{\epsilon}_\mathbf{t}$ across all channels.
- We add noise to $\mathbf{n}$ instances of $\bar{\epsilon}_\mathbf{t}$, and concatenate them together as $\hat{\epsilon}_\mathbf{t}$, obtaining the same shape as $\epsilon_\mathbf{t}$.
- Performing a convex sum between $\epsilon_\mathbf{t}$ and $\hat{\epsilon}_\mathbf{t}$ results in less or more details, depending on the applied coefficient, as shown in Fig. 5.

Figure 5 highlights that manipulation of $\epsilon_\mathbf{t}$ via $\mathbf{c_4}$ and $\mathbf{c_5}$ affects the level of detail in the output. Merging both together through a final convex summation provides the desired $\epsilon_\mathbf{t}$, which balances the degree of detail, smoothness, and cartoon effect together. Performing DDIM sampling upon the final sum until the last iteration returns the desired cartoon image as present in Fig. 6. In our algorithm, Convex sum means regular convex summation operation.

Our proposed Null-Toon uses several hyperparameters to achieve the desired outcome. Most of the hyperparameters are empirically found and need only min-

imal tuning during reconstruction. Among those parameters, **c1**, **c2** and **c3** are fixed to values **4.5**, **3.5**, **0.5**, respectively. **c4** and **c5** remain adjustable to balance between cartoonish appearance and image details. In our experiments, **c4** ranged between $[0.95, 0.98]$, and **c5** between $[1.02, 1.04]$. For the convex weights $\mathcal{A}$, we followed the weight distribution process of [24]. We kept $t_1 = 220$ and $t_2 = 920$, which was empirically obtained first via trialing on several natural images. and later set as a constant during the test.

3.2 Directed Delta Denoising Score (DDDS)

Starting from DDS [7], we can write the image editing Eq. 2 for Score Distillation Sampling (SDS) as follows:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}, \epsilon, \mathbf{t}) = \epsilon_\theta^\omega \left((\hat{\mathbf{z}}_\mathbf{t}, \mathbf{t}) - \epsilon \right) \frac{\partial \hat{\mathbf{z}}_\mathbf{t}}{\partial \theta} \tag{2}$$

However, editing with the formulation above results in a blurry output [7], often flattening out the context along with the regions, showing alignment towards the text prompt $\mathbf{y}$. The rationale for the above situation is more understandable via the following decomposition:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}, \epsilon, \mathbf{t}) := \delta_{text} + \delta_{bias} \tag{3}$$

According to DDS [7], δ_{text} makes the effective gradient drift towards the desirable direction that aligns with the prompt, and δ_{bias} conditions the outcome towards undesirable directions. The authors of DDS [7] provide an interesting solution to the above Eq. 3 by

$$\nabla_\theta \mathcal{L}_{\text{DDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}, \epsilon, \mathbf{t}) = \nabla_\theta \mathcal{L}_{\text{SDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}) - \nabla_\theta \mathcal{L}_{\text{SDS}}(\mathbf{z}, \mathbf{y}). \tag{4}$$

DDS [7] showed that $\nabla_\theta \mathcal{L}_{\text{DDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}, \epsilon, \mathbf{t})$ is almost equivalent to δ_{text}, because Eq. 4 returns a cleaner gradient by performing the proposed subtraction. In practice, Eq. 4 is effective, because in each iteration it estimates two directions dedicated to the target text and source text. Subtraction between two of them returns a unique direction that contains the necessary edits from the $\nabla_\theta \mathcal{L}_{\text{SDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}})$, and additional context bias from $\nabla_\theta \mathcal{L}_{\text{SDS}}(\mathbf{z}, y)$.

However, Eq. 4 fails to capture the desired edits in many cases, and the reason is apparent from the formulation of DDS. In Eq. 4, the editing direction is obtained from the gradient, and the gradient stems from the estimated noise difference between $\hat{\mathbf{z}}, \hat{\mathbf{y}}$ and $\mathbf{z}, \mathbf{y}$ pairs. For any given time step $\mathbf{t}$, from $\hat{\mathbf{z}}, \hat{\mathbf{y}}$ we get $\hat{\epsilon}_\mathbf{t}$, and ϵ_t from the $\mathbf{z}, \mathbf{y}$ pair. Here, we argue that if the difference between $\hat{\epsilon}_\mathbf{t}$, and ϵ_t is more biased towards the desired edit and minimal towards the context from the original image, then we can capture refined details that match $\hat{\mathbf{y}}$.

We can only emphasize the target prompt in the DDS setup by tuning ω in the classifier-free guidance stage. As a result, the DDS gradient is trimmed by the influence from ϵ_t, preserves the context, limits the edit fidelity, and improves the fundamental lack of image details present in SDS. This gives us two objectives to satisfy, which are listed below, along with proposed solutions:

- *How can* $\hat{\mathbf{y}}$ *be emphasized more?* We address this question with our proposed subtraction operation, where we direct the target prompt $\hat{\mathbf{y}}$ by $\hat{\mathbf{y}}_\mathbf{d} = \alpha_1 * \hat{\mathbf{y}} - \alpha_2 * \mathbf{y}$. Here, α_1 typically ranges between [1.1, 1.3], and $\alpha_2 = 0.05$. By this, we obtain a *directed* prompt that is more biased towards the intended image edits.
- *How can the effect of* $\mathbf{y}$ *be reduced?* In both SDS and DDS, $\mathbf{y}$ steers the gradient direction towards the source context, preserving the source image's similarity where editing is unnecessary. However, $\mathbf{y}$ reduces detail in the edited image in SDS, but results in undesirable results in DDS. We minimize this situation by considering an auxiliary null-text embedding $\mathbf{y_n} = \Phi$. In other words, we have used $\mathbf{y_n}$ instead of $\mathbf{y}$ in the original DDS equation. As a result, in the optimization phase, $\mathbf{y}$ puts least impact on the estimated gradients without causing undesirable alteration.

To summarize, our directed delta denoising score (DDDS) returns a cleaner gradient than DDS by projecting $\hat{\mathbf{y}}$ to $\hat{\mathbf{y}}_\mathbf{d}$, where editing embedding gets more attention. Additionally, to reduce the over-emphasizing effect of source embedding $\mathbf{y}$, we replace it with a null-text embedding $\mathbf{y_n}$, and $\mathbf{y_n} = \Phi$. Hence, the proposed directed delta denoising score is as follows:

$$\nabla_\theta \mathcal{L}_{\text{DDDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}_\mathbf{d}, \epsilon, \mathbf{t}) = \nabla_\theta \mathcal{L}_{\text{SDS}}(\hat{\mathbf{z}}, \hat{\mathbf{y}}_\mathbf{d}) - \nabla_\theta \mathcal{L}_{\text{SDS}}(\mathbf{z}, \mathbf{\Phi}) \tag{5}$$

In Eq. 5 estimated gradients are more precise than the gradients of Eq. 4 due to the above-mentioned reasons. We present the necessary demonstration in the following sections to support our claim.

4 Results

This section presents a visual comparison between the proposed methods and previous studies. We have listed separate baseline methods for each technique to present a fairground for the comparisons. Our Null-Toon algorithm uses null text inversion as its core process; we select the Null Text Inveriosn (NTI) [15], Direct Inversion (DI) [11], and Null Text Guidance (NTG) [26] to compare the cartoon translation performance. These three works use null text optimization as the central part of the given zero-shot image editing task. Similarly, we compare our directed delta denoising score algorithm with two other score-dependent algorithms: Delta Denoising Score (DDS) [7] and Contrastive Denoising Score (CDS) [17].

4.1 Implementation Details

For all the comparisons with baselines, we refer to the official code repositories of the respective studies. In the case of Null-Toon, we have not changed hyperparameters for other studies. However, score-based algorithms [7,17] depend heavily on the iteration count. Given the complexity of the image and corresponding target prompt, the iteration count can vary from case to case. Our experiments

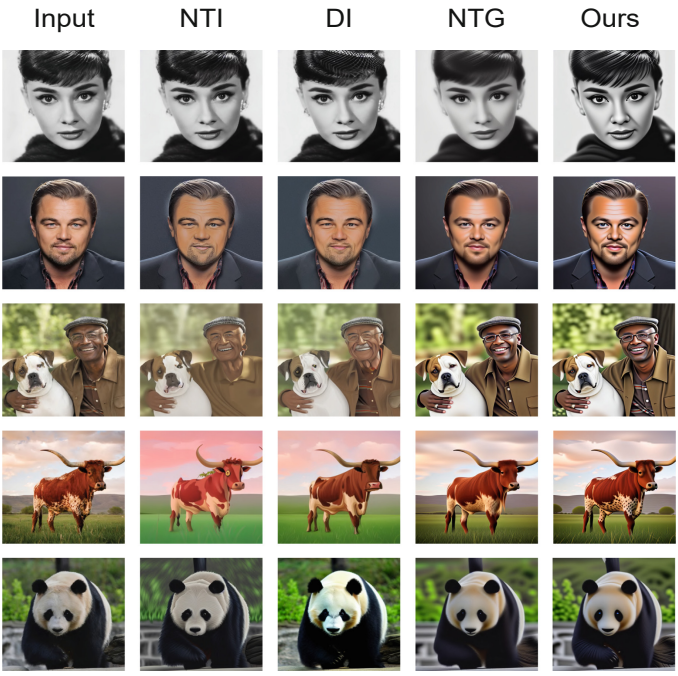


Fig. 7. Comparison of cartoon reconstruction between NTI [15], DI [11], NTG [26], and ours stands for the proposed Null-Toon algorithm.

showed that a higher iteration count results in over-editing and vice versa; pinpointing the ideal iteration count is not viable either. To present the results, we run DDS [7], CDS [17], and the proposed DDDS for 350 iterations.

4.2 Editing Comparison

Figure 7 demonstrates the visual comparison between the proposed cartoon translation algorithm and other baselines [11,15,26]. As shown in the topmost row of Fig. 7, our translation algorithm returns a crisper cartoon edit, whereas NTG [26] returns blurry output, and the other algorithms [11,15] perform almost no change to the input. In the second row, all of the baselines [11,15,26] provide reasonable editing results, but NTI [15] and DI [11] slightly alter the personal appearance of the character. Exaggerating the character is prevalent with NTI [15] and DI [11]; clearly, the character has become older, whereas NTI [15], and NTG [26] alter the dress of the character as well. Compared to these, our algorithm does not emphasize the sensitive features of the characters, which is evident from the given examples.

This trend is consistent with the rest of the examples as well, where our algorithm does not bend the horn of the cow or loosely reconstruct the leg of the panda while translating the images into cartoons. We can conclude that

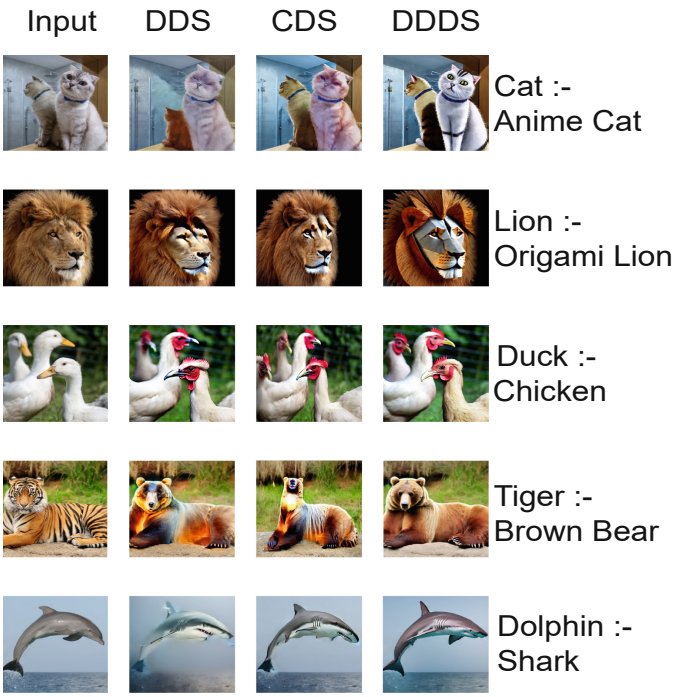


Fig. 8. Here, we show the zero-shot editing comparison between DDS [7]. CDS [17], and our DDDS. From the visual appearance, our algorithm offers better editing fidelity and reconstruction that closely matches the target prompt.

our cartoon algorithm can cartoonify natural photos without altering the salient features.

From Fig. 8, we can see the editing result comparison between the proposed directed delta denoising score (DDDS) algorithm and other baselines [7,17]. In the first row, our algorithm and [17] have returned the respective reconstruction of the cat image into anime art, unlike [7], which distorts the cat appearance in its reconstruction. In the second row, the task is to transform the lion's head into an origami structure. Surprisingly, both DDS [7] and CDS [17] return the output with minimal deviation from the input. On the contrary, our approach almost translates the lion's head into an origami-looking head. In the next row, DDS [7] transforms the chicken's beak with a metal look, and CDS [17] brings an extra chicken head on the top of the tail of the other chicken. Our method does not return such unnatural edits, but it does change the color of one chicken into a yellowish look, which is an anomaly compared to the all-white ducks from the input.

Likewise, our method remains consistent with its output quality with the rest of the images; for example, our algorithm transforms the tiger into a bear without distorting its head like CDS [17] does, or dolphin to a shark without

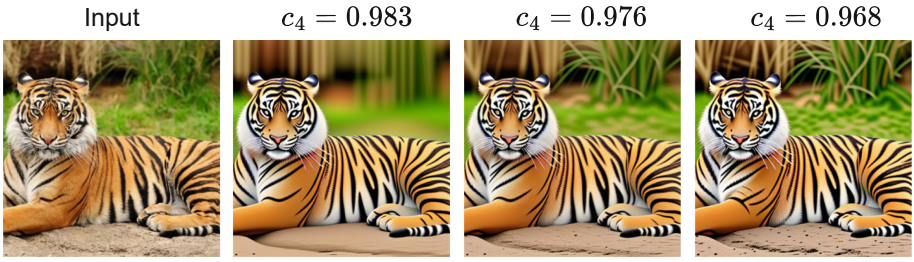


Fig. 9. In this picture, we show the impact of parameter c_4 on controlling image detail while translating the input image into a cartoon image using our Null-Toon algorithm. It is evident that lowering c_4 preserves the cartoon image closer to the original.

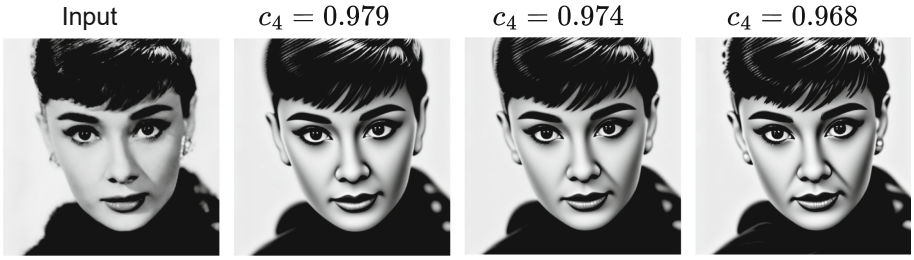


Fig. 10. The impact of parameter c_4 on a human face: by reducing c_4, we can bring more detail to the character's face.

changing the context like [7]. From this comparison, our algorithm brings significant improvement and consistency with zero-shot image editing tasks using score-based methods.

4.3 Impact of c_4

In Figs. 9 and 10, we show the impact of hyperparameter c_4 on our proposed cartoon reconstruction algorithm. We can see that lowering c_4 leads to higher image details for both images. Especially in Fig. 9, it can be seen that increasing c_4 greatly affects the amount of detail in the background, simultaneously resulting in a sharper cartoon image of the tiger. In Fig. 10, lowering the value of c_4 makes the edges of the face become sharper without losing the baseline cartoon look. Although our cartoon reconstruction algorithm has several other hyperparameters as well, we kept these fixed during the ablation study. Due to space limitations, it is not possible to explore the effects of other hyperparameters.

4.4 Local Edits Using DDDS

In Fig. 11, we show the performance comparison for editing local regions between delta denoising score [7] and our study. Here, we take an image of a dog and

Table 1. CLIP-similarity and LPIPS comparison between DDS [7], CDS [17], and the proposed DDDS. For CLIP, higher is better and for LPIPS lower is better.

Algorithm	CLIP similarity	LPIPS
DDS [7]	32.19	$0.14 \pm .07$
CDS [17]	33.06	$0.14 \pm .02$
DDDS	**33.79**	$\mathbf{0.13 \pm .04}$

Fig. 11. Here we demonstrate the editing difference between our algorithm and the baseline DDS [7] approach. Our algorithm achieved better visual reconstruction performance in all cases while keeping minimal context distortion.

add men's accessories to the dog, which were not present in the original image. For the sunglasses, we see that our algorithm can return an edited image where the added sunglasses appear more natural than with DDS [7]. Similarly, while adding a hat on the top of the dog's head, our algorithm can edit the region without distorting the original structure of the dog. On the contrary, one eye of the dog is undesirably edited while adding hat by DDS [7]. Finally, the dog's tie is semi-reconstructed, and the dog's color is drastically shifted by DDS [7] while adding a suit to the dog's image. Compared to that, editing results from our algorithm are more crisp and character-preserving.

4.5 After Edit Text Image Consistency

To evaluate the editing performance of our proposed DDDS algorithm, we used text-image similarity evaluation after editing completion. For this evaluation, we have used CLIP similarity, and this approach is adopted in other zero-shot editing studies as well [11,19]. In the CLIP similarity evaluation, we take in the edited image and its corresponding prompt and then calculate their similarity by projecting the image and text into the same embedding space. However, we can obtain this evaluation in two ways: estimating the similarity score for the edited region only by using a human-annotated mask or the whole image where we do not mask the non-edited regions. For our study, we have estimated the CLIP similarity score for the entire image, as presented in Table 1. Note that the CLIP

similarity score heavily relies on the rich textual description, and depending on the text prompt, we observed variation within the estimated scores. For this evaluation, we have compared our DDDS algorithm with DDS [7] and [17]. From the table, the obtained similarity score from our DDDS algorithm outperforms other methods for the edited images. We also include learned perceptual image patch similarity (LPIPS) score by following [7,17]. As shown in Table 1, the proposed work also achieves satisfactory performance using LPIPS metric.

5 Conclusion

In our study, we proposed two different methods for zero-shot image editing. Our first algorithm is dedicated to image cartoonization. We devised a unique reconstruction process by leveraging the null text trajectory to revise the previous noisy latent to introduce the cartoonization effect. In our reconstruction phase, we enabled a novel tuning setup for the user to include desired control over image details or smoothness without tampering the content. In our second algorithm, we show that *directing* the target text prompt via weighted subtraction between the source and target text prompt. As a result, our DDDS optimization returns more *directed* gradients and results in a more realistic image editing result without introducing zero computation increase over the original DDS optimization algorithm. We plan to extend our work to video and 3D instances in the future.

Acknowledgements. This work was partially funded by the ERDF project AI2Business.

References

1. Baranchuk, D., Rubachev, I., Voynov, A., Khrulkov, V., Babenko, A.: Label-efficient semantic segmentation with diffusion models. arXiv preprint arXiv:2112.03126 (2021)
2. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: learning to follow image editing instructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18392–18402 (2023)
3. Burgert, R., Ranasinghe, K., Li, X., Ryoo, M.S.: Peekaboo: text to image diffusion models are zero-shot segmentors. arXiv preprint arXiv:2211.13224 (2022)
4. Choi, J., Kim, S., Jeong, Y., Gwon, Y., Yoon, S.: ILVR: conditioning method for denoising diffusion probabilistic models. arXiv preprint arXiv:2108.02938 (2021)
5. Geng, Z., et al.: Instructdiffusion: a generalist modeling interface for vision tasks. arXiv preprint arXiv:2309.03895 (2023)
6. Han, L., et al.: Proxedit: improving tuning-free real image editing with proximal guidance. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 4291–4301 (2024)
7. Hertz, A., Aberman, K., Cohen-Or, D.: Delta denoising score. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2328–2337 (2023)

8. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D.: Prompt-to-prompt image editing with cross attention control. arXiv preprint arXiv:2208.01626 (2022)
9. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
10. Hoogeboom, E., Heek, J., Salimans, T.: Simple diffusion: end-to-end diffusion for high resolution images. In: International Conference on Machine Learning, pp. 13213–13232. PMLR (2023)
11. Ju, X., Zeng, A., Bian, Y., Liu, S., Xu, Q.: Direct inversion: boosting diffusion-based editing with 3 lines of code. arXiv preprint arXiv:2310.01506 (2023)
12. Kim, G., Kwon, T., Ye, J.C.: Diffusionclip: text-guided diffusion models for robust image manipulation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2426–2435 (2022)
13. Li, W., Yu, X., Zhou, K., Song, Y., Lin, Z., Jia, J.: SDM: spatial diffusion model for large hole image inpainting. arXiv preprint arXiv:2212.02963, vol. 1 (2022)
14. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Repaint: inpainting using denoising diffusion probabilistic models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11461–11471 (2022)
15. Mokady, R., Hertz, A., Aberman, K., Pritch, Y., Cohen-Or, D.: Null-text inversion for editing real images using guided diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6038–6047 (2023)
16. Mou, C., Wang, X., Song, J., Shan, Y., Zhang, J.: Dragondiffusion: enabling drag-style manipulation on diffusion models. arXiv preprint arXiv:2307.02421 (2023)
17. Nam, H., Kwon, G., Park, G.Y., Ye, J.C.: Contrastive denoising score for text-guided latent diffusion image editing. arXiv preprint arXiv:2311.18608 (2023)
18. Nichol, A., et al.: Glide: towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
19. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: text-to-3D using 2D diffusion. arXiv preprint arXiv:2209.14988 (2022)
20. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125, vol. 1, no. 2, p. 3 (2022)
21. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2020)
22. Tumanyan, N., Geyer, M., Bagon, S., Dekel, T.: Plug-and-play diffusion features for text-driven image-to-image translation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1921–1930 (2023)
23. Wallace, B., Gokul, A., Naik, N.: Edict: exact diffusion inversion via coupled transformations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22532–22541 (2023)
24. Wang, Q., Zhang, B., Birsak, M., Wonka, P.: MDP: a generalized framework for text-guided image editing by manipulating the diffusion path. arXiv preprint arXiv:2303.16765 (2023)
25. Xie, S., Zhang, Z., Lin, Z., Hinz, T., Zhang, K.: Smartbrush: text and shape guided object inpainting with diffusion model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22428–22437 (2023)
26. Zhao, J., Zheng, H., Wang, C., Lan, L., Huang, W., Yang, W.: Null-text guidance in diffusion models is secretly a cartoon-style creator. In: Proceedings of the 31st ACM International Conference on Multimedia, pp. 5143–5152 (2023)

Texture Spectral Decorrelation Criteria

Michal Haindl(✉) and Michal Havlíček

Institute of Information Theory and Automation, Czech Academy of Sciences Pod Vodarenskou vezi 4, 182 08 Prague, Czechia
{haindl,havlimi2}@utia.cas.cz

Abstract. We introduce texture spectral criteria, which allow us to predict whether simplified spectrally factorized random field-based texture models, a set of two-dimensional models, can faithfully replicate texture spectra compared to their fully spectrally correlated 3D counterparts. These probabilistic models incorporate essential two- or three-dimensional building factors for modeling the seven-dimensional Bidirectional Texture Function (BTF), the most advanced representation in real-world material visual properties modeling. While these models seamlessly approximate original measured massive data and extend them to arbitrary sizes or simulate unmeasured textures, evaluating typically involves time-consuming synthesis and psycho-physical evaluation. The proposed criteria provide an alternative approach, enabling us to bypass the spectral quality evaluation step.

Keywords: Texture spectral quality comparison · Texture modeling · Texture synthesis · Bidirectional texture function

1 Introduction

Photorealism in virtual scenes necessitates meticulously applying physically correct textures that accurately depict natural material surfaces, ensuring a seamless fusion of visual elements. Achieving lifelike material appearances relies on visual textures, conceptualized as manifestations of a mathematically defined random field (RF) possessing spatially uniform attributes. This texture model manifests as a discrete RF, comprising random vectors predominantly arranged on a rectangular lattice grid. The vector space's dimensionality corresponds to the texture's spectral planes, delineating the visual representation's richness and complexity.

Real-world materials exhibit intricate physical characteristics, with their micro-structures influencing reflectance and overall visual presentation. While imperceptible to the naked eye, these micro-structures significantly impact how materials reflect light under various viewing and illumination conditions. The most advanced representation currently available for quantifying and modeling these complexities is the Bidirectional Texture Function (BTF), introduced in [4]. The BTF is a seven-dimensional function encompassing four

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 324–333, 2025.
https://doi.org/10.1007/978-3-031-78172-8_21

parameters related to illumination and viewing angles: azimuthal and elevation angles, one parameter indexing spectral channels, and two parameters representing planar coordinates. This comprehensive model accurately preserves most visual effects inherent to natural materials, including self-occlusions, self-shadowing, inter-reflections, and sub-surface scattering, ensuring a faithful representation of real-world surfaces. Numerous applications (medical, automotive, airspace, cultural heritage preservation, safety, architecture, interior design, entertainment industry, movies, computer games, advertising, material recognition) [3,7,9,10,15,16,23–26] require to analyze or visualize real-world material visual properties and thus can profit from BTF models. Moreover, psychophysical studies of these data [6] have shown that analyzing different BTF samples can help understand human perception of real-world materials.

BTF is represented by thousands of given material surface images taken in different combinations of light sources and observation positions during measurement, which can reach up to several terabytes [13] even for a limited number of combinations of illumination and viewing angles and small planar size of the measured material. Usually several square centimeters [12]. These restrictions exclude the direct use of measured BTF in applications, and accordingly, some compression and enlargement using mathematical BTF models are necessary. BTF models offer extreme data compression without seams but may compromise visual quality.

Such quality compromise is hard to express as fully automatic texture quality assessment and mutual similarity evaluation of two or more of them present a significant but complex problem that needs to be solved. Validation of the state-of-the-art texture fidelity criteria [8] based on the online benchmark[1] demonstrated that none of the already published criteria is trustworthy. Psycho-physical evaluations are the only trustworthy alternative, but they are highly impractical, expensive, and generally demanding.

In this paper, we introduce straightforward texture spectral criteria that enable us to assess whether simplified spectrally factorized random field-based texture models, which require one-third of estimated parameters compared with the three-dimensional models, can faithfully replicate texture spectra compared to their fully spectrally correlated counterparts. Our proposed criteria provide a novel approach for determining when monospectral decorrelated channels can be effectively modeled using a set of simpler 2D random fields and when more complex, fully spectrally correlated 3D models are necessary.

2 BTF Random Field Models

The size of BTF data prohibits its direct integration into graphic applications, necessitating compression for practical usage. Additionally, BTF data is typically acquired under a limited set of illumination and viewing conditions and is measured in small planar sizes, mandating reconstruction and enlargement of

[1] http://tfa.utia.cas.cz.

the BTF space for real-world application. In addition to probabilistic BTF models, there is an alternative approach to approximate BTF data using pixel-wise generalizations of existing BRDF models, known as SVBRDF. However, this method comes with trade-offs, as it sacrifices visual quality by omitting critical features such as self-occlusions, self-shadowing, inter-reflections, and sub-surface scattering and cannot achieve the same level of compression efficiency as BTF.

Modeling BTF based on probabilistic models necessitates the utilization of multi-dimensional models. However, such models are rare and encounter various unresolved theoretical challenges, as noted in [12]. One potential workaround involves spectrally and spatially factorizing the BTF space, enabling its representation through a series of lower-dimensional models. Unfortunately, accurate data are correlated and can only be spectrally factorized approximately, which can lead to a loss of spectral information.

The texture pixels are defined as intensity values (2D) or intensity vectors (3D) on multiple finite $M \times N$ 2D lattice I. The 3D multiindex is $r = \{r_1, r_2, r_3\}$ with spatial (r_1, r_2) and spectral (r_3) indices $r \in I$. Markovian neighboring lattice locations are the set of relative lattice locations called contextual neighborhood (CN) I_r. Choosing an appropriate Complex Network (CN) significantly impacts the overall model performance. A CN with too few elements fails to capture all texture details effectively. Conversely, including unnecessary elements increases computational overhead and can degrade the model's performance by introducing additional noise.

2.1 Spectral Factorization

Modeling static multispectral texture images or single BTF space measurements, where d represents the number of spectral images (e.g., $(d = 3)$ for color), typically requires three-dimensional models. However, this dimensionality can be reduced to two by applying spectral factorization to the real image before modeling and inverse spectral factorization to the synthetic result using the Karhunen-Loève transformation (KLT) within the original centered spectral data space [12].

Unfortunately, the real spectral data space can only be decorrelated approximately. The approximation error is directly related to the extent of the color space in the modeled texture. While full 3D models allow unrestricted spatial-spectral correlation modeling, 2D models introduce some spectral errors, especially for highly colorful textures. Their primary drawback lies in the large number of parameters that need to be estimated and the necessity to estimate all these parameters simultaneously.

2.2 3/2D Auto-Regressive and Moving Average Models

The Causal Auto-Regressive (2/3DCAR), Multi-Spectral Simultaneous Auto-Regressive (2/3DMSAR) [14], Moving Average (2/3DMA) [18] (2D after spectral decorrelation) RF is a collection of random variables with a joint probability

density on the set of all possible realizations Y of the $M \times N$ lattice I which can be written in the following regression equation form [11]:

$$Y_r = \sum_{s \in I_r^a} A_s Y_{r-s} + \sum_{t \in I_r^b} B_t E_{r-t} = \Theta X_r + \Xi Z_r \ , \qquad \forall r \in I \tag{1}$$

where A_s, B_s are matrices (4) and the zero mean white Gaussian noise vector E_r has uncorrelated components with data indexed from unilateral or causal index set I_r^a but noise vector components can be mutually correlated with a constant covariance matrix Σ, I_r^b includes the r index (contrary to I_r^a), and

$$\Theta = [A_1, \ldots, A_\eta] \qquad X_r = [Y_{r-s}^T : \ \forall s \in I_r^a] \ , \tag{2}$$

$$\Xi = [B_1, \ldots, B_\iota] \qquad Z_r = [Er - t^T : \ \forall t \in I_r^b] \ , \tag{3}$$

Θ, Ξ are $d \times d\eta$ and $d \times d\iota$ parameter matrices, X_r is a vector of contextual neighbors, Z_r is a vector of white noise subvectors.

$$A_{s_1,s_2} = \begin{pmatrix} a_{1,1}^{s_1,s_2}, \ldots, a_{1,d}^{s_1,s_2} \\ \vdots, \ddots, \vdots \\ a_{d,1}^{s_1,s_2}, \ldots, a_{d,d}^{s_1,s_2} \end{pmatrix} \qquad B_{s_1,s_2} = \begin{pmatrix} b_{1,1}^{s_1,s_2}, \ldots, b_{1,d}^{s_1,s_2} \\ \vdots, \ddots, \vdots \\ b_{d,1}^{s_1,s_2}, \ldots, b_{d,d}^{s_1,s_2} \end{pmatrix} \tag{4}$$

are $d \times d$ parameter matrices. The 2DCAR differs from the 3DCAR with all diagonal parametric matrices $A_{s_1,s_2} \ \ \forall s \in I_r^c$ and diagonal covariance matrix Σ, i.e. single spectral component 2D models are mutually independent. All CAR model statistics can be solved analytically, we use the Bayesian estimates (for details see [11]).

Single models properties:

2DCAR Y spectrally decorrelated, Ξ unity $d \times d$ matrix, $Z_r = E_r$, I_r^a causal or unilateral, $I_r^b = \{r\}$,
3DCAR Y spectrally correlated, Ξ unity $d \times d$ matrix, $Z_r = E_r$, I_r^a causal or unilateral, $I_r^b = \{r\}$,
2DMSAR Y spectrally decorrelated, Ξ unity $d \times d$ matrix, $Z_r = E_r$, I_r^a non-causal, $I_r^b = \{r\}$, double-toroidal boundary condition ,
3DMSAR Y spectrally correlated, Ξ unity $d \times d$ matrix, $Z_r = E_r$, I_r^a non-causal, $I_r^b = \{r\}$, double-toroidal boundary condition ,
2DMA Y spectrally decorrelated, Θ zero matrix ,
3DMA Y spectrally correlated, Θ zero matrix .

The Bayesian parameter estimations of the 2/3CAR model with the normal-Wishart parameter prior, which maximizes the posterior density, allow to analytically compute synthetic model output in the form of conditional mean values estimate [11]:

$$\hat{Y}_r = \hat{\Theta} X_r \ . \tag{5}$$

The MSAR parameters under the assumed double-toroidal boundary condition are estimated using the least square method and synthesized using the discrete

fast Fourier transformation [1,14]. The 2DMA parameters are estimated using a method [22] similar to the one-dimensional (1D) Random Decrement Technique [2]. The 3DMA model estimation is estimated as proposed in [18]. 2D/3DMA texture models can generate synthetic arbitrary-size textures directly from their model equations. All 2D models' synthetic textures have to be subsequently transformed using the inverse Karhunen–Loeve transformation to their spectrally correlated final version.

2.3 Multi-spectral Markov and Pseudo-markov Random Field Models

The BTF Multi-Spectral Markov Random Field (BTF-MMRF) model and the BTF Pseudo Markov Random Field (BTF-PMRF) are based on (3DMMRF/3DPMRF) factor texture models [1]. A multi-spectral texture can be considered Markovian with respect to I_r^{ij} if it has following property [1]:

$$\begin{aligned} p(Y_{r,i} \,|\, Y_{s,j},\, \forall j \in \{1,\dots,d\} : j \neq i, \forall s \in I_r^{ij} : s \neq r) \\ = p(Y_{r,i} \,|\, Y_{s,j},\, \forall j \in \{1,\dots,d\}, \forall s \in I_r^{ij}) \ . \end{aligned} \tag{6}$$

As the conditional distributions of $Y_{r,i}$ given $\{Y_{s,j},\, \forall j \in \{1,\dots,d\} : j \neq i, \forall s \in I_r^{ij} : s \neq r\}$ and $Y_{r,i}$ given $\{Y_{s,j},\, \forall j \in \{1,\dots,d\}, \forall s \in I_r^{ij}\}$ are the same, the best linear estimator of Y can be written:

$$Y_r = \sum_{s \in I_r} A_s Y_{r-s} + \Psi_r = \Theta X_r + \Psi_r \ , \qquad \forall r \in I \tag{7}$$

where I_r is a non-causal neighborhood and Ψ_r is the correlated driving noise. The correlation structure of the stationary noise $\epsilon_{r,i}$ is [1]:

$$\Psi_{ij}^s = E\{\epsilon_{r,i}\epsilon_{r\oplus s,j}\} = \begin{cases} -a_{s,i,j}\sigma_j & s \in I_r^{ij} \quad \text{MMRF only} \ , \\ -a_{s,i,j}\sqrt{\sigma_i\sigma_j} & s \in I_r^{ij} \quad \text{PMRF only} \ , \\ \sigma_j & s = 0,\ i = j \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{8}$$

The parameters A_s are estimated using the Minimum Mean Square Error (MMSE) estimate [1]. Because the correlation functions have the symmetry property $\Psi_{ij}^s = \Psi_{ji}^{-s}$, there is an implicit requirement that I_r^{ij} and the associated coefficients are symmetric, i.e., $s \in I_r^{ij} \iff -s \in I_{-r}^{ji}$ and $\sigma_j \Psi_{ij}^s = \sigma_i \Psi_{ji}^{-s}$. The LS estimates are inherently nonlinear, and it is necessary to solve for all model parameters simultaneously, e.g., using the iterative approach [1]. The main difference between the MMRF model and the PMRF model and the significant advantage of the PMRF model is the fact that the estimate $\widehat{\sigma}$ is linear and independent of the estimate $\widehat{\Theta}$. Therefore, the model parameter estimation does not require an iterative process, unlike the case of the MMRF model parameter estimation, which reduces the computational burden. The algorithm for the synthesis of the MMRF/PMRF models is similar and identical to that one for the MSAR model, except for the calculations in its third step (see details and the stability condition in [1,17]).

3 Color Quality Criteria

3.1 Spectral Decorrelation Criteria

The spectral decorrelation criteria vary depending on whether they are applied to a single spatial resolution (as in Eq. (9)) or multiple spatial resolutions (as in Eqs. (10) and (11)).

$$\kappa(c_{\max}) = \frac{1}{c_{\max}^n} \det(\Sigma) \quad , \tag{9}$$

where Σ is a $n \times n$ material texture spectral covariance matrix, $c_{\max}$ is the maximal possible spectral value per channel. If the criterion $\kappa(c_{\max}) \leq 3$, we can replace a 3D random field model with its Karhunen-Loeve decorrelated version and model each decorrelated single spectral band with a 2D random field model without significant MEMD error (12), i.e., with negligible color loss.

Gaussian Pyramid. Spectral decorrelation criterion for Gaussian pyramid factorized image:

$$\kappa(c_{\max}, l) = \sum_{i=1}^{l} \frac{1}{c_{\max}^n} \det(\Sigma_i) \frac{s^{l-1} - s^l}{1 - s^l} \frac{1}{s^{i-1}} \quad , \tag{10}$$

where s is a sampling step ($s = 2$), l is the number of pyramid levels.

Gaussian-Laplacian Pyramid

$$\kappa(c_{\max}, l_{\max}) = \frac{1}{c_{\max}^n} \frac{s^{l_{\max}}}{1 + s^{l_{\max}}} \left(\det(\Sigma_1) + \det(\Sigma_{l_{\max}}) \frac{1}{s^{l_{\max}}} \right) \quad , \tag{11}$$

where s is a sampling step ($s = 2$), $l_{\max}$ is the last (Gaussian) pyramid level.

3.2 Color Composition Comparison

The Mean Exhaustive Minimum Distance (MEMD) [20] is used to compare the spectral composition of two textures and the cardinalities of the same colors but ignoring the locations of individual pixels. The comparison is performed by individually taking pixels from the first image and searching for the most similar, i.e., the closest in certain vector metric sense, to the ones in the second image. The pixel from the second image identified as the most similar is removed from the stack representing the second image and the local spectral error is recorded. The MEMD criterion is as follows:

$$\zeta(A, B) = \frac{1}{M} \sum_{(r_1, r_2) \in \langle A \rangle} \min_{(\acute{r}_1, \acute{r}_2) \in N} \left\{ \rho\left(Y^A_{r_1, r_2, \bullet}, Y^B_{\acute{r}_1, \acute{r}_2, \bullet} \right) \right\} \quad , \tag{12}$$

where $M = \min\{\sharp\{A\}, \sharp\{B\}\}$, $\sharp\{A\}$ is the number of pixels in A and similarly for $\sharp\{B\}$, $\min\{\emptyset\} = 0$, (r_1, r_2) denotes the location in A, $\langle A \rangle$ represents the set of all pixel indices of A, $(\acute{r}_1, \acute{r}_2)$ is the location in B, N is the set of unprocessed

Table 1. MEMD error values between the measured original O and its synthesized version M corresponding to the best-achieved results for individual models and color2 texture modifications 0-9 in experiment 2 and their $\kappa(256), \kappa(256, 2)$ criteria.

Model	0 $\zeta(O,M)$	1 $\zeta(O,M)$	2 $\zeta(O,M)$	3 $\zeta(O,M)$	4 $\zeta(O,M)$	5 $\zeta(O,M)$	6 $\zeta(O,M)$	7 $\zeta(O,M)$	8 $\zeta(O,M)$	9 $\zeta(O,M)$
2D CAR	15,75	16,5	15,72	15,38	13,37	13,39	11,57	10,03	7,66	3,87
2D MA	44,04	38,44	35,24	30,82	30,48	26,34	24,68	19,41	12,87	5,32
2D MSAR	4,65	4,29	4,22	4,34	5,21	7,35	9,28	9,65	8,45	5,02
2DMMRF	7,97	6,52	5,97	5,58	5,74	7,7	9,4	9,76	8,47	5,05
2D PMRF	8,03	6,54	5,95	5,55	5,73	7,69	9,4	9,75	8,47	5,05
3D CAR	5,74	4,67	4,27	4,48	4,93	6,89	8,68	9,04	7,87	4,67
3D MA	6	5,79	6,41	5	7,19	8,32	8,94	7,35	5,31	2,5
3D MSAR	5,4	5,44	5,52	6,27	6,66	8,28	9,7	9,86	8,57	5,11
3DMMRF	6,98	6,11	5,04	4,75	5,25	7,24	9,13	9,59	8,52	5,12
3D PMRF	7,44	6,11	5,03	4,75	5,26	7,25	9,14	9,6	8,52	5,12
$\kappa(\cdot)$	644	327	157	70	29	10	4	1,2	0,34	0,09
$\kappa(\cdot, 2)$	537	273	130	58	24	9	3	1	0,28	0,07

pixel indices of B, ρ is an arbitrary vector metric and $Y^A_{r_1,r_2,\bullet}$ represents the pixel at (r_1, r_2) in A, where $\bullet$ denotes all corresponding spectral indices, similarly for $Y^B_{\acute{r}_1,\acute{r}_2,\bullet}$. The term $\zeta(A, B)$ is evaluated using raster scanning of A. The algorithm stops when all pixels of A are scanned, or N becomes an empty set (see details in [19–21]).

4 Results

In our experiments, we utilized highly colored texture (*color2*) in two different scenarios, along with the BTF MAM2014 Dataset[2] [5] but details of these results, except two materials Fig. 2, are not shown here (16 BTF materials, each consisting of 6,561 images). **Experiment 1:** We gradually reduced the number of colors by replacing a randomly selected color with another randomly chosen color in a single degradation step. **Experiment 2** (Fig. 1): We computed the mean spectral intensity for each color. Subsequently, we replaced that color with a weighted combination of the original color and the mean value. The weights progressively increased toward achieving a monospectral texture. Average criterion (9) values over all 6,561 samples per material are $\bar{\kappa}(mica) = 2,9 \cdot 10^8$, $\bar{\kappa}(basketweave) = 1,1$. Highly reflective mica requires 3D models for every combination of elevation angles θ_i, θ_v, while the basketweave material has only one reflective peak for high-elevation $\theta_i = \theta_v = 75°$ angles (Fig. 2-rightmost), which can be neglected, and thus, this material can be completely modeled with a simpler set of 2D models.

The minor spectral decorrelation criterion across multiscale models (as indicated in Table 1 similar table for Experiment 1 is not shown here) suggests a

[2] http://btf.utia.cas.cz.

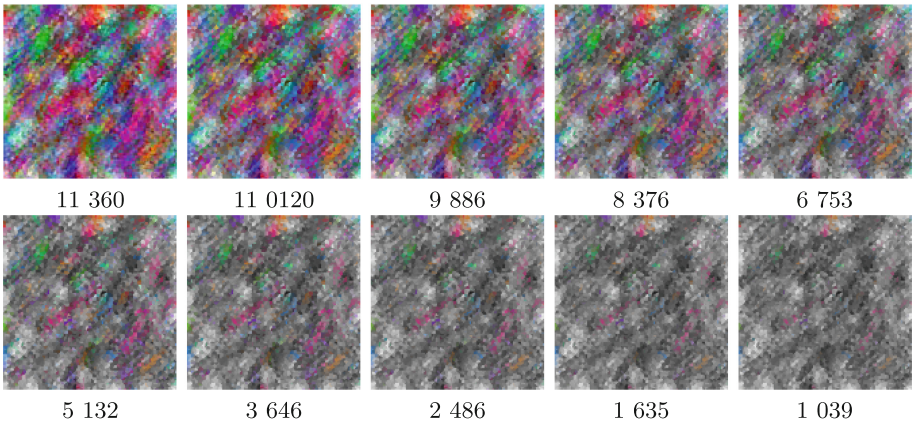


Fig. 1. Color texture with subsequent color reduction and the corresponding of the number of different colors (rightwards) in the experiment 2.

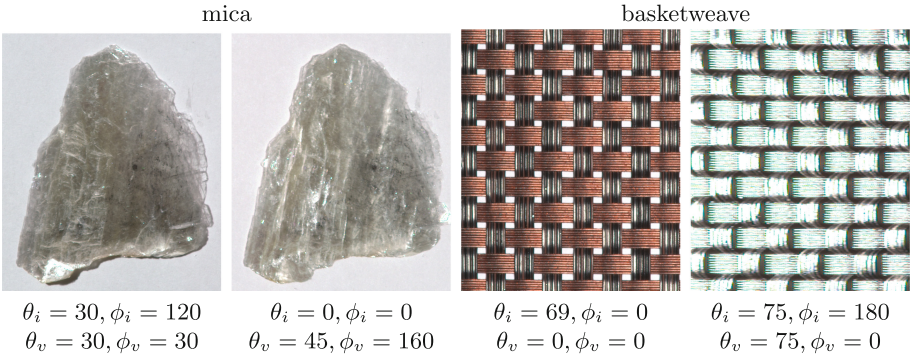


Fig. 2. BTF mica and basketweave texture samples.

slightly broader applicability of spectrally decorated multiscale 2D-factor models compared to their single-scale counterparts. In terms of color error $\zeta(\cdot)$, both experiments and single/multi-scale 3D models exhibit the following results: 3DCAR has the smallest median color error $\bar{\zeta}(256) = 5,34$. 3DMMRF and 3DPMRF perform the worst. 2D models demonstrate larger spectral $\zeta(\cdot)$ error due to their application on decorrelated data. The 2DMSAR has the best spectral modeling performance, while the far the worst is the 2DMA model. Additionally, the condition $\kappa(\cdot) < 1$ also suggests a small spectral $\zeta(\cdot)$ error less than 5% for all models.

5 Conclusions

We propose criteria that allow us to predict when a vast BTF data space can be effectively modeled using simpler 2D random field models (one-third of 3D model

parameters) instead of more complex 3D random field models, all while maintaining spectral quality. This prediction helps avoid computationally demanding experiments with both types of models: spectrally decorrelated 2D and fully correlated 3D RF. Additionally, it addresses the challenge posed by the lack of reliable texture quality criteria. These mathematical models offer attractive and practical alternatives for modeling, providing extreme data compression ($\approx 1 : 10^6$ for our BTF measurements). Instead of storing the original acquired data, only a small number of parameters need to be retained. To assess the quality of synthesized data, we compare it with the original data using the texture spectral composition criterion denoted as $\zeta(\cdot)$. The 3DCAR factor model stands out among the models, achieving robustness and superior spectral quality performance.

References

1. Bennett, J., Khotanzad, A.: Multispectral random field models for synthesis and analysis of color images. IEEE Trans. Pattern Anal. Mach. Intell. **20**(3), 327–332 (1998)
2. Cole Jr, H.A.: On-line failure detection and damping measurement of aerospace structures by random decrement signatures. Technical report, TMX-62.041, NASA (1973)
3. Cula, O., Dana, K., Murphy, F., Rao, B.: Bidirectional imaging and modeling of skin texture. IEEE Trans. Biomed. Eng. **51**(12), 2148–2159 (2004)
4. Dana, K.J., Nayar, S.K., van Ginneken, B., Koenderink, J.J.: Reflectance and texture of real-world surfaces. In: CVPR, pp. 151–157. IEEE Computer Society (1997)
5. Filip, J., Kolafová, M., Havlíček, M., Vávra, R., Haindl, M., Rushmeier, H.: Evaluating physical and rendered material appearance. Vis. Comput. 805–816 (2018). https://doi.org/10.1007/s00371-018-1545-3
6. Filip, J., Chantler, M.J., Green, P.R., Haindl, M.: A psychophysically validated metric for bidirectional texture data reduction. ACM Trans. Graph. (TOG) **27**(5), 138:1–138:11 (2008). https://doi.org/10.1145/1457515.1409091
7. Grim, J., Somol, P., Haindl, M., Daneš, J.: Computer-aided evaluation of screening mammograms based on local texture models. IEEE Trans. Image Process. **18**(4), 765–773 (2009). https://doi.org/10.1109/TIP.2008.2011168
8. Haindl, M., Kudělka, M.: Texture fidelity benchmark. In: 2014 International Workshop on Computational Intelligence for Multimedia Understanding (IWCIM), pp. 1 – 5. IEEE Computer Society CPS, Los Alamitos (2014). https://doi.org/10.1109/IWCIM.2014.7008812
9. Haindl, M., Mikeš, S.: Unsupervised mammograms segmentation. In: Lovell, B., Laurendeau, D., Duin, R. (eds.) Proceedings of the 19th International Conference on Pattern Recognition, ICPR 2008, pp. 1–4. IEEE Computer Society, Los Alamitos (2008). http://doi.ieeecomputersociety.org/10.1109/ICPR.2008.4761113
10. Haindl, M., Vácha, P.: Illumination invariant texture retrieval. In: Tang, Y., Wang, S., Yeung, D., Yan, H., Lorette, G. (eds.) Proceedings of the 18th International Conference on Pattern Recognition, ICPR 2006, vol. III, pp. 276–279. IEEE Computer Society, Los Alamitos (2006). http://doi.ieeecomputersociety.org/10.1109/ICPR.2006.678

11. Haindl, M.: Bidirectional texture function modeling. In: Chen, K., Schönlieb, C.B., Tai, X.C., Younes, L. (eds.) Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging, pp. 1023–1064. Springer, Cham (2023). https://doi.org/10.1007/978-3-030-98661-2_103
12. Haindl, M., Filip, J.: Visual Texture. Advances in Computer Vision and Pattern Recognition, Springer, London (2013). https://doi.org/10.1007/978-1-4471-4902-6
13. Haindl, M., Filip, J., Vávra, R.: Digital material appearance: the curse of terabytes. ERCIM News (90), 49–50 (2012). http://ercim-news.ercim.eu/en90/ri/digital-material-appearance-the-curse-of-tera-bytes
14. Haindl, M., Havlíček, M.: Bidirectional texture function simultaneous autoregressive model. In: Salerno, E., Çetin, A.E., Salvetti, O. (eds.) MUSCLE 2011. LNCS, vol. 7252, pp. 149–159. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32436-9_13
15. Haindl, M., Mikeš, S., Scarpa, G.: Unsupervised detection of mammogram regions of interest. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007. LNCS (LNAI), vol. 4694, pp. 33–40. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74829-8_5
16. Hasegawa, T., Tsumura, N., Nakaguchi, T., Iino, K.: Photometric approach to surface reconstruction of artist paintings. J. Electron. Imaging **20**, 013006 (2011)
17. Havlíček, M.: Bidirectional texture function three dimensional pseudo gaussian Markov random field model. In: Doktorandské dny ČVUT, pp. 53–62. ČVUT (2012)
18. Havlíček, M.: Extended bidirectional texture function moving average model. In: Doktorandské dny ČVUT, pp. 37–43. ČVUT (2015)
19. Havlíček, M., Haindl, M.: Optimized texture spectral similarity criteria. In: Wojtkiewicz, K., Treur, J., Pimenidis, E., Maleszka, M. (eds.) ICCCI 2021. CCIS, vol. 1463, pp. 644–655. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88113-9_52
20. Havlíček, M., Haindl, M.: Texture spectral similarity criteria. IET Image Process. **13**(11), 1998–2007 (2019). https://doi.org/10.1049/iet-ipr.2019.0250. https://digital-library.theiet.org/content/journals/10.1049/iet-ipr.2019.0250
21. Havlíček, M., Haindl, M.: Texture spectral similarity criteria comparison. Comput. Sci. Res. Notes **3301**, 100–106 (2023). https://doi.org/10.24132/CSRN.3301.13. http://wscg.zcu.cz/DL/wscg_DL.htm
22. Li, X., Cadzow, J., Wilkes, D., Peters, R., II Bodruzzaman, M.: An efficient two dimensional moving average model for texture analysis and synthesis. In: Proceedings IEEE Southeastcon 1992, vol. 1, pp. 392–395. IEEE (1992)
23. Malzbender, T., Gelb, D., Wolters, H.: Polynomial texture maps. In: Eurographics 2001, pp. 519–528. ACM Press (2001)
24. Scarpa, G., Gaetano, R., Haindl, M., Zerubia, J.: Hierarchical multiple Markov chain model for unsupervised texture segmentation. IEEE Trans. Image Process. **18**(8), 1830–1843 (2009). https://doi.org/10.1109/TIP.2009.2020534
25. Vácha, P., Haindl, M.: Illumination invariant and rotational insensitive textural representation. In: Bayoumi, M. (ed.) IEEE 16th International Conference on Image Processing - ICIP 2009, pp. 1333–1336. IEEE (2009)
26. Vácha, P., Haindl, M.: Content-based tile retrieval system. In: Hancock, E.R., Wilson, R.C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR /SPR 2010. LNCS, vol. 6218, pp. 434–443. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14980-1_42

A Low Rank Gaussian Mixture Latent Model for Face Generation

Benjamin Samuth(✉), Julien Rabin, Fréderic Jurie, and David Tschumperlé

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
{samuth.users.greyc.fr,benjamin.samuth,rabin.users.greyc.fr, julien.rabin,jurie.users.greyc.fr,frederic.jurie, tschumperle.users.greyc.fr,david.tschumperle}@unicaen.fr

Abstract. Generative modeling of natural images has seen significant progress, but large-scale foundation models raise concerns about environmental impact, privacy, and biases. This motivates investigating more efficient and interpretable generative models. This work proposes a simple latent parametric generative model focused on realistic face generation, a domain that has seen success with neural networks. The model uses a low-dimensional latent representation from a pre-trained autoencoder, and proceeds in two stages: (1) modeling the latent distribution as a mixture of multivariate Gaussians trained on a limited dataset, and (2) generating low-rank random codes from this prior and remapping them using nearest nneighbor matching. Comparative experiments demonstrate the advantages of the proposed approach.

Keywords: Generative Modeling · Auto-encoder · Face Generation

1 Introduction

The field of deep learning-based generative modeling of natural images has witnessed substantial advancements in recent years. Popular methods now achieve hyper-realistic image synthesis by combining visual and natural language cues, enabling their use for downstream tasks such as image editing.

These successful generative models rely on a few powerful techniques. The first is dimensionality reduction. Self-supervised representation learning, primarily built on autoencoding neural networks, allows for the compact representation of images in a low-dimensional embedding space. Careful architectural design enables an interesting trade-off between the fidelity of image reconstruction and the compression rate [7].

While dimensionality reduction is mandatory for designing realistic generative models [14], latent representations offer two main benefits for generative

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_22.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 334–349, 2025.
https://doi.org/10.1007/978-3-031-78172-8_22

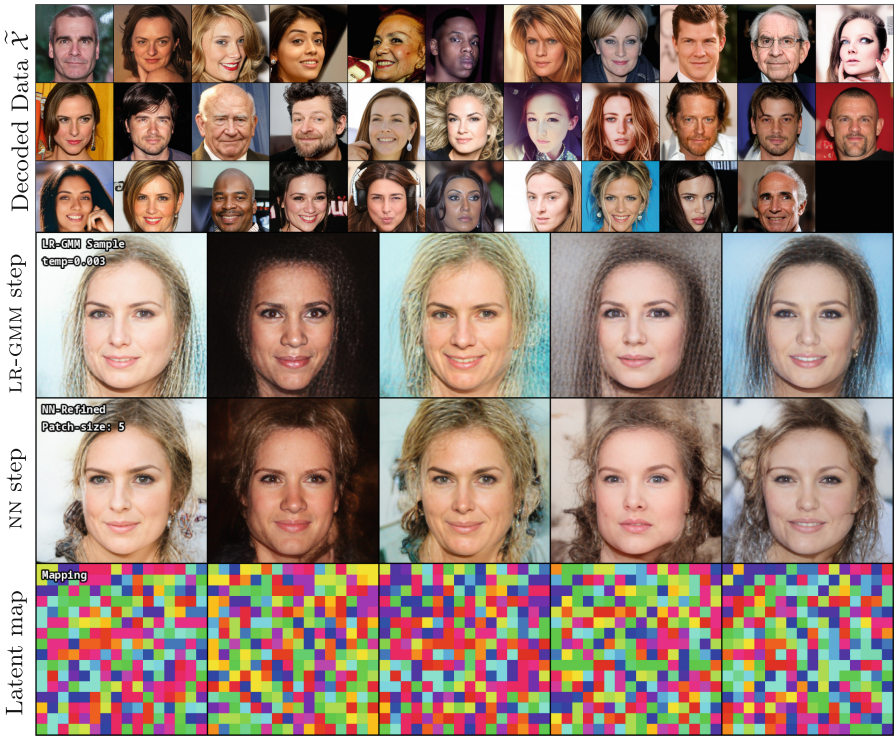


Fig. 1. Generated samples from a limited dataset of faces with LLR-GMM-NN, the proposed method described in Fig. 2. The LR-GMM step consists in estimating a low-rank latent gaussian mixture model fitted on a limited dataset $\mathcal{X}$ (here with only $N = 32$ training images $\widetilde{\mathcal{X}}$ showed after decoding by an auto-encoder). The dimensionality is controlled by a temperature parameter (here 0.3%). An additional NN step adds missing visual details by matching local features from the training set, captured by latent patches (with a size of 5). The latent map at the bottom indicates in color code the corresponding training sample used for each pixel.

modeling. First, pre-training a decoder helps reduce the complexity of generative models, improving training computational time and data requirements, as demonstrated in various cases [7,28,29]. Additionally, the encoder helps extract perceptually meaningful features that achieve better results than the original color space for tasks like image comparison [8,16,46] and interpolation [21].

The most successful approaches in recent years have utilized or combined one of the following key techniques: i) Auto-regressive models, which sequentially predict masked pixels, either in the color space (as in PixelCNN [26]) or in the latent space (as in VQ-VAE [28]), often employing decoder-type transformer architectures [40] for this sequential prediction; ii) Adversarial training, where feed-forward neural network architectures trained using generative adversarial networks (GANs) [10,17] can produce highly realistic images; and iii) Diffusion

models, iterative, denoising models that can generate realistic images, either in the color space [14], in the image latent representation [29], or in a multimodal representation [27].

The common thread among these successful techniques is their reliance on massive neural network architectures, often with billions of parameters, to attain levels of realism that can deceive human perception. As a result, such large models require enormous amounts of training data and extensive computational resources during both training and inference.

However, this level of achievement comes with some remaining challenges. First, the use of over-parameterized deep neural networks has raised legitimate concerns about data privacy [44], which have also been shown to occur in generative modeling, particularly in GANs [12,42]. Some mechanisms, such as differential privacy [43], have been derived to mitigate these issues. Additionally, other generative models based on transformers have been shown to be able to memorize some training samples [7,25], a concern that has also been scrutinized for diffusion models [4,38].

Besides, the proliferation of such data-intensive and computationally demanding models raises numerous ethical and social concerns [20], including their environmental impact [3], legal implications concerning training with copyrighted content [36], and the presence of biases, illegal material [39], or corrupted samples [37] within the training dataset. These concerns underline the need to explore alternative approaches that are more resource-efficient, transparent, and less susceptible to ethical dilemmas.

In this context, this work aims to show that a simple latent generative model, both in terms of parameters and computing power, can achieve high performance in comparison to large state-of-the-art auto-regressive and diffusion models. Additionally, we further demonstrate that it allows using less data and yields a more explainable model. Lastly, we show that such a model can address other downstream tasks, such as image editing.

The proposed approach, illustrated in Fig. 1, aims to generate plausible latent representations of images from a limited dataset of portrait examples, which can then be used to reconstruct the corresponding high-resolution images using a pre-trained decoder. The generative process is feed-forward and consists of two steps: i) Modeling the latent distribution of the limited dataset as a Gaussian mixture model (GMM). Combined with low-rank approximation (LR), this relatively simple model requires only a small number of parameters, thanks to the compact latent representation of the images, which allows for a limited training dataset. Yet, this approach still enables the imposition of long-range correlations in the final high-resolution images. ii) During inference, sparse latent codes are randomly generated to enforce the sharpness of the generated images. The second stage then consists of injecting missing details (high-frequency patterns) by iteratively projecting the latent pixels onto similar training samples, based on local comparisons (nearest-neighbor step or NN).

The rest of the paper is organized as follows: The next section (Sect. 2) describes related work in latent generative modeling and exposes their limita-

tions. Section 3 introduces the proposed two-step generative model, consisting of a combination of a latent low-rank Gaussian mixture (Sect. 3.2) and a nearest-neighbor projection (Sect. 3.3). The experimental section (Sect. 4) shows qualitative and quantitative results for face generation, demonstrating the advantages of the proposed method compared to several competitive methods from the literature. Finally, we conclude in Sect. 5 by discussing some perspectives and future lines of work.

2 Related Work

State-of-the-Art in Generative Modeling of Face. Our work focus on face generation which has been a long standing problem in generative image modeling. One of the first milestone in realistic portrait generation has been GAN-based approaches, starting with PG-GAN [18] which introduced a progressive adversarial training on a curated dataset of 30k registered HD images (CelebA-HQ). With a different architecture allowing for explicit image stylization, Style-GAN [19] improved upon this work with 70k high-quality images under permissive licenses (FFHQ). These generative models, mapping a random latent code to a realistic image, can be used as priors for image editing by implicitly exploiting the latent pre-image. However, such plug-and-play techniques requires gradient-based optimization or the posterior training of a dedicated encoder.

As mentioned in the introduction, a popular technique to reduce data dimension and model size and training time is the use of a latent generative representation [7,28,29]. In a first stage, an auto-encoder is trained on the target dataset to learn a compact latent representation (*i.e.* with a very small spatial resolution) *via* an encoder that is perceptually well reconstructed by a decoder. To achieve this result, variational auto-encoders (VAE [21]) have been widely used [28] and are often combined with some other techniques, such as GANs [7].

In a second stage, a latent generative model is trained on the same dataset. After training, the parametric model is used to generate a random but plausible latent code that is decoded to synthesize a realistic image.

Auto-regressive models for images has been popularized with PixelCNN [26] and combined with quantized latent representation in VQ-VAE [28] to produce realistic images. With the advent of transformers and attention-based architectures [40], token-based decoder transformers has been successfully combined with latent representation [7]. A limitation of such methods comes from their sequential nature, requiring a lot of computations and a large number of parameters to impose long range correlation in long sequence.

More recently, diffusion and score-based models have been popularized since the seminal work of [14] and its application for text-to-image generation scaled to large datasets [27,31], in combination of with auto-encoders [29]. While iterative in nature by requiring thousand of denoising steps, the distillation of such stochastic models has met some success recently to accelerate the generation by training a single step generative network [32]. Yet, these models are still large in nature. The sheer amount of parameters contained in a state-of-the-art model requires several weeks of training with relatively massive computing power.

Besides these computational requirements and the related environmental considerations, a growing number of ethical issues is related to the use of these large models as black-boxes (reproduction of copyrighted material, reproduction of bias from the training data, privacy of the training data, etc.). Addressing these concerns demand more explainable and transparent generative image models. In this work, we investigate this line of research in the context of frugal models.

Generative Model with Limited Parameters and Data. Child et al. [5] showed that simple multi-scale yet very deep VAE could compete with much larger auto-regressive models. While being able to generate high quality portrait, the proposed model is still limited to small resolution (256 pixels) with hundred of millions of parameters.

One common practice to avoid re-training large models from scratch is to finetune their parameters on another target dataset. In particular, [41] notes its efficiency for generative methods in adversarial networks. A recently more popular approach has been to train a smaller, low-rank, auxiliary model to adapt the parameters to the target distribution [15,30,45]. Yet, the auxiliary model uses the large model even during inference, which still requires huge amount of memory and computations.

Few-shot generation main goal is to be able to generate images from a limited dataset, with very scarce information. Training models becomes in principle more difficult as most architectures are in that case prone to overfitting or mode collapse. [24] proposes an adversarial network adapted to a low amount of data points. They introduce skip-layer excitation modules that weight high-resolution features with the low-resolution ones. This allows them a robust training with lower parameter count. [2] specializes models to the target distribution thanks to a quantization codebook that specifically encodes patches of the limited dataset. They then train an auto-regressive model that can only generate codes from that constrained codebook. Both approaches however require an entire retraining if the limited dataset were to change. Even though considerably lower than large models, the training process can take hours of computing for just a dozen images, for instance.

Last, our work share some similarities with Latent-Patch [33]. In both case, a shallow latent generative model is proposed to train from a limited dataset. However, these generative models are completely different in nature. In Latent-Patch, a multi-scale non-parametric auto-regressive model is used, inspired from patch-based texture synthesis algorithms and the PatchMatch approach introduced for image editing. A limitation of this approach is the lack of long-range correlation by the sequential nature of the synthesis, which can results in global inconsistencies, such as face asymmetries (different eyes' color, earring on only one side, etc.), different hair styles depending on the spatial location, or inconsistent background. In contrast, our model requires a few parameters (around 1M, so 10 to 100 times lower than other methods) to be trained, and only requires two steps that are parallel and thus faster to infer.

3 Latent LR-GMM-NN Generative Model

Overview. Like many aforementioned state-of-the-art approaches, we propose a latent generative model which is built on top of a pre-trained auto-encoder. The next Sect. 3.1 gives more detail on learning such a latent representation on an auxiliary dataset $\mathcal{A}$. The following sections then introduce the different steps of the proposed generative model, coined LLR-GMM-NN in this document, that are summarized in Fig. 2.

As exposed in Sect. 3.2 about learning the latent parametric model, latent representations of the training images of a limited target dataset $\mathcal{X}$ are first collected and compressed using dimensionality reduction operator P_0, by means of a principal component analysis (PCA). A Gaussian Mixture Model (GMM) is then fitted to the latent distribution of the target dataset $\mathcal{X}$ to capture the desired spatial correlation at coarse resolution. The first stage of inference (LR-GMM-step) finally consists in generating a latent code, and combines a Gaussian Mixture Model (GMM) with Low-Rank (LR) samples using sparse operators P_k.

The following NN-step, introduced in Sect. 3.3, stands for Nearest-Neighbor projection, and consists in injecting details in the generated latent sample by matching local features (latent patches) from training examples.

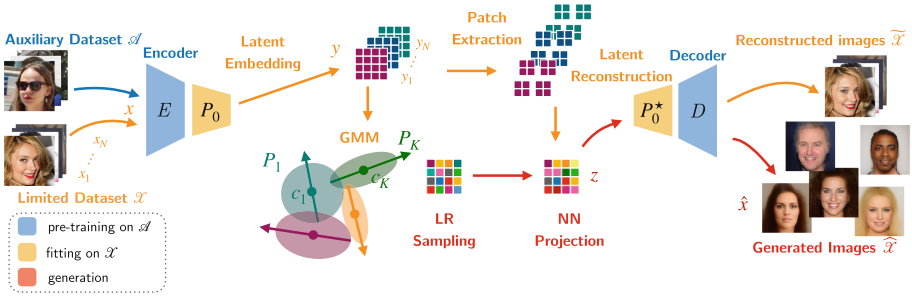


Fig. 2. Overview of the proposed LLR-GMM-NN method. A latent representation is obtained by pre-training an auto-encoder (E and D) on an auxiliary dataset $\mathcal{A}$. The model is fine-tuned on the target distribution of images by training a latent embedding (P_0 and $P_0^\star$) on an limited dataset $\mathcal{X}$ which reduce the dimension of the latent representation. A gaussian mixture model (GMM) is fitted on latent images, from which a low-rank approximation (LR) is used to generate random samples. A local Nearest-Neighbor (NN) projection based on patch-similarity is used to modify synthesized samples before decoding.

3.1 Latent Representation

A generic auxiliary dataset $\mathcal{A} = \{a_j\}_{j=1}^{M}$ is considered to learn an appropriate latent representation of images. We consider color images of size $3 \times H \times W$, where H (height) and W (width) indicates spatial dimension.

Auto-Encoder. To achieve this goal, we follow the paradigm of many state-of-the-art methods by first training an auto-encoder (AE) composed of an embedding deep-network E (which is used to learn an appropriate latent generative model later on), and a decoder D (used to synthesize color images from random latent codes). As this strategy has been proven to be effective to learn various generative models (see *e.g.* VQ-VAE [28], VQ-GAN [7], latent-diffusion [29], Latent-Patch [33]), we assume that we can use a generic on-the-shelf auto-encoder providing a latent representation y of images at coarse spatial information $d_{\text{latent}} = c \times h \times w$ (*i.e.* $h \ll H$). The spatial grid is referred to as $\Omega := \{1, \dots h\} \times \{1, \dots w\}$ in the following.

Finetuning. As proposed in [33], a limited set of N images $\mathcal{X} = \{x_i\}_{i=1}^N$ is used to fine-tune such a generic auto-encoder to the target distribution of images. In order to restrict the number of parameters of the generative model to be trained, an affine projector P_0 is learnt from PCA on latent c-dimensional features to further reduce the dimension to $q \ll c$. Note that this is done while keeping spatial resolution $h \times w$ of latent representation $y_i = E(x_i)$ of images, in such a way that $d_0 = qhw$ is limited to a few thousands dimension. Formally, such an operator operates on each latent-pixel position p and writes

$$\forall\, p \in \Omega,\ P_0 : y(p) \in \mathbb{R}^c \mapsto \text{diag}(\sqrt{s_0}^{-1}) V_0 (y(p) - \bar{y}) \in \mathbb{R}^q \tag{1}$$

where $\bar{y}$ is the average of latent pixels and $V_0 \in \mathbb{R}^{q \times c}$ is the q principal eigenvectors, normalized using the corresponding eigen-values $s_0 \in \mathbb{R}^q$. For decoding, the transpose of V_0 is used to reconstruct latent representation at the required dimension c:

$$\forall\, p \in \Omega,\ P_0^\star : z(p) \in \mathbb{R}^q \mapsto V_0^\top \text{diag}(\sqrt{s_0}) z(p) + \bar{y} \in \mathbb{R}^c. \tag{2}$$

3.2 A Low-Rank Latent Mixture Model

This first stage aims at learning a parametric model capable of synthesising random latent representation that: i) impose correlations across the coarse resolution of embedded images, and ii) capture the diversity of the target distribution, iii) with a limited budget, in both required training data, memory and computation time. To achieve these goals, we combine a gaussian mixture model with low-rank approximations that is trained on the limited target dataset.

GMM. The training dataset $\mathcal{X}$ is embedded as N latent vectors $\mathcal{Y} = \{y_i = P_0(E(x_i)) \in \mathbb{R}^{d_0}\}_{i=1}^N$. An expectation-maximization (EM) algorithm is used to fit the Gaussian Mixture Model (GMM) into K components, so that the training samples $\mathcal{Y}$ are split into K clusters $\{\mathcal{C}_k\}_{k=1}^K$. Each component is indexed by k and is parameterized with a multinomial probability π_k and with a multi-variate gaussian $\mathcal{N}(c_k, \Sigma_k)$. Mean $c_k \in \mathbb{R}^{d_0}$ and covariance $\Sigma_k \in \mathbb{R}^{d_0 \times d_0}$ are empirically estimated from the training data from $\mathcal{C}_k$. Due to the limited amount of data and the reduced dimension (*i.e.* N d_0-dimensional vectors), this training step is

quite fast (a few seconds for small dataset such as $N = 100$), with the additional acceleration from GPU-parallelization (see *e.g.* [23]).

Recall that drawing a random sample from this GMM consists in

1. randomly choosing the index $k \in \{1..K\}$ using the multinomial probability law $\pi = (\pi_k)_{k=1}^K$,
2. sampling a random sample z using

$$\varepsilon \sim \mathcal{N}(0, I_{d_0}) \mapsto z = P_k \varepsilon + c_k \ \in \mathbb{R}^{d_0}, \tag{3}$$

where ε is a random latent variable drawn from a standard normal distribution, and P_k is a lower triangular matrix from the Cholesky decomposition of the covariance matrix, such that $\Sigma_k = P_k P_k^\top$.

LR *Sampling.* While using a large number K of clusters allows to fit arbitrarily well the latent distribution, the number of parameters grows linearly with K. As a result, and in addition to restricting K to avoid overfitting, we propose to further reduce the number of parameters required to encode each Gaussian component by performing a supplementary dimensionality reduction. Relying again on PCA, a low-rank (LR) approximation of the covariance matrix Σ_k is used to limit the latent dimension of each component k to d_k. We have found that adapting d_k to the probability π_k give better visual results, as shown in experiments. In practice, we substitute the matrix $P_k \in \mathbb{R}^{d_0 \times d_k}$ in (3) which is defined, similarly to (1), from the matrix V_k of d_k principal eigenvectors of Σ_k as follows: $\forall\, k = 1..K$,

$$P_k : \varepsilon \in \mathbb{R}^{d_k} \mapsto z = \mathrm{diag}(\sqrt{s_k}^{-1}) V_k \varepsilon + c_k \ \in \mathbb{R}^{d_0} \tag{4}$$

As studied in experiments, this dimension reduction offers some control over the quality versus the diversity of generated samples.

3.3 Refinement Step with Iterated NN-Projection

The second step of the generative process is non-parametric. In our setting, for each spatial position $p \in \{1, \dots h\} \times \{1, \dots w\}$, the random latent feature $z(p) \in \mathbb{R}^q$ generated from cluster indexed by k (as in (3)) is substituted with the nearest-neighbor (NN) latent feature $y_j(p)$ at the same position p and in the same cluster k. To define a relevant comparison of latent feature while taking into account the context, we consider the local $\omega \times \omega$ square neighborhood around p, that is $\forall\, p \in \Omega$

$$\text{NN} : z(p) \in \mathbb{R}^q \mapsto y_j(p) \text{ with } j = \underset{i=1..N \text{ s.t. } y_i \in \mathcal{C}_k}{\mathrm{argmin}} \|\Phi z(p) - \Phi y_i(p)\|^2 \tag{5}$$

where Φ is the patch-extractor defined as, $\forall\, p \in \{\rho, \dots h - \rho\} \times \{\rho, \dots w - \rho\}$

$$\Phi : z(p) \in \mathbb{R}^q \mapsto (z(p + h))_{h \in \{-\rho..\rho\}^2} \in \mathbb{R}^{q \times \omega^2}, \tag{6}$$

considering patches with a discrete radius ρ, such that $\omega = 2\rho + 1$. Some special care is required at the boundary of the domain for which zero-padding may be used for efficiency. Observe as well that only training patches in the same cluster as the query and at the same location p are involved in the NN search (5). When considering non-registered data, such restriction can be like discarded but increase complexity, similarly to attention modules.

Comparison with Path-Based Synthesis. Last, note that this NN projection builds upon the observation from [33,34] that plausible latent code can be obtained by sampling directly local neighborhoods (or patches) of spatial resolution $\omega \times \omega$ from the training data. These approaches have been motivated by earlier patch-based techniques, such as the seminal work of [6] for texture synthesis, and the efficient algorithm of [1] for image editing.

However, contrarily to [33] where large patches from the training set are sequentially copied to generate a new latent representation similar to a patch-work, and to some extent to auto-regressive models [7,28] or diffusion models with attention mechanisms [29], the proposed nearest-neighbor projection only copies a latent feature $y_j(p)$ and is independent for each spatial position, and thus can be performed in parallel. This allows some significant speed-up of the latent synthesis and avoids overfitting the training set as shown in experiments.

Some links with non-local mean filtering and attention mechanism are further discussed in the appendix.

4 Experiments and Applications

In this section, we first describe the experimental setting (Sect. 4.1). Then, we show some quantitative and qualitative results of the proposed approach for face generation in Sect. 4.2. A comparative analysis with other methods is also carried out. Additional results, including an ablation study demonstrating the role of each generative step together with the impact of some hyper-parameters are proposed the supplementary material.

4.1 Experimental Setting

Auto-Encoder. We use the VQ-GAN auto-encoder [7], an architecture originally conceived for transformer-based generation. It notably combines an adversarial loss with a quantization codebook, learning a both compressed and meaningful latent representation. The auxiliary dataset $\mathcal{A}$ in our setting is FFHQ [19], a large dataset of faces high-quality faces.

The auto-encoder is trained using color images at resolution $H = W = 256$ pixels (*i.e.* for a total dimension $3HW = 196,608$). The latent representation of an image has a $h = w = 16$ pixels resolution with $c = 256$ channels (*i.e.* for a total dimension $d_{\text{latent}} = 65,536$).

Fine-Tuning. The feature dimension reduction of the auto-encoder using PCA (as detailed in Sect. 3.1) is performed on latent pixels of the limited target encoded dataset $E(\mathcal{X})$, in which $\mathcal{X}$ is a random subset of N images from CelebA-HQ [18], to learn the projector P_0 (1). Doing so allows us to reduce the channel dimension of pixels from $c = 256$ to $q = 8$ with minimal information loss. In the following, the default size of this limited dataset is set to $N = 1024$, but experiments will show that the model is sufficiently robust for N to range from as low as $N = 32$ (as in the few-shot regime shown in Fig. 1) to the full dataset with $N = 28k$.

Fitting the Generative Model. During the training phase of the proposed generative model (LR-GMM), several hyper-parameters need to be set. By default, we fit $K = 8$ components with dimensions $d_k = \lfloor \lambda \pi_k d_0 \rfloor$ where temperature parameter $\lambda = 0.12$. Optimization is performed using standard EM algorithm (here we rely on the non-parallel implementation of `sklearn.mixture.GaussianMixture`).

Memory and Computations. Additional details about memory footprint and computation time are provided in the supplementary material. In a nutshell, the proposed model is both fast to train (few seconds for $N = 1000$ images) and to sample from (few milliseconds on a GPU).

4.2 Face Generation

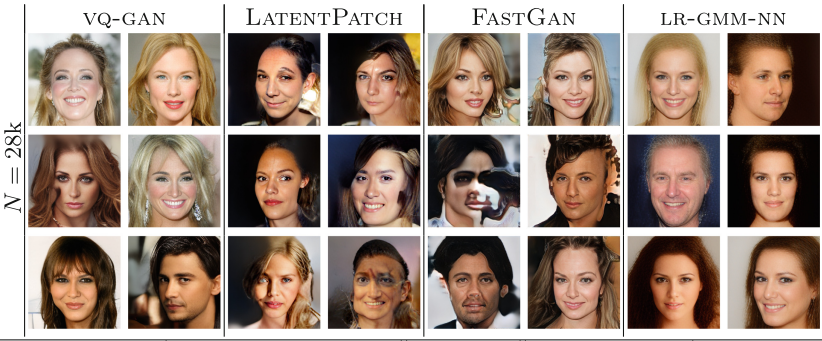
In this section, we discuss quantitative and qualitative results of the proposed method for portrait generation in two regimes (limited or large dataset), and we provide comparisons with relevant methods.

Fine-Tuning on a Small Dataset. Figure 1 illustrates the two steps of proposed generative model when resorting to only $N = 32$ random training images from CelebAHQ. In this setting, only $K = 1$ component is used. Random samples obtained from the LR-GMM step alone synthesize the appropriate main face features (eyes, nose and mouth) with plausible spatial correlation. However, some other regions is the image may be unrealistic, such as the face contour and the hair, the lack of details, the existence of artificial high-frequency patterns, etc. The NN projection step (here with only one iteration) aims at improving this aspect by replacing local features with examples from the training images. This observation is confirmed by several metrics reported later on. in Table 1. The "Latent map" (at the bottom of Fig. 1 with an arbitrary colormap) corresponds to the index of the nearest neighbor; in addition to the visual comparison with the training images (top of Fig. 1), it demonstrates that the model is not overfitting training samples, even locally.

Quantitative Analysis. Now we consider a large dataset in order to compute objective metrics such as FID [13] and IPR [22]. Recall that the Fréchet Inception Distance measures the dissimilarity between empirical distributions of real and synthesized samples, and that Improved Precision-Recall aims at measuring

the tradeoff between quality and diversity of synthetic images. For this purpose, a total of 10k images are generated with several methods. Results reported in Table 1 for $N = 1024$ and $N = 28$k first show that both steps (LR-GMM and NN) of the proposed method are required to improve the quality of samples. The comparison with LatentPatch demonstrates that the proposed method can achieve similar performance without overfitting the training set (which is not penalized by these metrics). Besides, comparisons with two larger parametric models (FASTGAN [24] and VQ-GAN [7]) that requires hours of training show that our method is competitive. Extensive visual comparisons in supplementary material corroborate these results: the proposed method yields high quality samples (high precision) but lacks some diversity (lower recall). In particular, the proposed method struggles to create realistic hair pattern, which is penalized by the FID.

Table 1. FID [13] and Precision-Recall [22] metrics for different methods. Our method (LR-GMM-NN) is tested in two settings: $N = 1024$ and $N = 28,000$ images. LR-GMM indicates the proposed method without the refinement NN step, demonstrating its interest. See the supplementary material for more visual samples. *Note that both the transformer and auto-encoder of VQ-GAN have been trained on $\mathcal{X}$ rather than $\mathcal{A}$.



Datasize $\lvert\mathcal{X}\rvert$	Method	FID (↓)	Precision (↑)	Recall (↑)
$N = 1024$	LatentPatch [33]	58.4	0.55	0.01
	LR-GMM	73.8	0.47	0.03
	LR-GMM-NN	58.1	0.52	0.07
$N = 28$k	VQ-GAN* [7]	10.5	0.52	0.51
	LatentPatch [33]	35.1	0.54	0.19
	FastGan [24]	31.8	0.50	0.04
	LR-GMM	84.1	0.53	0.01
	LR-GMM-NN	45.0	0.68	0.06

Comparison with LatentPatch. Figure 3 offers a comparison of both approaches when considering the same dataset of $N = 1024$ images. We already reported some methodological difference with LatentPatch [33] in Sect. 3.3,

completed with numerical (computation time) in the supplementary material where our approach is reported to yield a 30× speed-up. Recall that LatentPatch is a non-parametric patch-based generative model that does not require any training, and that aims at being explainable. To do so, similarly to other patch-based generative models such as [11], it explicitly synthesizes new image samples by copying local regions from examples images. This can be seen from the Nearest-Neighbor index, as defined by (5) and referred to the "latent map" (in this case, the colormap is cluster-dependant). In contrast with LatentPatch which copies large regions of latent features to achieve realistic portrait generation, the proposed method combines only latent pixels from training samples, thus with more variety without overfitting some training samples. As a result, thanks to the randomness of samples from the GMM, we only make use of the nearest feature for the refinement, rather than sampling randomly from the top k-NN as in LatentPatch or in VQ-GAN. For all methods, the use of a latent representation then allows for a nice blending of the selected features.

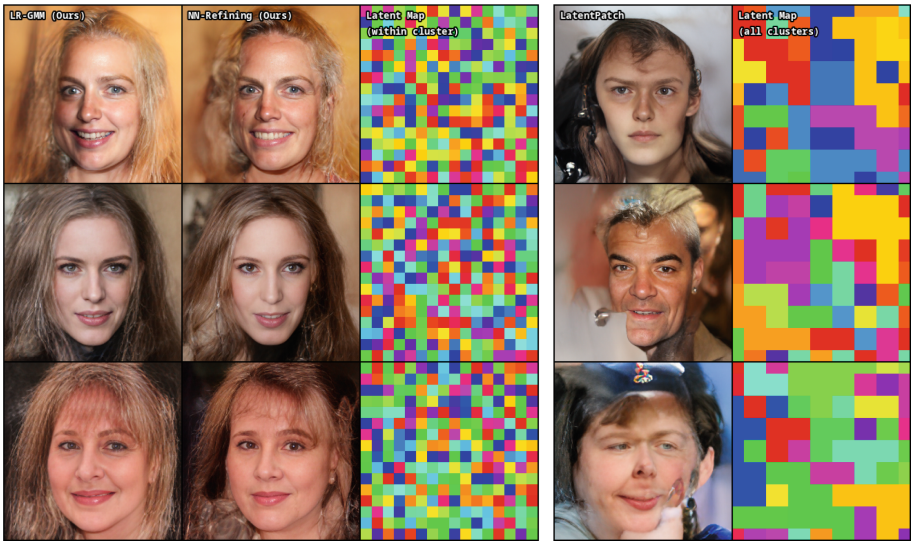


Fig. 3. Comparisons between LLR-GMM-NN on the left, and LatentPatch [33] on the right, both fitted on the same $N = 1024$ dataset. The proposed approach benefits from training a small parametric model to impose spatial coherence.

4.3 Ablation Study

The supplementary material includes an ablation analysis that offers extensive visual comparisons, corroborating and complementing the performance scores reported in earlier experiments. This is supplemented with additional experiments that discuss the significance of each stage of the proposed generative model, the role of various hyperparameters, and the computation time.

5 Discussion and Perspectives

Summary. A new approach has been presented for face-generation from limited dataset. Rather than training a large auto-regressive model such as VQ-VAE or VQ-GAN, our approach shows that in this context a simpler parametric model can be efficiently learned from a few samples. The proposed latent generative model mainly consists in two stages. The first one fits a gaussian mixture model from the limited dataset. This model is then reduced in dimension to accelerate training and inference, but also improve quality results. This simplistic model provides random samples with the desired spatial correlations. The final stage consists in improving the quality of the synthesized samples by local nearest-neighbor projections. Experimental analysis demonstrates the speed-up of the proposed method in comparison with some previous work from the literature, while still providing similar or better visual quality.

It is noteworthy that previous works, such as [9,35], have already demonstrated the benefits of combining regularized deterministic auto-encoders with Gaussian Mixture Models to enhance vanilla Variational auto-encoder models. Our approach differs from [35], where latent dimensions are assumed to be independent, enabling the estimation of univariate GMMs during training. Similar to [9], we demonstrate that a multivariate GMM can be post-estimated to capture correlations between latent variables and improve sample quality. However, our work also diverges from [9] in several key ways. First, we employ dimensionality reduction to enhance sample quality, as shown in the ablation study in the supplementary material. Additionally, our method leverages the spatial information in the latent representation to refine local statistics using NN projections. Furthermore, instead of utilizing the entire training dataset used for the regularized AE, we demonstrate that a limited sample is sufficient to fit the latent model of an AE-GAN to produce realistic samples.

Limitations and Perspectives. The proposed method however suffers from a few shortcomings when compared to some previous methods. The first one is related to the resolution of synthesized images that is same as training samples, like most generative models. This is yet not the case with auto-regressive models that can generate latent codes of arbitrary size, which is handy for other tasks such as generating landscapes. An interesting line of research in such context would be to extend our model by considering GMM with limited range for spatial correlations. In comparison with latentPatch, another current limitation of the proposed model is its extension to conditional generation. With an auto-regressive model, a latent code can be easily provided as a context. In our setting, this is also achievable but not as straightforward as one need to train a conditional gaussian noise, which is an interesting perspective left for future work.

Acknowledgements. This work is funded by the project ANR-19-CHIA-0017.

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
2. Ben-Moshe, L., Benaim, S., Wolf, L.: FewGAN: generating from the joint distribution of a few images. In: IEEE ICIP, pp. 751–755 (2022)
3. Berthelot, A., Caron, E., Jay, M., Lefèvre, L.: Estimating the environmental impact of generative-AI services using an LCA-based methodology (2023)
4. Carlini, N., et al.: Extracting training data from diffusion models. arXiv preprint arXiv:2301.13188 (2023)
5. Child, R.: Very deep VAEs generalize autoregressive models and can outperform them on images. In: International Conference on Learning Representations (2020)
6. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1033–1038 (1999)
7. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12873–12883 (2021)
8. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423 (2016)
9. Ghosh, P., Sajjadi, M.S.M., Vergari, A., Black, M., Scholkopf, B.: From variational to deterministic autoencoders. In: International Conference on Learning Representations (2020)
10. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
11. Granot, N., Feinstein, B., Shocher, A., Bagon, S., Irani, M.: Drop the GAN: in defense of patches nearest neighbors as single image generative models. In: Proceedings of the IEEE/CVF CVPR, pp. 13460–13469 (2022)
12. Hayes, J., Melis, L., Danezis, G., De Cristofaro, E.: Logan: membership inference attacks against generative models. Proc. Privacy Enhancing Technol. (2019)
13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
14. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Adv. Neural. Inf. Process. Syst. **33**, 6840–6851 (2020)
15. Hu, E.J., et al.: Lora: low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
16. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43
17. Kang, M., et al.: Scaling up GANs for text-to-image synthesis. In: Proceedings of the IEEE/CVF CVPR, pp. 10124–10134 (2023)
18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)
19. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF CVPR, pp. 4401–4410 (2019)

20. Kenthapadi, K., Lakkaraju, H., Rajani, N.: Generative AI meets responsible AI: practical challenges and opportunities. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 5805–5806 (2023)
21. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
22. Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., Aila, T.: Improved precision and recall metric for assessing generative models. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
23. Lee, S.X., Leemaqz, K.L., McLachlan, G.J.: A simple parallel EM algorithm for statistical learning via mixture models. In: 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1–8. IEEE (2016)
24. Liu, B., Zhu, Y., Song, K., Elgammal, A.: Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis. In: International Conference on Learning Representations (2020)
25. Nasr, M., et al.: Scalable extraction of training data from (production) language models. arXiv preprint arXiv:2311.17035 (2023)
26. van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, K., Vinyals, O., Graves, A.: Conditional image generation with pixelcnn decoders. In: Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc. (2016)
27. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv:2204.06125 (2022)
28. Razavi, A., van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with VQ-VAE-2. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
29. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF CVPR, pp. 10684–10695 (2022)
30. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 22500–22510 (2023)
31. Saharia, C., Chan, W., Saxena, S., Li, L., et al.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems, vol. 35, pp. 36479–36494. Curran Associates, Inc. (2022)
32. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. In: International Conference on Learning Representations (2021)
33. Samuth, B., Rabin, J., Tschumperlé, D., Jurie, F.: Latentpatch: a non-parametric approach for face generation and editing. In: 2023 IEEE International Conference on Image Processing (ICIP), pp. 1790–1794. IEEE (2023)
34. Samuth, B., Tschumperlé, D., Rabin, J.: A patch-based approach for artistic style transfer via constrained multi-scale image matching. In: 2022 IEEE International Conference on Image Processing (ICIP), pp. 3490–3494 (2022)
35. Saseendran, A., Skubch, K., Falkner, S., Keuper, M.: Shape your space: a gaussian mixture regularization approach to deterministic autoencoders. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021)
36. Shan, S., Cryan, J., Wenger, E., Zheng, H., Hanocka, R., Zhao, B.Y.: Glaze: protecting artists from style mimicry by {Text-to-Image} models. In: 32nd USENIX Security Symposium (USENIX Security 2023), pp. 2187–2204 (2023)

37. Shan, S., Ding, W., Passananti, J., Zheng, H., Zhao, B.Y.: Prompt-specific poisoning attacks on text-to-image generative models. arXiv preprint arXiv:2310.13828 (2023)
38. Somepalli, G., Singla, V., Goldblum, M., Geiping, J., Goldstein, T.: Diffusion art or digital forgery? Investigating data replication in diffusion models. arXiv preprint arXiv:2212.03860 (2022)
39. Thiel, D.: Identifying and eliminating CSAM in generative ml training data and models. Technical report, Stanford University, Palo Alto, CA (2023)
40. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
41. Wang, Y., Wu, C., Herranz, L., Van de Weijer, J., Gonzalez-Garcia, A., Raducanu, B.: Transferring GANs: generating images from limited data. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 218–234 (2018)
42. Webster, R., Rabin, J., Simon, L., Jurie, F.: Detecting overfitting of deep generative networks via latent recovery. In: Proceedings of the IEEE/CVF CVPR, pp. 11273–11282 (2019)
43. Yoon, J., Jordon, J., van der Schaar, M.: PATE-GAN: generating synthetic data with differential privacy guarantees. In: International Conference on Learning Representations (2019)
44. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (2017)
45. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF CVPR, pp. 3836–3847 (2023)
46. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–595 (2018)

Domain Adaptation for Machinery Fault Diagnosis Based on Critic Classifier GAN

Tso-Sung Hung and Shang-Hong Lai(✉)

National Tsing Hua University, Hsinchu, Taiwan
lai@cs.nthu.edu.tw

Abstract. Domain adaptation is a crucial challenge in the field of machinery fault diagnosis, as the performance of traditional fault diagnosis models could significantly degrade when applied to different working conditions or domains. This paper proposes a novel approach for domain adaptation for machinery fault diagnosis based on the Critic Classifier Generative Adversarial Network (GAN). Our method aims to improve diagnostic performance by aligning the source and target domains, enabling effective knowledge transfer between them. We leverage the power of the Critic Classifier GAN framework, which incorporates both a generator adversarial network and a critic classifier. This framework enables us to learn representations invariant to domain shifts, leading to accurate classification of fault patterns. Additionally, we employ domain discrepancy loss functions, such as Maximum Mean Discrepancy (MMD) and Maximum Classifier Discrepancy (MCD), to further enhance domain alignment and classifiers to align the feature distributions. Experimental evaluations conducted on various mechanical failure datasets confirm our proposed method's effectiveness and robustness compared to existing domain adaptation techniques. Our proposed solution effectively overcomes the challenges arising from domain shift and achieves state-of-the-art performance in machinery fault diagnosis under various working conditions.

Keywords: Unsupervised · Domain Adaptation · Classifier alignment · Fault Diagnosis · GAN

1 Introduction

In recent years, there have been notable advancements in mechanical fault diagnosis using deep learning methods combined with signal preprocessing. Convolutional Neural Networks (CNNs) have shown promising results for numerous classification tasks. Yet, their performance often depends on extensive labeled data, which may not be available in practical scenarios with varying data distributions. This domain shift problem arises because data from different sources may differ significantly. To address this challenge, State-of-the-art (SOTA) approaches, such as [1,2], have been proposed to adapt fault diagnosis models from a source

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 350–365, 2025.
https://doi.org/10.1007/978-3-031-78172-8_23

domain to a target domain significantly when their data distributions vary broadly. This approach aims to minimize the disparity between source and target domains, aligning their feature representations and transferring knowledge to enhance fault diagnosis performance in the target domain. Bearing fault diagnosis is crucial in smart factories, where machinery operates autonomously. In this paper, We propose a novel domain adaptive machinery fault diagnosis framework that utilizes datasets from different sources, reducing the need for excessive sensors in the field and improving efficiency. The proposed method leverages GAN and domain adaptation techniques to enhance fault diagnosis under different working conditions. It addresses domain shift challenges, improving model accuracy and generalization across domains. Experimental results demonstrate its effectiveness and reveal the impact of data characteristics on fault diagnosis performance.

This paper presents a novel domain adaptation method that leverages deep learning and domain adaptation to improve fault diagnosis under varying conditions. The proposed GAN-based approach effectively addresses domain shift challenges, enhancing the reliability and accuracy of fault diagnosis in practical applications. The contributions in this paper are listed as follows:

1. The proposed GAN-based method addresses the challenge of domain shift and improves the generalization of fault diagnosis models.
2. Through experiments on domain adaptation, we show that our method outperforms previous methods in fault diagnosis tasks under different working conditions.

2 Related Work

Domain Adaptation aims to identify similar instances between the source and target domains and train a classifier using source domain samples to transfer the model learned from the source domain to the target domain. Several domain adaptation methods have been proposed and can be classified into the following approaches.

A) Mapping-based methods: Traditional alignment methods consider the difference in the decision boundary between the source-domain and target-domain features generated by the feature extractor. These methods strive to bridge the gap between the source and target domains by aligning their feature distributions. For example, the Maximum classifier discrepancy (MCD) [3] can effectively align the difference on the decision boundary. The Maximum classifier alignment approach [4] has also been applied to many tasks. The Maximum Mean Discrepancy (MMD) [5] has been widely used as a distance metric to enforce the distributions of the learned domain representations closer to each other. In addition, CORAL [6] uses second-order statistics to minimize domain shifts for aligning two distributions.

B) Reconstruction-based methods: The reconstruction-based approach learns a model to capture the latent data structure and then creates a synthetic reconstruction; the main idea is to extract features that represent the data and assume that a high reconstruction error means a high probability of anomalous data. By minimizing the discriminability of the domain classifier, they align the feature distributions between the source and target domains. Deep reconstruction-based neural network architectures such as CRDN [7] and ReverseGrad [8] have been investigated more recently. These architectures are trained to reconstruct the input data accurately.

C) Instanced-based methods: The Instance-based methods utilize techniques such as re-weighting instances in the source domain and incorporating target domain statistics. These methods are commonly employed in deep learning models to address the domain shift between the source and target domains. For example, Adaptive Batch Normalization (AdaBN) [9] has been used to improve the generalization of DNNs. By modulating statistics from the source domain to the target domain in all Batch Normalization layers throughout the network, they only update the global mean and variance, equivalent to realizing domain adaptation.

D) Adversarial-based methods: GAN-based domain adaptation methods optimize feature extraction through adversarial training, enabling them to learn domain-specific knowledge with strong generalization capabilities. Recently, a domain adaptation approach [10]focuses on transferring from synthetic to accurate data. This approach aims to bring the source and target distributions closer in a learned joint feature space, thereby enhancing the effectiveness of domain adaptation. Another notable approach, Coupled Generative Adversarial Networks (CoGAN) [11], learns the joint distribution of images in two domains separately. This is achieved by enforcing a simple weight-sharing constraint while considering the marginal distributions within each domain. CoGAN leverages this joint distribution learning to improve the domain adaptation results and facilitate the alignment of features between the source and target domains.

The difficulty of domain adaptation can vary depending on the characteristics of the data and the domains involved. In the case of homogeneous and heterogeneous domains, specific challenges contribute to the complexity of domain adaptation. Kang et al. [12] proposed a CNN network incorporating Contrastive Domain Discrepancy (CDD), which builds upon MMD to measure intra-class and inter-class discrepancies across domains. The CDD aims to minimize intra-class domain discrepancy within a class while maximizing inter-class domain discrepancy, facilitating cross-domain adaptation. Lin et al. [13] and Dai et al. [14] employed neural networks and adversarial learning techniques, respectively, to minimize the discrepancy of joint distribution (MMD) and maximize classifier discrepancy (MCD) to produce more discriminative features for unsupervised domain adaptation. Overall, the difficulty of domain adaptation in homogeneous and heterogeneous domains arises from data distribution deviation, feature relevance, and the challenge of generalizing models across domains with limited labeled data.

3 Proposed Method

We propose a domain adaptation method to transfer the fault diagnosis model from the source domain to the target domain through adversarial learning. Our method learns a joint feature space by utilizing adversarial image generation to minimize the distribution distance between the source and target domains, thus resolving the domain shift in the feature space learned by the encoder. The details of our method are described in this section.

3.1 Overview

We introduce an innovative framework combining unsupervised target distribution data with supervised learning based on source distribution samples. The framework consists of three core components:

F-Network (Feature Extraction Network): F-Network (Feature Extraction Network): This network extracts features from input data to create comprehensive time-frequency images through feature vector generation. We recommend using wide kernels in the initial convolutional layer to capture relevant signal information, especially in intermediate and low-frequency bands. Subsequent layers with smaller kernels enable deeper network learning for more detailed feature extraction, effectively suppressing high-frequency noise.

GAN-Network: Responsible for predicting source labels and assisting the G-Network (Generator) generate realistic source images. The authenticity of these images is evaluated by the D-Network (Discriminator). The adversarial process continues until the generator creates samples indistinguishable from accurate data. Simultaneously, the F-Network receives updates from the D-Network via the C-Network and from the G-Network within the adversarial framework. In the unlabeled target domain, the F-Network relies on gradient updates from adversarial training, allowing it to learn discriminative features effectively by leveraging domain knowledge from the generator-discriminator pair.

Alignment Method: The discriminator serves as a multi-category classifier, employing the Maximum Classifier Discrepancy (MCD) method to identify target samples deviating from the source data distribution. Discrepancies are reduced by comparing the outputs of the two classifiers, known as the Critic method. This leads to the training of a feature generator that produces target features closely aligned with the source, minimizing discrepancies. Trained feature extraction and generators can be used directly to reconstruct more compact time-frequency data, providing more accurate reconstruction errors. Furthermore, we incorporate the Maximum Mean Discrepancy (MMD) and Gradient Reversal Layer (GRL) [15] techniques into our framework.

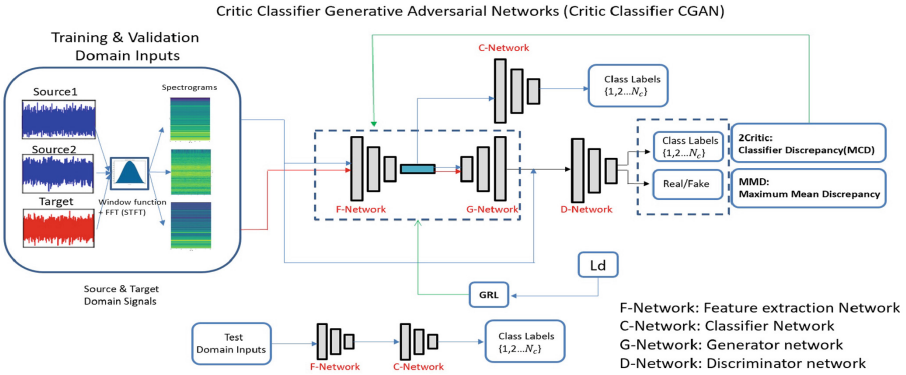


Fig. 1. The architecture of the proposed model includes an F-network with 1D convolutional layers and a G-D-C network based on the Auxiliary Classifier GAN (ACGAN) framework.

3.2 Network Architecture

We employ Auxiliary Classifier Generative Adversarial Network (ACGAN) architecture [16] for our model. It is a variant of Conditional GAN to classify images [17] accurately. The architecture of the proposed domain adaptation model is depicted in Fig. 1.

In a standard GAN and Conditions GAN, the discriminator network D estimates the probability that the input image is a sample drawn from the data-generating distribution. This is typically implemented using a feedforward network. However, Semi-Supervised GAN (SGAN) [18] can also be implemented using a Softmax output layer, and the discriminator can also act as a classifier with one unit for each of the classes (real or fake).

ACGAN is derived from the above GAN module, integrating CGAN and SGAN, which provides conditional information at the input and models the discriminator as a multi-classifier. At the same time, it can improve the performance of the original generative task model by providing additional information to the GAN. Specifically, it can use auxiliary class label information to generate high-quality samples and solve domain adaptation and classification problems. To address the challenges of domain adaptation in fault diagnosis, we modify the auxiliary classifier of ACGAN to the Critic method, which identifies target samples deviating from the source data distribution and uses a feature generator to reduce discrepancy and generate target features closer to the support. Furthermore, the network architecture integrates MMD and GRL techniques, enhancing the domain adaptation capability and improving the overall performance.

3.3 Domain Adaptation

Domain adaptation facilitates knowledge transfer from a source domain to a related yet distinct target domain. Our extended ACGAN model introduces sev-

eral components to facilitate adversarial learning and domain alignment, enhancing performance. MMD measures dissimilarity between probability distributions by comparing samples from each distribution. We aim to find a transformation function that minimizes the discrepancy between the transformed data from the source domain and the target domain. The MMD distance is approximated as:

$$MMD(X_s, X_t) = \left\| \frac{1}{|X_s|} \sum_{x_s \in X_s} \phi(X_s) - \frac{1}{|X_t|} \sum_{x_t \in X_t} \phi(X_t) \right\| \tag{1}$$

$$\mathcal{L}_{mmd} = \lambda MMD^2(X_s, X_t) \tag{2}$$

where λ is a hyperparameter controlling the strength of the domain confusion term in the loss function.

Maximum Classifier Discrepancy (Critic Classifier) identifies target samples far from the source domain's support by leveraging the discrepancy between two classifiers. Training a feature generator to produce target features near the support minimizes this discrepancy. The critic classifier discrepancy loss is defined as:

$$\mathcal{L}_{critic} = d(p_1, p_2) = \frac{1}{K} \sum_{k=1}^{K} |p_{1k} - p_{2k}| \tag{3}$$

where p_{1k} and p_{2k} represent probability outputs for class k. Gradient Reversal Layer (GRL) is introduced to handle the conflicting objectives of the Generator and Discriminator in GANs. GRL multiplies the error passed to a layer by a negative number(λI), creating an adversarial effect by opposing the training objectives before and after GRL.

3.4 Objective Functions

The primary loss function remains the same as that in ACGAN. We have integrated Critic and MMD module block loss functions into the feature reconstruction F-network to enhance domain adaptability and improve the fault diagnosis model's generalization ability and performance:

$$\mathcal{L}_f = \mathcal{L}_c + \mathcal{L}_{c,scr} + \beta \mathcal{L}_{F_{adv}} + \mathcal{L}_{critic} + \mathcal{L}_{mmd} \tag{4}$$

3.5 Training Procedure

Algorithm Iterative training procedure of Critic Classifier GAN.

1: The number of training iterations = N
2: for t in 1:N, do
3: sample n signals with labels from source domain $D_s = \{(X_{si}, Y_{si})\}_{i=1}^{n}$
4: Let $f_i = F(X_{si})$ The embeddings are computed for the source signals.
5: Sample n signal from target domain $D_t = \{X_{ti}\}_{i=1}^{n}$
6: Let $h_i = F(X_{t^i})$ be the embeddings are computed for the target signals.
7: Sample n random noise sample $\{Z_i\}_{t=1}^{n} \mathcal{N}(0, 1)$.

8: Let f_{gi} and h_{gi} be the concatenated inputs to the generator G-network.
9: Update discriminator D-network using the following objectives:
$\mathcal{L}_d = \mathcal{L}_{data,src} + \mathcal{L}_{adv,tgt} + \mathcal{L}_{critic} + \mathcal{L}_{mmd}$

* $\mathcal{L}_{data,src} = \max_D \frac{1}{n} \sum_{i=1}^{n} \log(D_{data}(X_{si}) + \log(1 - D_{data}(G(f_{gi})))$
* $\mathcal{L}_{c,src} = \max_D \frac{1}{n} \sum_{i=1}^{n} \log(D_{critic}(X_{si})_{yi})$
* $\mathcal{L}_{adv,tgt} = \max_D \frac{1}{n} \sum_{i=1}^{n} \log(1 - D_{data}(G(h_{gi})))$
* $\mathcal{L}_{critic} = \min_{P1,P2} = \frac{1}{K} \sum_{k=1}^{K} |p_{1k} - p_{2k}|$
* $\mathcal{L}_{mmd} = \left\| \lambda \frac{1}{|X_s|} \sum_{x_s \in X_s} \phi(X_s) - \lambda \frac{1}{|X_t|} \sum_{x_t \in X_t} \phi(X_t) \right\|$

10: The G-network is updated exclusively using source data, while the discriminator D-network gradients are computed using real labels.
$\mathcal{L}_g = \min_G \frac{1}{n} \sum_{i=1}^{n} -log(D_{ctitic}(G(f_{gi})_{yi})) + \log(1 - D_{data}(G(f_{gi})))$
11: The feature F-network is updated by incorporating a linear combination of adversarial loss and classification loss. This combination is used to guide the updating process:
$\mathcal{L}_f = \mathcal{L}_c + \mathcal{L}_{c,src} + \beta \mathcal{L}_{F_{adv}} + \mathcal{L}_{critic} + \mathcal{L}_{mmd}$

* $\mathcal{L}_c = \min_C \min_F \frac{1}{n} \sum_{i=1}^{n} -\log(C(f_i)_{yi})$
* $\mathcal{L}_{c,src} = \min_F \frac{1}{n} \sum_{i=1}^{n} -\log(D_{critic}(G(f_{gi}))_{yi})$
* $\mathcal{L}_{F_{adv}} = \min_F \frac{1}{n} \sum_{i=1}^{n} \log(1 - D_{data}(G(h_{gi})))$
* $\mathcal{L}_{critic} = \min_{P1,P2} = \frac{1}{K} \sum_{k=1}^{K} |p_{1k} - p_{2k}|$
* $\mathcal{L}_{mmd} = \left\| \lambda \frac{1}{|X_s|} \sum_{x_s \in X_s} \phi(X_s) - \lambda \frac{1}{|X_t|} \sum_{x_t \in X_t} \phi(X_t) \right\|$
* Parameter β is the weight coefficient for the target adversarial loss.

12: end for

4 Experimental Results

Some important details in our experiment are outlined as follows:

a) Dataset Variation: We experiment with different datasets and explore three input types.
   - Time Domain (T): Raw, unprocessed signals.
   - Frequency Domain (F): Signals transformed into the frequency domain.
   - Time-Frequency Domain (T-F): Signals in both time and frequency domains.

b) Training Configuration: Each epoch employs mini-batches with a size of 100. Backpropagation updates all parameters, and we utilize the Adam optimization method. And the total epoch number of interactions is set at 100.

c) Data Splitting: The time series data is randomly split into two sets, with 80% allocated to the training set and 20% to the testing set.

Table 1. The accuracy (%) achieved on domain adaptation problems using the CWRU dataset

	A->B	A->C	B->A	B->C	C->A	C->B	AVG
SVM-FFT [20]	68.60	60.00	73.20	67.60	68.40	62.00	66.63
MLP-FFT [20]	82.10	85.60	71.50	82.40	81.80	79.00	80.40
DNN-FFT [20]	82.20	82.60	72.30	77.00	76.90	77.30	78.05
WDCNN [21]	99.20	91.00	95.10	91.50	78.10	85.10	90.00
A2CNN [22]	99.99	99.30	98.18	99.90	97.93	99.99	99.21
OUR(T)	97.59	92.77	97.59	100.00	96.98	96.38	96.88
OUR(F)	100.00	89.64	96.75	94.49	92.85	100.00	95.62
OUR(T-F)	100.00	99.02	98.69	99.02	98.21	100.00	99.15

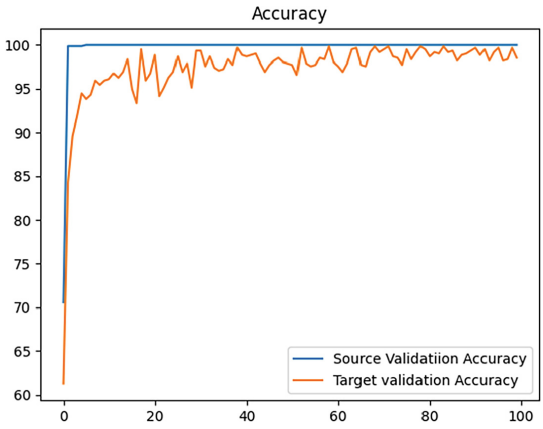


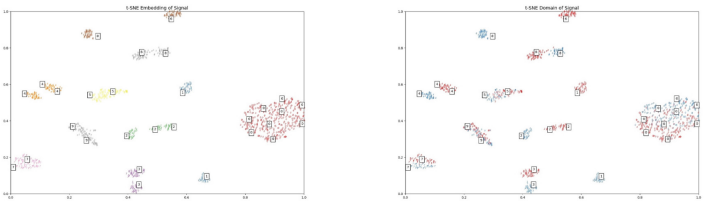
Fig. 2. Accuracy and fault classification of B->C (T-F) of CWRU Dataset.

4.1 CWRU Bearing Dataset Experiment Details

The CWRU Bearing Fault Dataset [19], provided by Case Western Reserve University's Bearing Data Center, is a widely recognized dataset extensively used in fault diagnosis research. The dataset comprises four sets, each characterized by varying loads and fixed revolutions per minute (RPM) conditions. There are ten categories in each group, and we use the Motor Load A(1HP), B(2HP), and C(3HP), respectively (Fig. 3).

4.2 Experiment of CWRU Comparison and Analysis

The diagnostic performance comparison under different domain classifiers is presented in Table 1. The results demonstrate that some previous methods, such as SVM-FFT, MLP-FFT, DNN [20] and WDCNN [21], have limited success in domain adaptation. The overall method of sample extraction in different domains



(a) Fault classification (b) Domain distribution

Fig. 3. Domain distribution and classification B->C (T-F) of CWRU Dataset.

is unsuitable for cross-domain fault classification tasks. However, what sets our architecture apart from A2CNN [22] is that it considers domain overlaps and the decision boundaries of specific feature tasks. Experiments demonstrate that our approach can achieve high accuracy with minimal interactions, making it more adaptable and generalizable.

A2CNN requires a huge labeled dataset for pre-training to obtain the best result, which could lead to the over-fitting problem. In addition, A2CNN only considers the distribution of the target domain and the source domain, but it does not consider the decision boundary of the specific task. In contrast, our method finds the global domain distribution MMD and employs MCD to make the features away from the classification boundary, thus leading to excellent generalization ability. It is presented in Fig. 2. Table 1 demonstrates the superior adaptation capability of our model in different input types. Our method can achieve high accuracy even with minimal training and iterations, and it can be quickly trained and deployed in practical applications. Under different domain faults, our model consistently outperforms other comparison methods, achieving higher accuracy. This highlights the effectiveness and robustness of our method.

4.3 JNU Bearing Dataset Experiment Details

The JNU bearing datasets, as provided by Jiangnan University [23], include three separate datasets of bearing vibration data acquired at a sampling rate of 50 kHz under different rotational speeds: Dataset A at 600 RPM, Dataset B at 800 RPM, and Dataset C at 1000 RPM. Each dataset contains data for one healthy condition and four fault modes: Normal state (NB), inner ring fault (IF), outer ring fault (OF), and rolling element fault (RF). Consequently, twelve classes are in total, determined by the different working conditions within the datasets.

4.4 Experiment of JNU Comparison and Analysis

We compare the performance of our model with several state-of-the-art fault diagnosis methods on the bearing dataset from JNU Datasets. Table 2 shows that our process differs from the DANN and CDAN models using the GAN

Table 2. The accuracy (%) achieved on domain adaptation problems using the JNU dataset

	A->B	A->C	B->A	B->C	C->A	C->B	AVG
1D-CNN [24]	89.80	78.19	73.75	89.93	85.53	88.84	84.34
AdaBN [8]	84.68	73.65	84.10	87.68	85.87	88.12	84.01
MK-MMD [25]	97.44	96.93	92.15	96.93	92.83	97.44	95.62
JMMD [26]	97.44	97.78	93.34	97.61	92.15	97.95	96.04
CORAL [27]	82.94	71.67	67.41	83.96	79.86	91.47	79.55
JMMD [15]	96.25	94.88	92.83	97.10	92.83	92.15	94.34
CDAN [28]	97.44	95.39	92.83	96.93	91.13	90.61	94.05
OUR(T)	97.61	96.07	87.37	98.63	88.73	98.80	94.53
OUR(F)	76.96	77.64	73.37	84.12	74.06	84.81	78.49
OUR(T-F)	91.63	92.63	85.06	97.01	88.13	95.73	91.70

method. According to the experimental comparison, their input signals for the JNU dataset use the time domain (T). Still, our model uses the frequency domain (F) input, resulting in poor accuracy. From the above two experiments, it is found that different data sets use different input types but have different results. Possible explanations for why the time domain may perform better than the frequency domain in our experimental results could be the characteristics of the signal and the nature of the signal itself. Specific fault patterns or features may be more easily detectable or distinguishable in the time domain representation. On the other hand, frequency domain representations, such as Fourier transforms, may be more sensitive to noise or irrelevant frequency components. In contrast, the time domain representation directly captures the temporal variations of the signal, which may be less affected by noise. Another reason could be that the feature extraction methods used in the frequency domain may not effectively capture the relevant information for fault diagnosis in your specific dataset. It might be necessary to consider different frequency bands or spectral features to improve the performance. Furthermore, exploratory time-frequency domain (T-F) experiments present that more high-frequency signals can be captured by applying suitable filters. This can provide a more comprehensive signal analysis and improve fault diagnosis accuracy. May relate the other to the method of data collection. For example, only different speeds are used in the JNU data set, while CWRU uses different speeds and loads to simulate. However, from the above experimental results, domain confrontation seems helpful for domain shift and alignment. Overall, the choice of input types and the selection of appropriate feature extraction methods depend on the specific characteristics of the signal and the diagnostic task. It is essential to consider these factors when interpreting the performance of different domains in signal analysis experiments (Figs. 4 and 5).

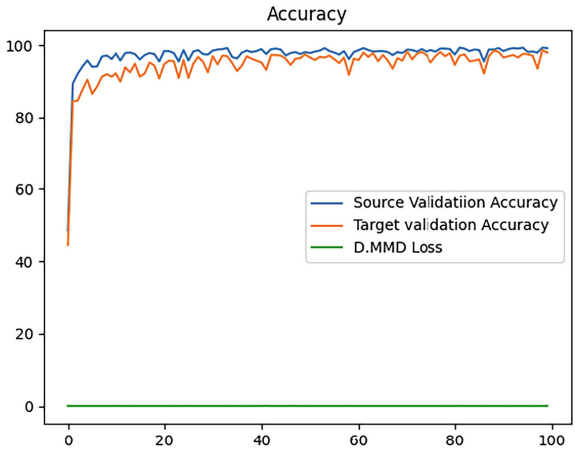
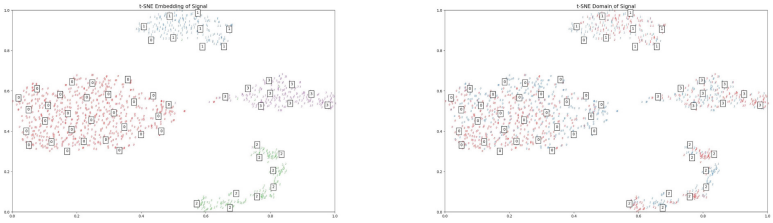


Fig. 4. Accuracy and fault classification of B->C (T) of JNU Dataset.



(a) Fault classification (b) Domain distribution

Fig. 5. Domain distribution and classification B->C (T) of JNU Dataset.

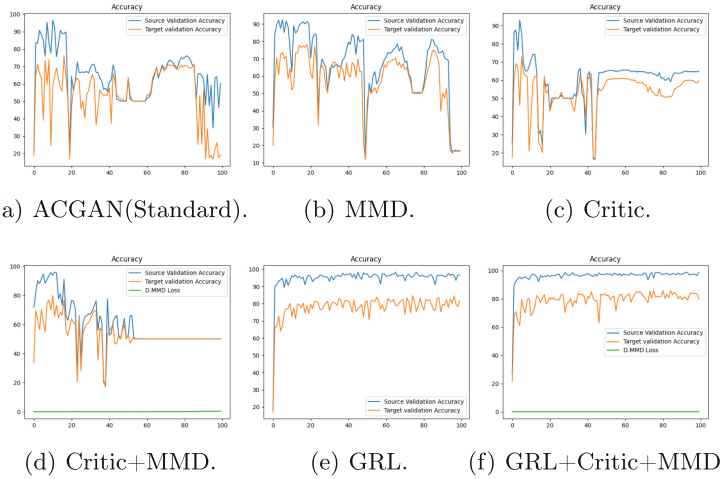
4.5 Ablation Study

In this experiment, we investigate the impact of incorporating additional components in ACGAN on the overall performance. We conduct an ablation study using the worst-performing dataset as an example to observe the effects on the overall performance. The ablation study results are shown in Table 3, The following components are analyzed:

1. Using Auxiliary Classifier Generative Adversarial Network (ACGAN) to analyze the model's performance in the source and target domains and serve as a baseline.
2. Using MMD loss in the adversarial network of ACGAN, We investigate the effects of augmenting the adversarial network with an increased Maximum Mean Discrepancy (MMD) loss to enhance the alignment between domains.
3. Changing the auxiliary classifier in the original D-Network with a Critic (discrepancy): We explore the impact of replacing the auxiliary classifier in the original D-Network with a Critic that measures discrepancy.

Table 3. Ablation study of B->A (T) of JNU Dataset.

Item	Model Setting	Best Accuracy	Last Accuracy
1	ACGAN(Standard)	76.27	19.28
2	ACGAN + MMD	78.32	16.55
3	ACGAN + Critic	73.03	59.72
4	ACGAN + Critic + MMD	79.86	50.00
5	ACGAN + GRL	84.64	81.74
6	ACGAN + GRL + Critic + MMD	86.00	79.86



(a) ACGAN(Standard). (b) MMD. (c) Critic.

(d) Critic+MMD. (e) GRL. (f) GRL+Critic+MMD.

Fig. 6. Comparing the accuracy of combined ACGAN and sub-modules.

4. Combining MMD loss and Critic in adversarial networks: We study the effect of adding maximum mean difference (MMD) and Critic losses in adversarial networks, aiming to enhance the inter-domain and alignment of the classifiers.
5. Add the effect of adding the Gradient Reversal Layer (GRL) to the G and F networks in the ACGAN adversarial network.
6. The sub-modules such as MMD, Critic, and GRL have been added to the ACGAN architecture to analyze the model's performance in both the source and target domains. This forms the structure of this article.

By analyzing the performance of these variations, we aim to gain insights into how each component influences the model's overall performance. The ablation mentioned above research result reveals that the original ACGAN module already possesses the ability of adaptation alignment, as depicted in Fig. 6(a). However, when processing specific datasets with relatively low accuracy between source and target, Accuracy curves failed to converge, indicating potential overfitting caused by learned features of the F-G network. To address this issue,

We introduce MMD to enhance further the alignment of feature distributions between source and target domains, successfully reducing distribution differences Fig. 6(b). We also promote more effective feature learning by introducing the Critic module to enhance the model's knowledge in the target domain and maximize the difference between classifiers, as shown in Fig. 6(c), we found that the Critic module effectively stabilizes model performance. However, when used alone, its effect on improving accuracy is limited, suggesting that it may be more suitable as an auxiliary to other techniques. Therefore, We tested the combination of Critic and MMD modules, with the results shown in Fig. 6(d). When these two modules are used together, it can be observed that there is a perfect feature alignment process between the source and target domains. Still, after stabilizing, the accuracy does not improve further. This indicates that while these combined techniques can achieve stable learning effects, they may limit the model's ability to learn further, especially when dealing with complex inter-domain differences. For example, certain situations may cause the model to get stuck in a local minimum, limiting further progress in learning. However, the model's performance in both the source and target demonstrates a certain degree of stability. We introduced gradient reversal layers (GRL) into experiments on the original ACGAN model. We found that the model's accuracy in the target domain was significantly improved, as shown in Fig. 6(e). The primary function of GRL is to align the feature distributions of the source and target domains through adversarial training, allowing the model to transfer knowledge between the two domains more effectively, thereby improving the learning outcomes in the target domain. Our experiments demonstrate the effectiveness of this technology in domain adaptation. Finally, we verified our architecture through further experiments. In our study, we used ACGAN in combination with GRL, Critic, and MMD. We observe an increase in accuracy and overall improvement in performance. This combination produced the best results, as shown in Fig. 6(f), with the model demonstrating high accuracy and excellent stability in both the source and target domains. Furthermore, It improves performance in the target domain and reduces distribution differences in the learning process, demonstrating strong domain adaptation capabilities. We observe an increase in accuracy and overall improvement in performance.

5 Conclusions

This paper proposes a novel domain adaptation method for fault diagnosis under different working conditions. Our approach combines GAN and two discrepancy techniques to enhance domain adaptation and improve the generalization and performance of fault diagnosis models. By leveraging the Critic Classifier GAN, we have significantly enhanced diagnostic performance by aligning the source and target domains and facilitating knowledge transfer. Through experiments and evaluations, we have demonstrated the effectiveness and robustness of our proposed method in addressing the challenges of domain shifts in fault diagnosis. Our method achieves state-of-the-art fault classification results compared to

existing approaches. Furthermore, our study sheds light on the influence of data features on fault diagnosis performance. We observe that different data representations, such as the time domain and frequency domain, have varying impacts on the accuracy of fault diagnosis models. This insight can guide future research and aid in selecting appropriate data representations for fault diagnosis tasks. We introduced gradient reversal layers (GRL) into experiments on the original ACGAN model and found that they significantly improved its accuracy in the target domain. The primary function of GRL is to align the feature distributions of the source and target domains through adversarial training, allowing the model to transfer knowledge between the two domains more effectively, thereby improving the learning outcomes in the target domain. Our experiments demonstrate the effectiveness of this technology in domain adaptation.

Acknowledgments. This work was supported in part by the National Science and Technology Council, Taiwan under grants NSTC 111-2221-E-007-106-MY3 and NSTC 112-2634-F-007 002.

References

1. Zhang, W., Li, X., Ma, H., Luo, Z., Li, X.: Universal domain adaptation in fault diagnostics with hybrid weighted deep adversarial learning. IEEE Trans. Industr. Inf. **17**(12), 7957–7967 (2021)
2. Li, X., Zhang, W., Xu, N.-X., Ding, Q.: Deep learning-based machinery fault diagnostics with domain adaptation across sensors at different places. IEEE Trans. Industr. Electron. **67**(8), 6785–6794 (2019)
3. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, pp. 3723–3732 (2018)
4. Huo, C., Jiang, Q., Shen, Y., Lin, X., Zhu, Q., Zhang, Q.: A class-level matching unsupervised transfer learning network for rolling bearing fault diagnosis under various working conditions. Appl. Soft Comput. **146**, 110739 (2023)
5. Borgwardt, K.-M., Gretton, A., Rasch, M.-J., Kriegel, H.-P., Schölkopf, B., Smola, A.-J.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), e49–e57 (2006)
6. Sun, B., Feng, J., Saenko, K.: Return of frustratingly easy domain adaptation. Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA, vol. 30, no. 1 (2016)
7. Ghifary, M., Kleijn, W.-B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction classification networks for unsupervised domain adaptation. In: Computer Vision-ECCV (2016): 14th European Conference, Amsterdam, The Netherlands, pp. 597–613 (2016)
8. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International Conference on Machine Learning. PMLR 37, Lille, France, pp. 1180–1189 (2015)
9. Li, Y., Wang, N., Shi, J., Liu, J., Hou, X.: Revisiting batch normalization for practical domain adaptation. arXiv preprint arXiv:1603.04779 (2016)

10. Sankaranarayanan, S., Balaji, Y., Castillo, C.-D., Chellappa, R.: Generate to adapt: aligning domains using generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, pp. 8503–8512 (2018)
11. Liu, M.-Y., Tuzel, O.: Coupled generative adversarial networks. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
12. Kang, G., Jiang, L., Yang, Y., Hauptmann, A.-G.: Contrastive adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, pp. 4893–4902 (2019)
13. Lin, Z., et al.: Improving maximum classifier discrepancy by considering joint distribution for domain adaptation. In: Hacid, H., Cellary, W., Wang, H., Paik, H.-Y., Zhou, R. (eds.) WISE 2018. LNCS, vol. 11234, pp. 253–268. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02925-8_18
14. Dai, S., Cheng, Y., Zhang, Y., Gan, Z., Liu, J., Carin, L.: Contrastively smoothed class alignment for unsupervised domain adaptation. In: Proceedings of the Asian Conference on Computer Vision (2020)
15. Ganin, Y., et al.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(59), 1–35 (2016)
16. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, Australia, vol. 70, pp. 2642–2651 (2017)
17. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
18. Odena, A.: Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583 (2016)
19. Loparo, K.: CWRU, Case western reserve University bearing data center, Seeded Fault Test Data. https://engineering.case.edu/bearingdatacenter/download-data-file. Accessed 12 Mar 2024
20. Jia, F., Lei, Y., Lin, J., Zhou, X., Lu, N.: Deep neural networks: a promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. Mech. Syst. Signal Process. **72**, 303–315 (2016)
21. Zhang, W., Peng, G., Li, C., Chen, Y., Zhang, Z.: A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. Sensors **17**(2), 425 (2017)
22. Zhang, B., Li, W., Hao, J., Li, X.-L., Zhang M.: Adversarial adaptive 1-D convolutional neural networks for bearing fault diagnosis under varying working condition. arXiv preprint arXiv:1805.00778 (2018)
23. Li, K., Ping, P., Wang, H., Chen, P., Cao, Y.: Sequential fuzzy diagnosis method for motor roller bearing in variable operating conditions based on vibration analysis. Sensors **13**(6), 8013–8041 (2013)
24. Han, T., Liu, C., Yang, W., Jiang, D.: Learning transferable features in deep convolutional neural networks for diagnosing unseen machine conditions. ISA Trans. **93**, 341–353 (2019)
25. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: International Conference on Machine Learning. PMLR, Lille, France, vol. 37, pp. 97–105 (2015)
26. Long, M., Zhu, H., Wang, J., Jordan, M.-I.: Deep transfer learning with joint adaptation networks. In: International Conference on Machine Learning. PMLR, Sydney, Australia, vol. 70, pp. 2208–2217 (2017)

27. Sun, B., Saenko, K.: Deep CORAL: correlation alignment for deep domain adaptation. In: Hua, G., Jégou, H. (eds.) ECCV 2016. LNCS, vol. 9915, pp. 443–450. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_35
28. Long, M., Cao, Z., Wang, J.: Conditional adversarial domain adaptation. In: Advances in Neural Information Processing Systems, vol. 31 (2018)

Data Augmentation Pipeline for Enhanced UAV Surveillance

Solmaz Arezoomandan(✉), John Klohoker, and David K. Han

Drexel University, Philadelphia, PA 19104, USA
{sa3747,dkh42}@drexel.edu, jfk82@dragons.drexel.edu

Abstract. The growing use of Unmanned Aerial Vehicles (UAVs) presents considerable safety and security challenges, requiring improved capabilities for drone detection. On the other hand, collecting real-world data is costly, time consuming, and in some cases subject to rules and regulations. This paper addresses this issue by introducing a novel two-phase data augmentation approach aimed at improving the accuracy and efficiency of long-range drone detection systems. The initial phase involves the generating of synthetic data, using Unreal Engine, to increase the diversity and richness of the dataset. Subsequently, we employ Cycle-consistent Generative Adversarial Network (CycleGAN) to translate these synthetic images into more realistic representations, thus bridging the gap between synthetic and real datasets. This methodology not only seeks to refine the precision of drone detection algorithms but also presents a cost-effective solution for creating extensive and varied drone detection datasets. To evaluate the efficacy of our proposed data augmentation pipeline, we trained the You Only Look Once (YOLO) detection model under three distinct scenarios: utilizing purely real data, augmenting real data with synthetic data, and augmenting real data with CycleGAN-translated synthetic data. The performance of these models was assessed using four separate, unseen real datasets: DetFly, Drone Detection, UAV Detect, and New Batch. Our findings indicate a marked improvement in drone detection capabilities when the training dataset includes both synthetic and translated images, with the most significant enhancement observed in scenarios where real data is augmented with translated data.

Keywords: Drone detection · CycleGAN · Unreal Engine · Data augmentation · AirSim

1 Introduction

The rapidly growing use of Unmanned Aerial Vehicles (UAVs) highlights the critical need for accurate drone detection models. As drones have become increasingly prevalent in various sectors, including disaster response, infrastructure

This research was funded by the Federal Aviation Administration (FAA) through a grant awarded to ASSURE's Five Research Universities under the FAA Center of Excellence Grants Program, UAS Cooperative Agreement 15-C-UAS-DU-010.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 366–380, 2025.
https://doi.org/10.1007/978-3-031-78172-8_24

inspection, and law enforcement, there is a clear and urgent demand for the development and improvement of drone detection algorithms. This is essential to ensure safety, security, and compliance with evolving regulatory frameworks.

This extensive use of drones presents notable risks to safety and security. Unauthorized drone flights have the potential to breach restricted airspace. These intrusions pose substantial threats to air traffic safety, with near-collisions between commercial aircraft and drones becoming increasingly common, as illustrated by a recent incident involving a passenger plane and a drone in Chicago [12]. Moreover, the versatility of drones presents opportunities for malicious exploitation, such as contraband smuggling, unauthorized surveillance [6], or even cyber-attacks. Detecting drones accurately and efficiently is therefore of paramount importance, safeguarding not only critical infrastructure but also privacy and security.

In response to these complexities and the dynamic evolution of drone applications, this paper addresses the imperative need for robust, adaptive drone detection models. Acknowledging the limitations of conventional data gathering methods for drone detection—marked by high costs, temporal constraints, and access issues—this paper introduces a novel, two-phased approach aimed at enhancing the efficacy and reliability of detection algorithms. By integrating synthetic data with real-world scenarios and employing domain adaptation using CycleGAN [42] for realistic image translation, this method not only promises improvement in the accuracy of drone detection but also presents a cost-effective strategy for creating diverse and representative datasets. The main contributions of this paper are as follows:

- We develop a comprehensive augmentation pipeline that incorporates synthetic data generation, utilizing Unreal Engine and AirSim, and employs CycleGAN for domain transfer, thereby enriching real datasets with high fidelity.
- By adapting CycleGAN for the nuanced task of domain adaptation, we significantly bolster the realism of simulated images, thereby improving the drone detection process.

2 Related Works

When real datasets are scarce or difficult to access, synthetic data proves to be an invaluable alternative, offering a solution to bridge the gap for machine learning model development. Various methods exist for generating such datasets, with computer graphics software and game engines like Unreal Engine [13,31] and Unity [18] at the forefront. These tools are instrumental in producing realistic imagery within dynamic settings, enabling the creation of synthetic datasets that closely resemble real-world conditions. This capability is crucial for the comprehensive testing and refining of machine learning algorithms, particularly in fields where real data is either not available or ethically sensitive to use.

In [16] a wide range of UAV simulators with different functionalities are studied and categorized into two groups of network and physical simulators.

Physics simulators such as AirSim [34] and MATLAB UAV toolbox [21] provide environments of high fidelity suitable for applications in Artificial Intelligence and machine learning. Authors in [28] aim to create a simulated dataset using the quadcopter-based flight control mode in MATLAB simulink. The simulated dataset is modified to obtain drone videos to detect object categories such as pedestrians, other drones, and obstacles while navigating in a simulated environment.

Flightmare [37] is a quadrotor simulator which comprises of two primary elements: a customizable rendering engine developed using Unity and an adaptable physics engine designed for dynamic simulation.

In another method to generate synthetic dataset a hybrid approach can be adopted by incorporating real backgrounds and placing computer-generated drone images within them [26].

The concept of domain randomization introduces randomness into synthetic data, enhancing the robustness of machine learning models against real-world variability. Barisic et al. [3] propose a pipeline to generate a synthetic aerial dataset by creating models and textures for drones in Blender software [14] and placing them in real-world background images.

Domain adaptation serves as a versatile tool for both adapting models to new domains and generating synthetic datasets. Generative models, particularly Generative Adversarial Networks (GANs) [17], play a key role in tailoring synthetic datasets to closely mimic the unique characteristics and variations present in the real domain.

Conditional Generative Adversarial Networks (CGANs) [23,27,39], form a distinct class of generative models, notable for their ability to incorporate conditional input, such as class labels or images. This enables precise and controlled synthesis of realistic data, offering a valuable approach for domain adaptation.

An alternative strategy involves domain adaptation in scenarios where paired datasets are unavailable. Notably, CycleGAN stands out as a widely recognized unsupervised image-to-image translation method. The concept of cycle-consistent adversarial learning, introduced in CycleGAN, has been employed in various studies for translating images from synthetic to real domains [30,35]. Additionally, CycleGAN with a multiscale attention module has been applied to enhance the quality of synthetically generated images [26].

Some unsupervised methods leverage segmentation labels from the synthetic domain to enforce consistency of relevant semantics before and after translation [20,25,29]. The authors in [32] employed CycleGAN to translate contrast-enhanced CT images into non-contrast images for data augmentation, significantly enhancing the generalizability of CT segmentation models. In [19] CycleGAN is used to translate MRI to CT images and then augmenting the dataset for multi-organ detection using YOLO. Similarly, [5] employed CycleGAN to generate petrochemical pipeline defect images to overcome the inbalance in the dataset. In a novel approach, authors in [38] incorporated a vision transformer into the generator of CycleGAN to enhance its overall performance.

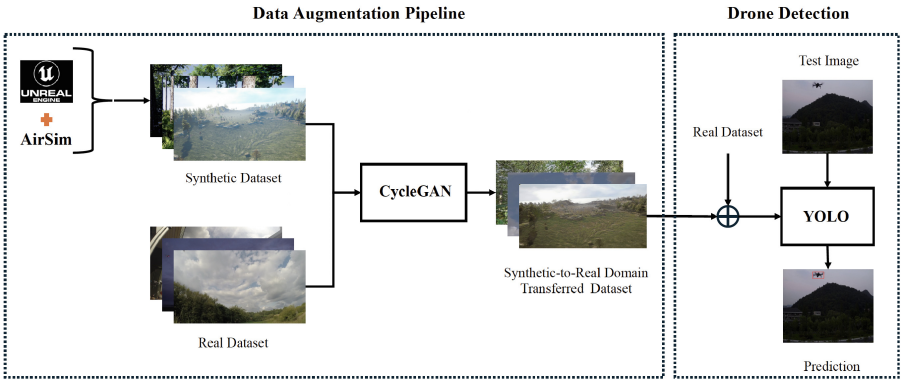


Fig. 1. The proposed framework: data augmentation pipeline consisting of synthetic data generation and domain adaptation using CycleGAN (left) and YOLO trained on real data and domain transferred synthetic data for drone detection (right).

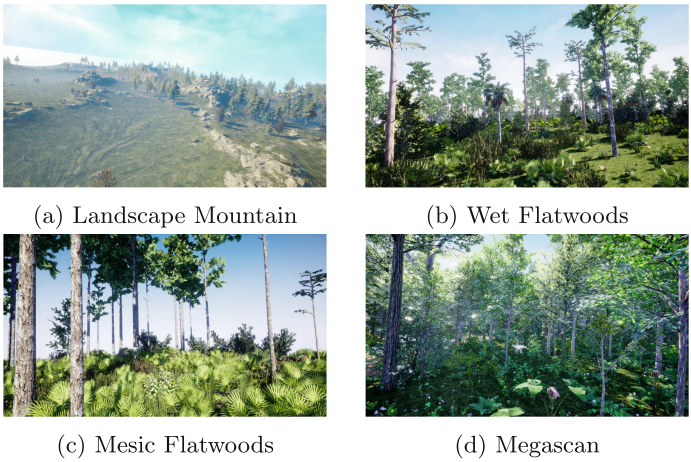
3 Proposed Methodology

In our work, we generate a drone synthetic dataset using Unreal Engine and AirSim [34] to closely replicate real-world dynamics and scenarios relevant to drone detection. We then employ domain adaptation technique using CycleGAN to bridge the gap between synthetic and real-world data. This allows us to enrich a real dataset with synthetic data that has been translated by the GAN, ensuring a closer alignment with real-world conditions. This strategy is designed to enhance the performance of YOLO algorithm [24] for drone detection. The methodology and workflow of our approach are depicted in Fig. 1.

3.1 Synthetic Data Generation

Machine learning algorithms thrive on data, using it to recognize patterns, make predictions, and learn from experiences. However, acquiring and preparing high-quality, diverse, and representative datasets can be a challenging and resource-intensive task. This is where synthetic data comes into play, serving as a valuable supplement or even an alternative to traditional datasets.

In this paper, we leveraged synthetic data to augment a real dataset, Drone vs. Bird [7], with the goal of enhancing the performance of YOLO model for detecting drones over extended ranges. Here, Unreal Engine and AirSim were employed to simulate a synthetic environment. We selected AirSim for this study due to its capability to gather data for machine learning algorithms and its compatibility with Unreal Engine (UE). Additionally, AirSim facilitates seamless integration with Python, enabling efficient automation of data collection.



(a) Landscape Mountain (b) Wet Flatwoods

(c) Mesic Flatwoods (d) Megascan

Fig. 2. Image samples generated using Unreal Engine across four different environments.

In this research, Unreal Engine (UE) was utilized to generate 3D settings that serve as backdrops for drone simulations. Specifically, four UE environments were employed: Landscape Mountain, Wet Flatwoods, Mesic Flatwoods, and Megascan Forest. These environments highlight forest landscapes that closely resemble the natural settings observed in the Drone vs. Bird dataset. Each of the four environments possess distinct plant life from diverse climates, with varying levels of foliage density. In Fig. 2, we can see sample images generated in each environment.

3.2 Domain Adaptation

Domain adaptation is a technique in machine learning and computer vision aimed at improving the deployment of models trained on a dataset from one domain (source) to another with different characteristics (target). This research focuses on translating images from a synthetic environment to real-world images, with the goal of reducing the "reality gap" and augmenting the dataset used for training detection models. The challenge lies in the differences between synthetic and real-world images, which can hinder the performance of models trained on synthetic data when applied in real-world scenarios. This research tackles the issue as an unsupervised domain adaptation problem and attempts to bridge this gap without the need for directly corresponding paired data.

In this study, we employed CycleGAN for domain adaptation due to its versatility as a general domain adaptation method, suitable for translating between any two related datasets. Additionally, it operates in an unsupervised manner, eliminating the need for paired datasets, which aligns with the requirements of our case. The algorithm employs two Generative Adversarial Networks (GANs), one for each domain, which are trained simultaneously. The generators aim to

transform images from one source domain to another target domain, while the discriminators strive to distinguish between real and generated images.

3.3 Drone Detection

Object detection is a computer vision task focused on identifying and localizing objects within an image or video. Object detection aims to recognize and locate distinct objects, providing both their class labels and bounding box coordinates. For our purpose, we chose YOLOv8, provided by Ultralytics [24] for detecting drones.

For evaluating the performance of drone detection models, two metrics were employed. Mean Average Precision at IoU 0.5 (mAP50) which calculates the accuracy of predicted bounding boxes against ground truth bounding boxes, with an IoU threshold set at 0.5. Similarly, Mean Average Precision from IoU 0.5 to 0.95 (mAP50-95) evaluates precision across a range of IoU thresholds with 0.05 increments. For each detected object, precision is computed at IoU values from 0.5 to 0.95. The average precision is then calculated for each specific IoU threshold. The final mAP50-95 is obtained by averaging these precision values over the entire IoU range, offering a nuanced assessment across various IoU thresholds.

4 Experiments and Results

In this section we demonstrate the effectiveness of the proposed pipeline in augmenting the real dataset and improving the performance of detection models. All the experiments were conducted on one NVIDIA GeForce RTX 3090 GPU.

4.1 Datasets

In the conducted experiments, we employed two datasets: a real dataset and a synthetic dataset. The synthetic dataset, generated by the Unreal Engine and AirSim, provided a controlled environment for generating various scenarios, as explained in Sect. 3.1.

Real Dataset. For our research, we selected the "Drone vs. Bird" (DvB) dataset, which consists of 77 unique videos totaling 103,124 frames. This dataset is characterized by its diversity, featuring videos of different resolutions that showcase eight types of drones—three with fixed wings and five with rotary wings. The dataset presents a wide range of challenges, including varied backgrounds like sky and vegetation, different weather conditions (e.g., cloudy, sunny), direct sunlight interference, and variations in camera quality. Figure 3 highlights the dataset's variability.

Fig. 3. Sample frames from Drone vs. Bird dataset [7].

Additionally, the dataset captures a significant variance in the distance between drones and the camera, affecting the apparent size of drones in the videos. This variance is evident both within and across videos, leading to a broad spectrum of drone sizes, from as small as 15 pixels to over 1,000,000 pixels. Despite this range, most drones are relatively small, typically measuring less than 16×16 pixels or between 16×16 to 32×32 pixels [8]. The predominance of these smaller drones underscores the dataset's relevance to our study's goal which is improving the detection of drones at long distances.

Notably, the dataset also includes footage of birds, which are not individually annotated and serve as the primary source of interference, further complicating drone detection. This combination of factors makes the "Drone vs. Bird" dataset a suitable target dataset for developing and testing drone detection technologies under a variety of challenging conditions.

Synthetic Dataset. A synthetic dataset of drone images in the four vegetated environments was generated using the method explained in Sect. 3.1. In our experiments, we use the term UE, which stands for Unreal Engine, to refer to this synthetic dataset. The resulting 10,000-image Unreal Engine dataset contains 500 images for five drone models in four different environments. One of the environments, Megascan Forest, has a higher foliage density, resulting in random distances between 4 and 9 unreal units, while the other environments have a range of 7 to 17 unreal units. Each image has random roll, pitch, and yaw, with the observing drone's pitch and yaw set between -15 and $15°$ and roll between -5 and $5°$ [2]. To simulate the distribution of real-world dataset, this random rotation positions the target drone in the middle 40% of the x-axis and the middle 50% of the y-axis of the image [22] (see Fig. 2).

Fig. 4. Synthetic images (top row) alongside their corresponding translations into the real domain (bottom row).

4.2 Domain Adaptation

We trained the CycleGAN with two datasets: UE as the synthetic dataset, and a real dataset to translate synthetic images into the real domain. The real dataset was a combination of various datasets, including Drone vs. Bird, Rotor Drone [9], Lacmus Drone Dataset (LaDD) [36], Drone Dataset [1], Drone Dtection [11], and Fixed Wing Drone Detection [4]. For these datasets, we selectively utilized images that closely matched our synthetic data. In total, 6500 synthetic images and 5500 real images were utilized for training the CycleGAN. We adopted the hyperparameters outlined in [42], with specific adjustments such as a load size of 1080 and a crop size of 360 to generate high-resolution images. In our experiments we refer to this translated dataset as UE_CycleGAN. Figure 4 shows some of synthetic images alongside their translated counterparts.

4.3 Drone Detection

Due to the size of our datasets and the associated memory limitations, we used YOLOv8 medium version (YOLOv8m). This model served as our drone detection solution, providing a balance between accuracy and computational efficiency.

The Drone vs. Bird (DvB) dataset was divided into two distinct sets, ensuring that images from the same source video were grouped together. The first set designated approximately 50% of the images for training purposes, labeled as DvB50. The second set utilized about 80% of the images as input data, termed DvB80. Consequently, our drone detection model was applied to six distinct types of datasets, namely DvB50, DvB50+UE, DvB50+UE_CycleGAN, DvB80, DvB80+UE, and DvB80+UE_CycleGAN.

In all our experiments, we evaluated the models using the reserved segment of the Drone vs. Bird (DvB) dataset, which was not utilized during the training phase. This approach ensured that the models were assessed on data they had not previously encountered. Subsequently, the trained models underwent testing on four real, unseen datasets: DetFly [41], UAV Detect [15], Drone Detection

Table 1. Inference results of trained models on DetFly, Drone Detection, UAV Detect, and New Batch datasets.

Trained Models	DetFly		Drone Detection		UAV Detect		New Batch	
	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95
DvB50	0.462	0.22	**0.66**	**0.282**	0.58	0.288	0.3825	0.18
DvB50+UE	0.475	0.24	0.66	0.282	0.574	0.306	0.3904	**0.2087**
DvB50+UE_CycleGAN	**0.524**	**0.262**	0.614	0.263	**0.601**	**0.334**	**0.3954**	0.1957
DvB80	0.461	0.226	0.645	0.292	0.61	0.328	0.3774	0.2237
DvB80+UE	0.487	0.238	0.677	0.3	0.575	0.304	0.41	0.2161
DvB80+UE_CycleGAN	**0.51**	**0.246**	**0.694**	**0.311**	**0.612**	**0.352**	**0.4173**	**0.2455**

[10], and New Batch [33]. This approach allows for assessment of the drone detection model's generalization across various dataset compositions and real-world scenarios.

The DetFly dataset consists of 7,346 images of drones, mostly captured from long distances, featuring a wide range of backgrounds and lighting conditions. The Drone Detection dataset includes 1,793 images of drones at both close and long ranges, with varied backgrounds. The UAV Detect dataset offers 4,409 images of drones at different distances, with diverse backgrounds and lighting conditions; some of these images were taken with a GoPro lens. In the New Batch dataset, the training set contained images augmented by rotation, flipping, and shearing. The validation set of New Batch is composed of images without the image augmentation. To cater to real-world scenarios, our models were tested on the validation set only.

This validation set contains 1,317 images of drones at long distances with sparsely vegetated backgrounds. It also provides a diverse range of angles, covering bird's eye to ground level views and eye-level perspectives.

4.4 Discussions and Limitations

As illustrated in Table 1, the introduction of synthetic data from Unreal Engine generally enhances performance compared to using only real data. The incorporation of CycleGAN on top of Unreal Engine data further elevates performance, with these models achieving the highest mAP50 and mAP50-95 values in most cases.

However, there is discernible variability in performance across datasets, underscoring the influence of dataset characteristics on model efficacy. For DetFly, the improvement is consistently observed across metrics, indicating that both introducing synthetic data and translated data contributes to enhanced model performance.

In the UAV Detect case, we notice higher values for both metrics, mAP50 and mAP50-95, in the DvB80 model compared to the DvB50 model. This discrepancy is attributed to the composition of the training sets. The DvB80 training set includes more images captured with a GoPro lens. As the UAV Detect dataset also contains this type of imagery, we observe an improvement in our metrics.

Despite the DvB50 model having fewer training samples, the introduction of translated data into the training set serves to compensate for the shortage of real data.

Furthermore, when investigating the impact of augmenting real datasets—DvB50 and DvB80—in two specific instances, DetFly and UAV Detect, it becomes apparent that DvB50+UE_CycleGAN models exhibit greater improvement compared to DvB80+UE_CycleGAN models. This observation suggests that augmenting the smaller subset of the Drone vs. Bird dataset for these datasets leads to enhanced generalization of our models.

In the case of the Drone Detection dataset, utilizing DvB50 as the real dataset for augmentation negatively affects model performance. This can be attributed to substantial differences in backgrounds between the Drone Detection and Drone vs. Bird datasets, particularly with most images capturing drones at close range. Additionally, the presence of horizontally rotated images, a characteristic absent in the Drone vs. Bird dataset, may contribute to the observed performance decline. Nevertheless, when expanding the size of our real dataset to DvB80, models trained on augmented datasets exhibit improved performance. Another factor contributing to the distinct behavior of Drone Detection compared to the other two datasets is the dataset size, as Drone Detection is considerably smaller than DetFly and UAV Detect datasets.

The detection performance on New Batch dataset, confirms the results driven from other test sets. The introduction of synthetic and translated images generally improves the detection performance. The model DvB80+UE_CycleGAN achieved the highest performance for both mAP50 and mAP50-95 on this dataset.

Across all images, it is evident that dataset augmentation plays a crucial role in enabling the models to rectify detection errors amidst diverse backgrounds. In Figs. 5 and 6 we can see the effect of introducing the translated images in the real datasets on drone detection.

The primary purpose of this research is to evaluate the effectiveness of our method in enhancing the detection model's generalization ability. The CycleGAN we developed is specifically designed to transform synthetic images generated by UE to resemble the DvB dataset. To test the generalizability of the proposed approach, the selected test sets differ from the DvB dataset in terms of backgrounds, lighting conditions, drone models, and viewing angles. We demonstrated that, despite these differences, our augmentation method effectively enhances detection performance.

For practical implementation in a specific environmental scenario, the datasets should be adjusted to cover either the specific environment where drone detection is performed or include a variety of environmental scenarios. By training the model on data that closely resembles the target environment, it becomes more adept at recognizing and adapting to the specific challenges and variations present in that environment, leading to improved performance and accuracy in real-world drone detection scenarios.

Fig. 5. Inference results of DvB50 (left column) and DvB50+UE_CycleGAN (right column) on images from DetFly, UAV Detect, and Drone Detection.

Moreover, using a variety of environmental scenarios in training data can significantly enhance the model's generalization ability. Exposure to diverse backgrounds, lighting conditions, and viewing angles during training allows the model to learn and adapt to different situations. This broadens the model's capability to detect drones in varied and unpredictable real-world conditions, thereby improving its robustness and effectiveness across multiple environments.

Fig. 6. Inference results of DvB80 (left column) and DvB80+UE_CycleGAN (right column) on images from DetFly, UAV Detect, and Drone Detection.

5 Conclusions

This study delves into the crucial role of data augmentation in enhancing the performance of detection algorithms, with a specific focus on the intricate task of long-range drone detection. Given the substantial need for training data in this domain, the research places a central emphasis on augmenting datasets for UAV detection. It introduces a novel pipeline that proposes the generation of simulated images covering various aspects and their translation into the real domain. This approach streamlines the annotation process by automating tasks

such as image segmentation and bounding box labeling, reducing labor intensity and cost.

In the initial phase of our pipeline, the research examines the impact of simulated images on algorithmic performance, highlighting the potential of simulation to address challenges inherent in UAV detection. The study effectively demonstrates the utility of synthetic data generated through Unreal Engine as valuable training material. Notably, the findings advocate for the integration of synthetic data as a complementary resource to real-world data, enriching the development of more accurate and resilient datasets for UAV detection.

In the subsequent phase of the pipeline, the study explores narrowing the reality gap between synthetic and real datasets and evaluates its effect on the performance of drone detection models. The use of CycleGAN successfully reduces the distribution gap, yielding visually appealing results and the addition of translated images to the dataset further enhances drone detection.

The insights derived from this research have implications for advancing UAV detection algorithms. Leveraging synthetic and translated data proves effective in overcoming challenges related to acquiring extensive real-world training data. The proposed pipeline for data augmentation enables researchers to compile comprehensive and diverse datasets, ultimately improving the accuracy and resilience of detection algorithms.

One promising direction for future research in image translation involves exploring the Diffusion models [40], which have shown compelling results in generating realistic high-resolution synthetic images.

References

1. Project 986i8: Drone dataset (2023). https://universe.roboflow.com/project-986i8/drone-uskpc. Accessed 25 Nov 2023
2. Arezoomandan, S., Klohoker, J., Han, D.K.: Analyzing the efficacy of synthetic images in unmanned aerial vehicle detection. In: 2024 IEEE International Conference on Consumer Electronics (ICCE), pp. 1–6 (2024). https://doi.org/10.1109/ICCE59016.2024.10444259
3. Barisic, A., Petric, F., Bogdan, S.: Sim2air-synthetic aerial dataset for UAV monitoring. IEEE Robot. Autom. Lett. **7**(2), 3757–3764 (2022)
4. Celikkol, B.: Fixed wing dataset (2022). https://www.kaggle.com/datasets/burhanclkkl/fixed-wing-dataset
5. Chen, K., et al.: An automatic defect detection system for petrochemical pipeline based on cycle-GAN and YOLO v5. Sensors **22**(20), 7907 (2022)
6. CNN: Drones pose a risk to critical infrastructure (2022). https://www.cnn.com/2022/09/30/politics/drones-risk-critical-infrastructure-spotted-louisiana-chemical-facilities/index.html
7. Coluccia, A., et al.: Drone vs. bird detection: deep learning algorithms and results from a grand challenge. Sensors **21**(8), 2824 (2021)
8. Coluccia, A., et al.: Drone vs. bird detection: deep learning algorithms and results from a grand challenge. Sensors **21**(8) (2021). https://doi.org/10.3390/s21082824. https://www.mdpi.com/1424-8220/21/8/2824

9. Detect company2: rotor drone dataset (2023). https://universe.roboflow.com/detect-company2/rotor-drone. Accessed 25 Nov 2023
10. Drone detection dataset (2023). https://universe.roboflow.com/drone-detection-7usmm/drone-detection-w29we. Accessed 14 Oct 2023
11. DL: Drone-detection dataset (2023). https://universe.roboflow.com/dl-wzwpj/drone_detection-jcchu. Accessed 25 Nov 2023
12. DroneDJ: Envoy air passenger plane hits drone after Chicago takeoff (2021). https://dronedj.com/2021/08/24/envoy-air-passenger-plane-hits-drone-after-chicago-takeoff/
13. Epic Games: Unreal Engine 4.27 (2021). Accessed Oct 2023
14. Flavell, L.: Beginning blender: open source 3D modeling, animation, and game design. Apress (2011)
15. GET: UAV detect dataset (2023). https://universe.roboflow.com/get/uav-detect-pfiqs. https://universe.roboflow.com/get/uav-detect-pfiqs. Accessed 15 Oct 2023
16. Gill, J.S., Velashani, M.S., Wolf, J., Kenney, J., Manesh, M.R., Kaabouch, N.: Simulation testbeds and frameworks for UAV performance evaluation. In: 2021 IEEE International Conference On Electro Information Technology (EIT), pp. 335–341. IEEE (2021)
17. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
18. Haas, J.K.: A history of the unity game engine. Diss. Worcester Polytech. Inst. **483**(2014), 484 (2014)
19. Hammami, M., Friboulet, D., Kechichian, R.: Cycle GAN-based data augmentation for multi-organ detection in CT images via yolo. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 390–393 (2020). https://doi.org/10.1109/ICIP40778.2020.9191127
20. Hoffman, J., et al.: Cycada: cycle-consistent adversarial domain adaptation. In: International Conference on Machine Learning, pp. 1989–1998. PMLR (2018)
21. Horri, N., Pietraszko, M.: A tutorial and review on flight control co-simulation using matlab/simulink and flight simulators. Automation **3**(3), 486–510 (2022)
22. Isaac-Medina, B.K., Poyser, M., Organisciak, D., Willcocks, C.G., Breckon, T.P., Shum, H.P.: Unmanned aerial vehicle visual detection and tracking using deep neural networks: a performance benchmark. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1223–1232 (2021)
23. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 (2017)
24. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics yolov8 (2023). https://github.com/ultralytics/ultralytics
25. Li, P., Liang, X., Jia, D., Xing, E.P.: Semantic-aware grad-GAN for virtual-to-real urban scene adaption. arXiv preprint arXiv:1801.01726 (2018)
26. Liu, W., Luo, B., Liu, J.: Synthetic data augmentation using multiscale attention cyclegan for aircraft detection in remote sensing images. IEEE Geosci. Remote Sens. Lett. **19**, 1–5 (2021)
27. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
28. Mittal, P., Sharma, A., Singh, R.: A simulated dataset in aerial images using simulink for object detection and recognition. Int. J. Cogn. Comput. Eng. **3**, 144–151 (2022)

29. Mütze, A., Rottmann, M., Gottschalk, H.: Semi-supervised domain adaptation with cyclegan guided by a downstream task loss. arXiv preprint arXiv:2208.08815 (2022)
30. Oprea, S., et al.: H-GAN: the power of GANs in your hands. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2021)
31. Sanders, A.: An introduction to Unreal engine 4. AK Peters/CRC Press (2016)
32. Sandfort, V., Yan, K., Pickhardt, P., et al.: Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in CT segmentation tasks. Sci. Rep. **9**, 16884 (2019). https://doi.org/10.1038/s41598-019-52737-x
33. School: new_batch dataset (2024). https://universe.roboflow.com/school-1kgrs/new_batch. Accessed 08 July 2024
34. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: high-fidelity visual and physical simulation for autonomous vehicles (2017)
35. Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., Webb, R.: Learning from simulated and unsupervised images through adversarial training. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2107–2116 (2017)
36. Shuranov, M., Shurenkov, D., Ruzhitsky, D., Martynova, V., Bykova, E., Perevozchikov, G.: LADD: lacmus drone dataset (2023). https://www.kaggle.com/datasets/mersico/lacmus-drone-dataset-ladd-v40
37. Song, Y., Naji, S., Kaufmann, E., Loquercio, A., Scaramuzza, D.: Flightmare: a flexible quadrotor simulator. In: Conference on Robot Learning, pp. 1147–1157. PMLR (2021)
38. Torbunov, D., et al.: Uvcgan: Unet vision transformer cycle-consistent GAN for unpaired image-to-image translation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 702–712 (2023)
39. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional GANs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8798–8807 (2018)
40. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3836–3847 (2023)
41. Zheng, Y., Chen, Z., Lv, D., Li, Z., Lan, Z., Zhao, S.: Air-to-air visual detection of micro-UAVs: an experimental evaluation of deep learning. IEEE Robot. Autom. Lett. **6**(2), 1020–1027 (2021)
42. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV) (2017)

Generative Adversarial Networks for Imputing Sparse Learning Performance

Liang Zhang[1,2(✉)], Mohammed Yeasin[1,2], Jionghao Lin[3,4], Felix Havugimana[2], and Xiangen Hu[1,2,5]

[1] Institute for Intelligent Systems, University of Memphis, Memphis, TN 38152, USA
{lzhang13,myeasin}@memphis.edu, xiangen.hu@polyu.edu.hk
[2] Department of Electrical and Computer Engineering, University of Memphis, Memphis, TN 38152, USA
fhvgmana@memphis.edu
[3] Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
jionghal@cs.cmu.edu
[4] Centre for Learning Analytics, Monash University, Melbourne, VIC 3800, Australia
[5] Department of Applied Social Sciences, Hong Kong Polytechnic University, Hong Kong, PR, China

Abstract. Learning performance data, such as correct or incorrect responses to questions in Intelligent Tutoring Systems (ITSs) is crucial for tracking and assessing the learners' progress and mastery of knowledge. However, the issue of data sparsity, characterized by unexplored questions and missing attempts, hampers accurate assessment and the provision of tailored, personalized instruction within ITSs. This paper proposes using the Generative Adversarial Imputation Networks (GAIN) framework to impute sparse learning performance data, reconstructed into a three-dimensional (3D) tensor representation across the dimensions of learners, questions and attempts. Our customized GAIN-based method computational process imputes sparse data in a 3D tensor space, significantly enhanced by convolutional neural networks for its input and output layers. This adaptation also includes the use of a least squares loss function for optimization and aligns the shapes of the input and output with the dimensions of the questions-attempts matrices along the learners' dimension. Through extensive experiments on six datasets from various ITSs, including AutoTutor, ASSISTments and MATHia, we demonstrate that the GAIN approach generally outperforms existing methods such as tensor factorization and other generative adversarial network (GAN) based approaches in terms of imputation accuracy. This finding enhances comprehensive learning data modeling and analytics in AI-based education.

Keywords: Learning Performance Data · Data Imputation · Generative Adversarial Imputation Networks · Generative Artificial Intelligence Model · Intelligent Tutoring System

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 381–396, 2025.
https://doi.org/10.1007/978-3-031-78172-8_25

1 Introduction

The learning performance data, recorded during ITS interactions, documents the sequence of questions-answering activities, tracking the identifications of questions and cataloging the attempts and performance responses made by different learners. However, real-world learning performance data is often incomplete and sparse, with unexplored questions and limited attempts posing challenges for data analysis and modeling. Multiple reasons contribute to data sparsity, including participant's dropout from learning tasks [1], learner disengagement due to off-task behavior [2], random data loss from design and operational errors [3], biases within sample groups [4], among others. This sparsity hinders a comprehensive understanding and assessment of learning performance. Such limitations hinder AI-powered educational systems from effectively delivering educational content, especially in their capabilities to track learning processes, monitor advancement, and gauge learners' mastery of knowledge through their performance data. Thus, accurately imputing sparse learning performance data is critical for advancing learning analytics and modeling, which facilitates the comprehensive exploration of learning insights and ultimately enhancing learner progress within ITSs.

Although traditional data imputation methods (e.g., indicator or mean imputation [5,6], regression imputation [7], and multiple imputation [8]) have proven effective in the literature, they provide a cost-effective solution that avoids labor-intensive experiments and leverages observed data to estimate unobserved data, capitalizing on underlying patterns and characteristics [9]. Indicator or mean imputation may introduce bias by oversimplifying missing data complexities [5,6]. Regression imputation often fails to capture the full spectrum of the underlying data structure [7]. Multiple imputation may not adequately address complex, high-dimensional correlations [10]. Recently, generative AI models, specifically Generative Adversarial Networks (GANs) [11], have demonstrated remarkable success in handling data sparsity through the reconstruction mechanism [12,13], achieving higher accuracy and effectively addressing those issues in traditional data imputation methods. A notable GAN-based model, Generative Adversarial Imputation Nets (GAIN) demonstrated its effectiveness on imputing human health data [12]. The GAIN model extends GAN structure by conditioning the generator on observed data and using a hint mechanism to enhance the discriminator's accuracy in identifying missing data patterns [12,14]. Further research has shown GAIN's superior performance in diverse datasets and applications, from healthcare to machine health monitoring, validating its effectiveness over traditional methods like MICE and missForest [12,13,15].

Despite its impressive imputation performance in prior studies, GAIN's potential for imputing missing data in sparse learning performance datasets within ITSs remains unexplored. The complex nature of learning performance data, characterized by individual learners, questions, and attempts, presents significant challenges for generative models in data imputation. These challenges include achieving higher accuracy based on existing data distribution and handling the complexities of data interactions and variations across different

attempts. Therefore, how can we effectively represent learning performance data to ensure compatibility with the GAIN framework? Additionally, what modifications are necessary to facilitate accurate predictions through specialized computations and algorithms tailored for learning performance scenarios, considering the stability of the models amid dynamic changes in learning performance data?

In response to these challenges in learning performance data, our study aims to perform the data imputation for the sparse learning performance data using the GAIN framework, enriched with detailed revisions. We are guided by the following two **R**esearch **Q**uestions:

- **RQ 1**: How effectively does the GAIN-based method impute sparse learning performance data in ITSs compared to established baseline methods?
- **RQ 2**: How does the stability of the GAIN-based model's performance vary with changes in the number of attempts influencing the sparsity levels of learning performance data?

The generative AI model, GAIN, was leveraged to impute the sparse learning performance data, with an additional focus on exploring the stability of GAIN. Therefore, this study's contributions are twofold:

- It enhances the accuracy of imputing learning performance data, thereby enriching data representation for more detailed analytics and modeling.
- The findings are expected to provide valuable imputation methods for comprehensively tracing and assessing learners' progress within AI-based educational systems like ITS.

2 Related Work

2.1 Addressing Data Sparsity in ITSs

In AI-based education, many studies have focused on tackling data sparsity in sparse learning performance in ITSs. Chen et al. [16] employed the prerequisite concept map for knowledge tracing to mitigate data sparsity. Pandey et al. [17] developed the self-attentive mechanism to predict the learner's performance on unanswered questions by analyzing the relevance previously answered questions. Wang et al. [18] integrated question-knowledge hierarchies into a deep learning framework to improve predictions despite data sparsity. Despite these advances, challenges persist: (1) high demands for expert effort in mapping and annotating knowledge concepts [19], (2) ignorance of temporal learning dynamics [20], and (3) disruption of sequential learning effects even with some methodological recognition [21,22].

In addressing data sparsity in ITSs, tensor factorization has also played a pivotal role by leveraging multidimensional relationships to enhance prediction accuracy and knowledge representation. This approach has evolved from simple matrix factorization to sophisticated multi-dimensional frameworks that incorporate temporal effects and sequential learning dynamics, significantly improving the understanding and prediction of learner performance [20,23,24]. Such

advancements in tensor factorization have laid the groundwork for employing more advanced generative models like GAIN, which further enhance data imputation by maintaining the natural multidimensional structure of learning data. This alignment with deep generative models has directly influenced our adoption of GAIN to effectively address the complex challenges of data sparsity in ITSs [12].

2.2 Generative AI Models for Educational Data Imputation

There have been tremendous progress in generative AI model for educational data imputations in ITSs. Morales-Alvarez et al. [25] explored the application of generative models, incorporating structured latent spaces and graph neural network-based architectures, to achieve competitive or superior performance in data imputation for real-world mathematics datasets of Eedi (a leading educational platform which millions of students interact with daily around the globe on diagnostic multiple-choice mathematics questions [26]), surpassing traditional baselines such as MICE and missForest. Ma et al. [27] also employed deep generative models to effectively impute data for multiple-choice question learning data, addressing over 70% missing rates in the Eedi educational dataset for mathematics questions. Zhang et al. [28] investigated the use of generative AI models, GAN and GPT, for data augmentation to address sparse learning performance patterns in adult reading comprehension, finding GAN to provide more stable augmentation across various sample sizes.

Learning performance data from ITSs such as AutoTutor CSAL, which focuses on reading comprehension [29], along with ASSISTments [30] and MATHia [31], which both focus on math learning for middle and high-school students, document learners' responses as correct or incorrect to a sequence of questions posed by the system. Inspired by GANs' capability to impute missing regions in images by training on vast amounts of image data and filling in missing areas to maintain coherence with the existing context [32], we are motivated to apply a similar data imputation approach to tensor-based learning performance data. There is an ongoing need for effective imputation of learning performance data in ITSs like AutoTutor CSAL, ASSISTments and MATHia to enhance educational technology's ability to track and assess learners' performance comprehensively.

3 Methods

3.1 Dataset

In this study, we utilized datasets from three primary sources: AutoTutor CSAL lessons[1], ASSISTments[2] and the MATHia[3] dataset from mathematics class. The

[1] AutoTutor Moodel Website: https://sites.autotutor.org/; Adult Literacy and Adult Education Website: https://adulted.autotutor.org/.
[2] ASSISTments Website: https://new.assistments.org/.
[3] MATHia Website: https://www.carnegielearning.com/solutions/math/mathia/.

Table 1. Dataset for the CSAL AutoTutor, ASSISTments and MATHia lessons

Dataset	Lesson Topics	#Learners	#Questions	#Attempts
CSAL Lesson 1	Cause and Effect	118	9	9
CSAL Lesson 2	Problems and Solution	140	11	5
ASSISTments Lesson 1	Algebra Symbolization Studies	318	64	4
ASSISTments Lesson 2	Skill Builder	392	20	4
MATHia Lesson 1	Scale Drawings	500	28	4
MATHia Lesson 2	Analyzing Models of Two-Step Linear Relationships	500	6	4

AutoTutor CSAL lessons cover topics such as "*Cause and Effect*" (CSAL Lesson 1) and "*Problems and Solution*" (CSAL Lesson 2), each comprising 8 to 11 multiple-choice questions designed to test adults' reading comprehension skills. This study was granted ethical approval with the Institutional Review Board (IRB) number: H15257. The ASSISTments dataset cover the lesson topics including "*Algebra Symbolization Studies*" (ASSISTments Lesson 1)[4] and "*Skill Builder*" (ASSISTments Lesson 2)[5]. The MATHia lesson dataset[6] include algebra lessons on "*Scale Drawings*" (MATHia Lesson 1) and "*Analyzing Models of Two-Step Linear Relationships*" (MATHia Lesson 2). Details such as the total number of learners, the total number of questions, and the maximum number of attempts are summarized in Table 1.

3.2 The 3-D Tensor Representation of Sparse Learning Performance

We define the 3-D tensor $\boldsymbol{\mathcal{T}}_{sparse}$ to encapsulate the learning performance records of U learners on N questions over a sequence of up to M attempts, with $\boldsymbol{\mathcal{T}}_{sparse} \in [0, 1, NaN]^{U \times N \times M}$. Here, $U = max(1, 2, 3, \cdots, u)$ is the maximum number of learners, $N = max(1, 2, 3, \cdots, n)$ the maximum number of questions, and $M = max(1, 2, 3, \cdots, m)$ the maximum number of attempts. Each element τ_{uij} within $\boldsymbol{\mathcal{T}}_{sparse}$ encodes the learning performance as 1 for correct answer, 0 for an incorrect answer, and NaN to signify unobserved data for a specific question's performance at certain attempt.

3.3 The Proposed GAIN-Based Imputation Architecture

Consider the $\boldsymbol{\mathcal{T}}_{sparse}$, representing the learning performance of all learners. This tensor comprises layers along the learner dimension, represented as $\boldsymbol{\mathcal{T}}_{sparse} = (\mathcal{T}_{l_1}, \mathcal{T}_{l_2}, \cdots, \mathcal{T}_{l_n})$. Each layer, akin to a single-channel "learner image", is a matrix that encapsulates performance values across different questions and attempts for an individual learner. This is visualized in Fig. 1.

For each matrix-based layer $\mathcal{T}_l \in (\mathcal{T}_{l_1}, \mathcal{T}_{l_2}, \cdots, \mathcal{T}_{l_n})$, each entry τ_{lij} in the $N \times M$ matrix may include the performance values of 0, 1 or NaN to present

[4] Assistments 2008–2009: https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=388.

[5] Assistments 2012–2013: https://sites.google.com/site/assistmentsdata/datasets/2012–13-school-data-with-affect?authuser=0.

[6] MATHia 2019–2020: https://pslcdatashop.web.cmu.edu/Project?id=720.

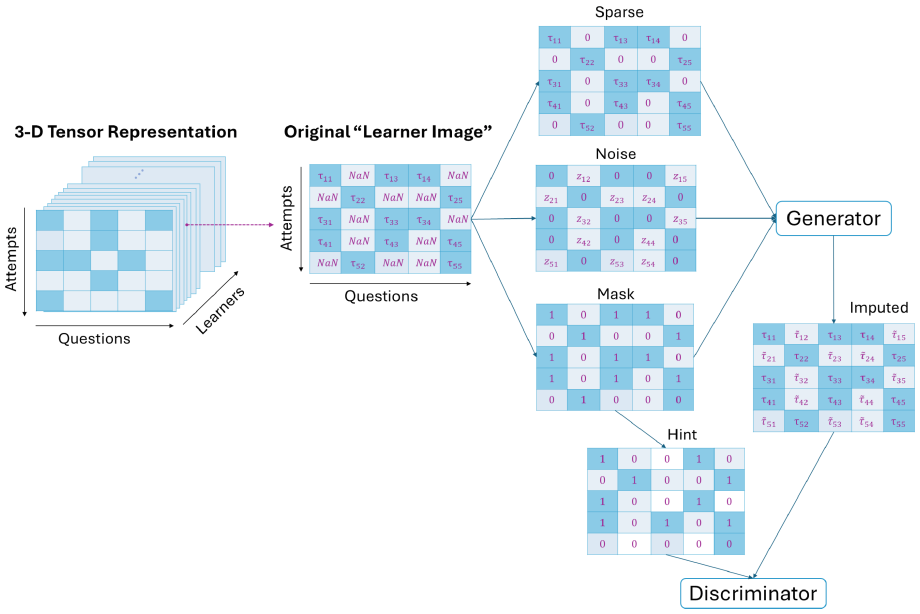


Fig. 1. The proposed GAIN-based imputation architecture for sparse learning performance [12].

the observed data and unobserved data, respectively. One mask matrix $\mathcal{T}_{l_{mask}}$ is supposed to map the observed and unobserved entries within the matrix $\mathcal{T}_l$, with 1 signifying observed data, and 0 indicates unobserved data. One noise matrix $\mathcal{Z}$ with dimensions matching $\mathcal{T}_l$, is initialized. These matrices collectively function as inputs to the generator in the GAIN architecture, producing the output $\mathcal{T}_{lG} = G(\mathcal{T}_l, \mathcal{T}_{l_{mask}}, (1 - \mathcal{T}_{l_{mask}}) \odot \mathcal{Z})$ [12]. Here, the $\odot$ denotes as Hadamard product, indicating element-wise multiplication. The imputed matrix $\mathcal{T}_{l_{imputed}} = \mathcal{T}_{l_{mask}} \odot \mathcal{T}_l + (1 - \mathcal{T}_{l_{mask}}) \odot \mathcal{T}_{lG}$, effectively merges observed and generated data to fill in unobserved entries. Particularly, a hint matrix $\mathcal{T}_{l_{hint}}$, also matching the dimensions of $\mathcal{T}_l$ and derived from the mask matrix $\mathcal{T}_{l_{mask}}$, is introduced. It employs a hint rate to specify the conditional probability that a specific entry in $\mathcal{T}_{l_{imputed}}$ can be observed, given both $\mathcal{T}_{l_{imputed}}$ and $\mathcal{T}_{l_{hint}}$. Thereby, the discriminator within the GAIN architecture, formulated as $D(\mathcal{T}_{l_{imputed}}, \mathcal{T}_{l_{hint}})$, evaluate this as probability [12]. We train $D(\cdot)$ to maximize the probability of correctly predicting the $\mathcal{T}_{l_{mask}}$, while the $G(\cdot)$ is trained to minimize the likelihood of $D(\cdot)$ correctly predicting $\mathcal{T}_{l_{mask}}$. So, we introduce the objective function $V(D, G)$ [12]:

$$V(D,G) = E[\mathcal{T}_{l_{mask}}^T log D(\mathcal{T}_{l_{imputed}}, \mathcal{T}_{l_{hint}}) + (1 - \mathcal{T}_{l_{mask}})^T log(1 - D(\mathcal{T}_{l_{imputed}}, \mathcal{T}_{l_{hint}}))] \quad (1)$$

Our proposed imputation architecture incorporates several novel modifications and configurations from the initial GAIN architecture [12]. See below for further details:

- The convolutional layers are employed for both the generator and discriminator, diverging from the original architecture' reliance on dense layers. Five convolutional neural network (CNN) layers [33], excluding the input and output layers, with the ReLU activation function are applied to the output of each layer.
- During the iterative training phase, the observed data from $\mathcal{T}_l$ and the corresponding imputed data from $\mathcal{T}_{lG}$ are utilized for optimization via the least square loss function, specifically the Root Mean Square Error (RMSE). This method is chosen to not only ensure enhanced stability and superior quality of the generated data [34,35] but also align with probability-based predictions of learning performance in peer research on ITSs [36–38].
- By incorporating a reshape function in the generator's output layer, the shape of generated data $\mathcal{T}_{lG}$ is flexible adjustment to fit the given "learner image" shape, thus accommodating variations across different lesson scenarios without being constrained to a fixed shape, as commonly seen in image-oriented research [11,39,40].

The theoretical foundation for understanding the inference logic and model assumptions in our study includes:

- **Inference Logic.** The entry set within $\boldsymbol{\mathcal{T}}_{sparse}$ can be categorized into two subsets: $\mathcal{T}_{observed}$ for existing values (0 and 1) and $\mathcal{T}_{unobserved}$ for missing ones (NaN). The inference model, formulated as $f_{impute}(\mathcal{T}_{unobserved}|\mathcal{T}_{observed})$, is principle for data imputation, leveraging observed data patterns to impute missing values and predict outcomes [41].
- **Model Assumptions.** Our imputation model operates under several key assumptions within a tensor-based framework: **(a)** *Probability-based prediction*: Assumes predicted learning performance is a continuous probability between 0 and 1, indicative of knowledge mastery [42]. **(b)** *Latent domain knowledge relations*: Posits that unobservable latent relationships within the domain knowledge implicitly influence knowledge mastery [43,44]. **(c)** *Similarity in learning for individual learners*: Suggests a shared relevance and usefulness of knowledge among learners, aiding in predicting knowledge mastery [23,45]. **(d)** *Performance interactions influenced by sequence effects*: Acknowledges that learners' interactions with sequential questions are shaped by priming and recency effects, affecting comprehension and performance [21,46]. **(e)** *Maximum attempt assumption*: Defines a theoretical maximum number of attempts a learner might need, emphasizing the importance of evaluating comprehensive learning states through repeated trials [43].

3.4 Baselines

This study will compare the proposed GAIN-based imputation method against a range of baseline methods, including those from the tensor factorization series and GAN series. Detailed descriptions of these baselines are provided below.

Tensor Factorization: The basic tensor factorization factorizes the sparse tensor $\boldsymbol{\mathcal{T}}_{sparse}$ into two components: a learner latent matrix capturing abilities and learning-related features, and a latent tensor representing knowledge

during question attempts [28,47]. A rank-based constraint is used to maintain a generally positive learning trend and accommodate forgetting or slipping [48]. This refined method enhances data imputation within tensor-based structures, providing a robust solution for handling sparse data.

CANDECOMP/PARAFAC Decomposition (CPD): Drawing on the principle of classic CPD [49,50], the sparse tensor $\boldsymbol{\mathcal{T}}_{sparse}$ is decomposed into three factor tensors that capture learner, attempt and question-related factors in a multidimensional tensor form. A rank-based constraint is additionally applied to enhance the decomposition's accuracy.

Bayesian Probabilistic Tensor Factorization (BPTF): The BPTF [51] is employed to approximate the sparse tensor $\boldsymbol{\mathcal{T}}_{imputed}$ through the decomposition into a sum of outer products of three lower-dimensional factor tensors. This approach leverages Bayesian inference for sampling both the factor tensors and the precision of observed entries, effectively enhancing the model's capacity to manage data sparsity and uncertainty [51,52].

Generative Adversarial Network (GAN): At one core of the GAN, the "learner image" extracted from $\boldsymbol{\mathcal{T}}_{sparse}$ (depicted in Fig. 1), constitutes the base input for the GAN. The GAN architecture includes a generator that simulates data resembling observed entries and a discriminator that assesses the authenticity of this generated data [11]. It uses a consistent CNN layer configuration and least squares loss for optimization.

Information Maximizing Generative Adversarial Nets (InfoGAN): The InfoGAN [39] enhances the traditional GAN framework by integrating the noise with two structured latent variables, allowing for the capture of salient, structured semantic features, such as those relating to learner attributes in ITSs (e.g., initial learning ability and learning rate). The generator generates imputed $\boldsymbol{\mathcal{T}}_{imputed}$ and decodes latent variables. An auxiliary distribution improves the estimation of these variables' posterior, boosting mutual information between latent codes and observations and ensuring that the generated outcomes are meaningfully informed.

AmbientGAN: AmbientGAN [53] is used to impute sparse learning performance data by training on partially observed or corrupted data within a GAN framework. It incorporates a dynamically adjusted Gaussian blur in the measurement process, enabling the discriminator to effectively distinguish between real and generated data measurements and accurately infer the original dataset's true distribution.

3.5 Experimental Setup and Evaluation

The experimental setup for imputing sparse learning performance data incorporates several tailored configurations to optimize model training and evaluation. **(a)** *Cross-Validation*: A systematic five-cycle, five-fold cross-validation strategy is rigorously employed for each model to ensure consistency and reliability of results. **(b)** *Varying Attempt Setting:* The stability of models' data imputation

performance is tested under various maximum attempt settings to handle different degrees of data sparsity. **(c)** *Maximum Iterations:* All models are allowed up to 100 iterations to ensure thorough adaptation and learning. **(d)** *Learning Rate:* A learning rate of either 0.0001 or 0.00001 is selected to promote steady progress and convergence during model training. **(e)** *Regularization Techniques:* Dropout and Batch Normalization are integrated into the training process of GAN-based methods to prevent overfitting. **(f)** *Imputation Accuracy Evaluation Metric:* The Root Mean Square Error (RMSE), as referenced in peer papers [12,24,51], is used to evaluate models' performance in data imputation. **(g)** *Measuring Sparsity Level:* The sparsity level of a tensor-based distribution for learning performance data is computed as the percentage of missing values in the total number of elements in the distribution. **(h)** *Correlation Evaluation:* The Spearman correlation coefficient [54] is used to assess the relationship between RMSE and the varying maximum number of attempts.

4 Results and Discussion

Data Sparsity Levels. Figure 2 displays the variation in sparsity levels within learning performance data across six lessons, categorized by the maximum number of attempts. The sparsity level for each lesson increases with the number of attempts, suggesting a progressive introduction of missing data or non-responses for learners in learning process. This trend is consistent across all courses, albeit with varying rates of increase. Notably, "ASSISTments Lesson 2" exhibits a gradual ascent, recording the highest sparsity levels across all attempts when compared to other lessons. In contrast, "MATHia Lesson 1" and "MATHia Lesson 2" demonstrates a lower initial sparsity level, with the former experiencing a sharp increase and the latter following a more gradual trajectory as attempts progress. Particularly, "CSAL Lesson 1" records the maximum number of attempts observed for this class. The distinct sparsity patterns observed in Fig. 2 highlight the heterogeneity of data completeness and the extent of missingness across different lesson datasets.

Models' Imputation Performance. **RQ1** investigates how effectively the GAIN-based method imputes sparse learning performance data in ITSs compared to established baseline methods. This question is addressed by the following results. Figure 3 presents the RMSE results of imputations performed by various models on sparse learning performance data across lessons. GAIN significantly outperforms other baseline models across most datasets, particularly in the context of ASSISTments

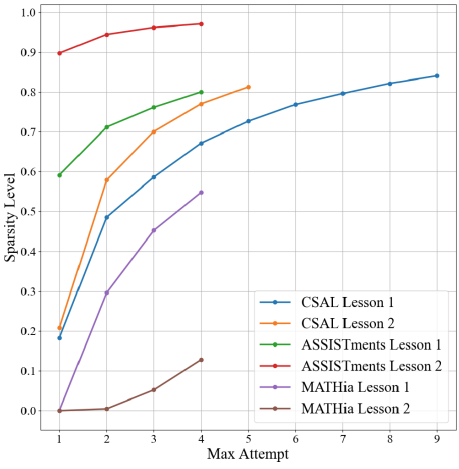


Fig. 2. Data sparsity levels

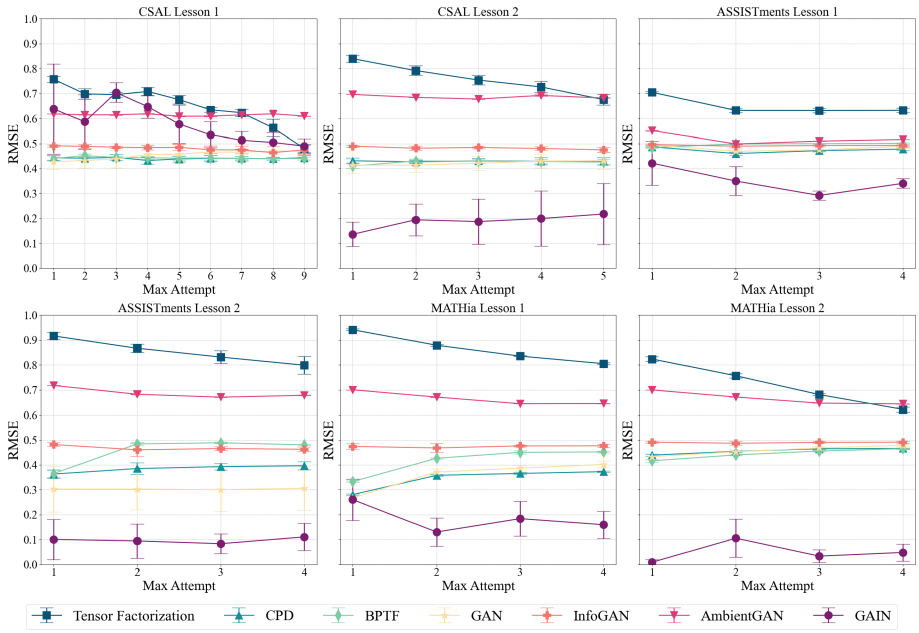


Fig. 3. RMSE performance in different models for all lessons dataset

and MATHia lessons, as well as in certain CSAL lessons, save for an exception in CSAL Lesson 1. In CSAL Lesson 1, GAIN's RMSE is lower than that of BPTF, GAN, and InfoGAN, and its extended error bars indicate a comparative decrease in imputation precision and consistency. Remarkably, by the 9th attempt, GAIN's RMSE approaches the minimal value, signifying high accuracy in data imputation toward the higher max attempt. The unique case of CSAL Lesson 1 underscores complex data or model interactions that merit in-depth research to unravel the specific factors influencing its imputation challenges. Additionally, the GAN model exhibits performance surpassing that of other models, albeit slightly less robust than GAIN, while CPD and BPTF demonstrate competitive capabilities.

Stability of Imputation with Varying Attempts. RQ2 examines how the stability of the GAIN-based model's performance varies with changes in the number of attempts influencing the sparsity levels of learning performance data, as demonstrated by the subsequent results. Despite its overall superior performance, GAIN exhibits greater variance in its results, as indicated by the longer error bars (see Fig. 3), which suggests less stability in its data imputation. The heightened variance suggests that while GAIN generally delivers superior imputation, its consistency is compromised under certain data conditions, possibly requiring additional tuning or pre-processing to stabilize its performance. Moreover, the comparative analysis of other baseline models like Tensor Factorization and CPD indicates a lower variance, suggesting that these may provide more

reliable imputations in certain contexts, despite not always achieving the lowest RMSE.

Iterative Changes of RMSE for GAIN. As depicted in Fig. 4, the RMSE trajectory during the example training stage demonstrates the evolution of GAIN's imputation performance across various lessons with each iteration. Implementing an early stopping criterion is essential when the model exhibits satisfactory performance, typically when the RMSE approximates 0.1, to prevent overfitting. Initially, there is a pronounced reduction in RMSE across all lessons, signaling GAIN's rapid enhancement in accuracy. Particularly, "MATHia Lesson 1", "MATHia Lesson 2", and "ASSISTments Lesson 1" exhibit the most considerable decrease, achieving optimal RMSE levels within fewer than 20 iterations. The prolonged convergence for "CSAL Lesson 2" beyond 40 iterations implies that additional gains in accuracy are marginal, prompting considerations for early stopping to optimize computational resources. The variability in the number of iterations required to reach convergence further underscores the diversity of the underlying data distributions. These findings illuminate the complexity of learning performance patterns and the distinctive characteristics of knowledge acquisition that GAIN captures, albeit with varying rates of convergence.

Spearman Correlation. Figure 5 provides a comparison of the Spearman correlation coefficients, quantifying the relationship between RMSE values and the varying maximum number of attempts across various models applied to different lessons. A positive correlation signifies that the model's RMSE rises as the maximum number of attempts increases, which aligns with a trend of rising sparsity levels, thereby indicating a decline in model performance. Conversely, negative values suggest that the RMSE does not necessarily rise with sparsity, potentially indicating a model's higher performance to missing data. For tensor factorization methods, the prevalent negative correlations across the lessons suggest that their RMSE tends to decrease alongside sparsity. CPD exhibits varied outcomes, with certain lessons reflecting slight positive correlations, while others show negative, indicating inconsistent behavior across different datasets. BPTF predominantly shows positive correlations, with the exception of the CSAL lessons, suggesting a general tendency for model performance to decrease with sparsity within these contexts. For the majority of lessons, GAN demonstrate positive correlations, signaling weaker performance as sparsity enlarges. InfoGAN mostly records neg-

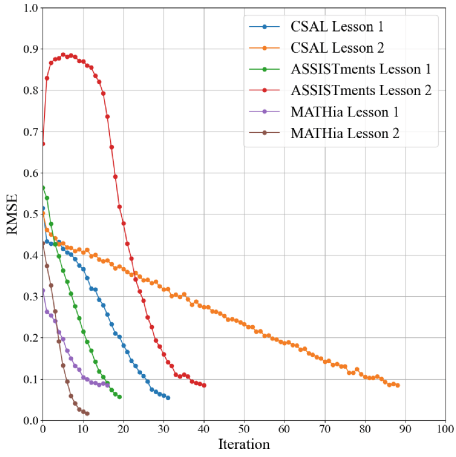


Fig. 4. Data sparsity levels

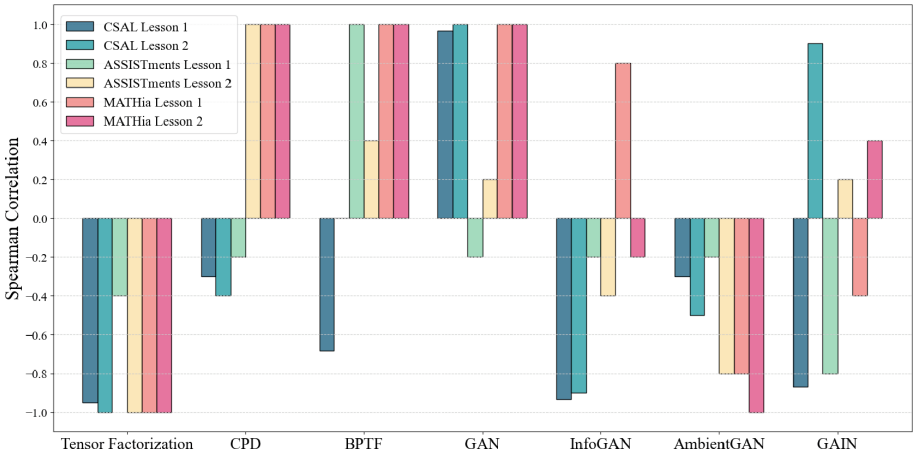


Fig. 5. Spearman correlation of RMSE with varying attempts.

ative values, suggesting improved performance in the face of increasing sparsity for most lessons. AmbientGAN consistently exhibits negative correlations for all lessons, implying an increase in model performance despite greater sparsity. GAIN's results are varied, reflecting its nuanced response to the distinct traits of each dataset.

4.1 Limitations and Future Works

However, the exploration into the adaptive mechanisms of GAIN also highlights areas for future research, particularly in refining model architecture and expanding model explainability to better understand the underlying imputation processes. As educational data continues to grow in complexity and size, further advancements in generative models will be crucial in fully harnessing the potential of generative AI to transform AI-based educational systems.

Future research will delve deeper into the analysis of tensors imputed by GAIN and other methods, focusing on the following areas:

- Enhanced evaluation of imputed data is imperative, with an emphasis on comparing imputed and original data patterns.
- The use of synthetic datasets, with their adjustable sparsity levels and known ground truth, will further facilitate the evaluation process.
- The exploration of generative AI for educational data imputation is still evolving. A systematic comparison of various generative models, including Autoencoders (AE), Variational Autoencoders (VAE), and their interpretable counterparts such as adversarial AEs and Denoising Autoencoders (DAE) [55], is planned.
- A concerted effort will also be made to distinguish between the semantics of zero values and NaN values, enhancing the quality of the imputation for learning performance data.

- An ablation study can verify the effectiveness of the new GAIN-based architecture for imputing sparse learning performance data. By iteratively testing variants by removing or replacing components such as convolutional layers, the hint mechanism, or the least squares loss function, and comparing their performance using key metrics like RMSE, we can identify the contributions of individual components and refine the architecture for optimal effectiveness.
- Another potential is the application of generative models in data imputation, which can facilitate ITSs in tracing and predicting learning performance data, especially in real-time and dynamic learning environments. The emerging generative language models, with their reasoning capabilities and the powerful computational abilities of deep generative models, can potentially lead to new advancements in AI-educational applications and research [56,57].

5 Conclusion

In this study, we propose a GAIN-based method for imputing sparse learning performance data from ITSs. Our systematic comparison with tensor factorization and other GAN-based methods shows that GAIN not only surpasses these traditional models in terms of imputation accuracy but also demonstrates remarkable adaptability across various educational datasets. However, GAIN's performance is marked by increased variance and diminished stability in data imputation, influenced by varying levels of data sparsity and not uniformly consistent across different lessons. Furthermore, the initial tensor-based representation within a 3D tensor space preserves the original sequence effects and structure, which, when combined with GAIN's use of CNN for its input and output layers, effectively bridges the gap between employing generative AI models for imputing sparse learning performance data and retaining essential temporal educational dynamics. The success of GAIN in this context lays the groundwork for more robust educational data analytics, enhancing decision-making in educational settings, especially in ITSs. This study significantly enriches the application of GAIN in the fields of learning engineering and learning science.

Acknowledgements. This paper is grateful to Prof. Philip I. Pavlik Jr. from the University of Memphis and Prof. Shaghayegh Sahebi from the University at Albany - SUNY for their invaluable assistance with tensor factorization in the early stages of this research. Additionally, we extend our thanks to Prof. Arthur C. Graesser, also from the University of Memphis, for his insightful communications that significantly enriched our understanding and inspired deeper analytical thinking.

References

1. Psathas, G., Chatzidaki, T.K., Demetriadis, S.N.: Predictive modeling of student dropout in MOOCs and self-regulated learning. Computers **12**(10), 194 (2023)
2. Baker, R.S.: Modeling and understanding students' off-task behavior in intelligent tutoring systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1059–1068 (2007)

3. Saarela, M.: Automatic knowledge discovery from sparse and largescale educational data: case Finland, PhD thesis. University of Jyväskylä (2017)
4. Greer, J., Mark, M.: Evaluation methods for intelligent tutoring systems revisited. Int. J. Artif. Intell. Edu. **26**(1), 387–392 (2016)
5. Batista, G.E., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. Appl. Artif. Intell. **17**(5-6), 519–533 (2003)
6. Donders, A.R., et al.: A gentle introduction to imputation of missing values. J. Clin. Epidemiol. **59**(10), 1087–1091 (2006)
7. Zhang, Z.: Missing data imputation: focusing on single imputation. Ann. Trans. Med. **4**(1) (2016)
8. Rubin, D.B.: Multiple imputations in sample surveys-a phenomenological Bayesian approach to nonresponse. In: Proceedings of the survey research methods section Of the American Statistical Association. Vol. 1, pp. 20–34 American Statistical Association Alexandria, VA, USA (1978)
9. Rubin, D.B.: Assignment to treatment group on the basis of a covariate. In: J. Edu. Stat. **2**(1), 1–26 (1977)
10. Seaman, S.R., Bartlett, J.W., White, I.R.: Multiple imputation of missing covariates with non-linear effects and interactions: an evaluation of statistical methods. In: BMC Medical Research Methodology **12**, pp. 1–13 (2012)
11. Goodfellow, I., et al.: Generative adversarial nets. Adv. Neural Inf. Proce. Syste. **27** (2014)
12. Yoon, J., Jordon, J., Schaar, M.: Gain: missing data imputation using generative adversarial nets. In: International Conference on Machine Learning. PMLR, pp. 5689–5698 (2018)
13. Dong, W., et al.: Generative adversarial networks for imputing missing data for big data clinical research. BMC Med. Res. Methodol. **21**, 1–10 (2021)
14. Zhang, Y., Zhang, R., Zhao, B.: A systematic review of generative adversarial imputation network in missing data imputation. Neural Comput. Appl. **35**(27), 19685–19705 (2023)
15. Wenyang, H., Wang, T., Chu, F.: Fault feature recovery with Wasserstein generative adversarial imputation network with gradient penalty for rotating machine health monitoring under signal loss condition. IEEE Trans. Instrum. Meas. **71**, 1–12 (2022)
16. Chen, P., Lu, Y., Zheng, V.W., Pian, Y.: Prerequisite-driven deep knowledge tracing. In: 2018 IEEE international conference on data mining (ICDM), pp. 39–48. IEEE (2018)
17. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. In: arXiv preprint arXiv:1907.06837 (2019)
18. Wang, T., Ma, F., Gao, J.: Deep hierarchical knowledge tracing. In: Proceedings of the 12th International Conference on Educational Data Mining (2019)
19. Novak, J.D., Cañas, A.J.: The theory underlying concept maps and how to construct them. Florida Inst. Human Mach. Cogn. **1**(1), 1–31 (2006)
20. Thai-Nghe, N., et al.: "Factorization techniques for predicting student performance". In: Educational Recommender Systems and Technologies: Practices and Challenges. IGI Global, pp. 129–153 (2012)
21. Conway, C.M., Christiansen, M.H.: Sequential learning in non-human primates. In: Trends Cognitive Sci. **5**(12), 539–546 (2001)
22. Conway, C.M.: Sequential Learning. In: Encyclopedia of the Sciences of Learning. Ed. by Norbert M. Seel. https://doi.org/10.1007/978-1-4419-1428-6_72. Boston, MA: Springer US, pp. 3047–3050. isbn: 978-1-4419-1428-6 (2012). https://doi.org/10.1007/978-1-4419-1428-6_72

23. Thai-Nghe, N., et al.: "Matrix and tensor factorization for predicting student performance". In: International Conference on Computer Supported Education. Vol. 2. SciTePress, pp. 69–78 (2011)
24. Sahebi, S., Lin, Y.R., Brusilovsky, P.: Tensor factorization for student modeling and performance prediction in unstructured domain." In: International Educational Data Mining Society (2016)
25. Morales-Alvarez, P., et al.: Simultaneous missing value imputation and structure learning with groups. Adv. Neural. Inf. Process. Syst. **35**, 20011–20024 (2022)
26. Boyle, A., et al.: EEDI evaluation report (2021)
27. Ma, C., Zhang, C.: Identifiable generative models for missing not at random data imputation. Adv. Neural. Inf. Process. Syst. **34**, 27645–27658 (2021)
28. Zhang, L., et al.: "3DG: a framework for using generative AI for Handling Sparse Learner Performance Data From Intelligent Tutoring Systems". In: arXiv preprint arXiv:2402.01746 (2024)
29. Graesser, A.C., et al.: "Reading comprehension lessons in AutoTutor for the center for the study of adult literacy". In: Adaptive educational technologies for literacy instruction. Routledge, pp. 288–293 (2016)
30. Heffernan, N.T., Heffernan, C.L.: The ASSISTments ecosystem: building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching". Int. J. Artif. Intell. Edu. **24**, 470–497 (2014)
31. Ritter, S., et al.: Cognitive Tutor: applied research in mathematics education. Psychon. Bull. Rev. **14**, 249–255 (2007)
32. Pathak, D., et al.: "Context encoders: Feature learning by inpainting". In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536–2544 2016
33. LeCun, Y., et al.: Handwritten digit recognition with a back-propagation network. Adv. Neural Inf. Proce. Syst. **2** (1989)
34. Mao, X., et al.: Least squares generative adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2794–2802 (2017)
35. Yoon, S., Sull, S.: GAMIN: generative adversarial multiple imputation network for highly missing data". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8456–8464 (2020)
36. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized Bayesian knowledge tracing models. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 171–180. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_18
37. Gervet, T., et al.: "When is deep learning the best approach to knowledge tracing?" J. Edu. Data Mining **12**(3), 31–54 (2020)
38. Pavlik, P.I., Eglington, L.G., Harrell-Williams, L.M.: "Logistic knowledge tracing: a constrained framework for learner modeling". IEEE Trans. Learn. Technol. **14**(5), 624–639 (2021)
39. Chen, X., et al.: "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". Adv. Neural Inf. Proce. Syst. **29** (2016)
40. Wang, Y., et al.: PC-GAIN: pseudo-label conditional generative adversarial imputation networks for incomplete data. Neural Netw. **141**, 395–403 (2021)
41. Rubin, D.B.: "Inference and missing data". Biometrika **63**(3), pp. 581–592 (1976)
42. Baker, R.S.J., Corbett, A.T., Aleven, V.: More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 406–415. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69132-7_44

43. Corbett, A.T., Anderson, J.R.: "Knowledge tracing: Modeling the acquisition of procedural knowledge". In: User Modeling and User-adapted Interaction vol. 4, pp. 253–278 (1994)
44. Essa, A.: A possible future for next generation adaptive learning systems. Smart Learn. Environ. **3**(1), 1–24 (2016). https://doi.org/10.1186/s40561-016-0038-y
45. Thai-Nghe, N.,et al.: "Factorization Models for Forecasting Student Performance." In: EDM. Eindhoven, pp. 11–20 (2011)
46. Ramscar, M.: Learning and the replicability of priming effects. Curr. Opin. Psychol. **12**, 80–84 (2016)
47. Zhang, L., et al.: "Exploring the individual differences in multidimensional evolution of knowledge states of learners". In: International Conference on Human-Computer Interaction. Springer, pp. 265–284 (2023)
48. Doan, T.N., Sahebi, S.: "Rank-based tensor factorization for student performance prediction". In: 12th International Conference on Educational Data Mining (EDM) (2019)
49. Carroll, J.D., Chang, J.J.: "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition". In: Psychometrika **35**(3), pp. 283–319 (1970)
50. Harshman, R.A., et al.: "Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multi-modal factor analysis". In: UCLA working papers in phonetics **16**(1), pp. 84 (1970)
51. Xiong, L., et al.: "Temporal collaborative filtering with bayesian probabilistic tensor factorization". In: Proceedings of the 2010 SIAM International Conference on Data Mining. SIAM, pp. 211–222 (2010)
52. Morise, H., Oyama, S., Kurihara, M.: Bayesian probabilistic tensor factorization for recommendation and rating aggregation with multicriteria evaluation data. Expert Syst. Appl. **131**, 1–8 (2019)
53. Bora, A., Price, E., Dimakis, A.G.: "AmbientGAN: Generative models from lossy measurements". In: International conference on learning representations (2018)
54. Spearman, C.: "The proof and measurement of association between two things" (1961)
55. Vincent, P., et al. "Extracting and composing robust features with denoising autoencoders". In: Proceedings of the 25th International Conference on Machine Learning, pp. 1096–1103 (2008)
56. Zhang, L., et al.: "Predicting learning performance with large language models: a study in adult literacy". In: International Conference on Human-Computer Interaction. Springer, pp. 333–353 (2024)
57. Zhang, L., et al.: "SPL: A Socratic Playground for Learning Powered by Large Language Mode". In: arXiv preprint arXiv:2406.13919 (2024)

SWave: Improving Vocoder Efficiency by Straightening the Waveform Generation Path

Pan Liu, Jianping Zhou, Xiaohua Tian, and Zhouhan Lin(✉)

Shanghai Jiao Tong University, Shanghai, China
{wslp1999,jianpingzhou,xtian}@sjtu.edu.cn, lin.zhouhan@gmail.com

Abstract. Diffusion model-based vocoders have exhibited outstanding performance in the realm of speech synthesis. However, owing to the curved nature of generation path, they necessitate traversing numerous steps to guarantee the speech quality, hindering their applicability in real-world scenarios. In this paper, we propose *SWave* (Code and speech samples are available at https://swave-demo.github.io/), a novel vocoder based on rectified flow which improves the efficiency of speech synthesis by *S*traightening the *Wave*form generation path. Specifically, we employ rectification to transform the noise distribution into the data distribution with a probability flow that is as straight as possible. Subsequently, we use distillation and fine-tuning to further enhance the generation efficiency and quality, respectively. Experiments on the LJSpeech dataset demonstrate that compared with other vocoders such as FastDiff and WaveGrad, SWave enhances the generation efficiency. In particular, with a straightforward sampling schedule, SWave generates comparable speech to WaveGrad with significantly fewer steps (2 steps vs 25 steps).

Keywords: Speech Synthesis · Vocoder · Rectified Flow · Generation Efficiency

1 Introduction

Deep generative models have made remarkable advancements in the field of speech synthesis [1,10,15,16]. In the early studies, autoregressive models [8,15,16] dominate this field and greatly improve the quality of synthesized speech. Nevertheless, these models require a substantial number of steps for sequential generation, impeding their applicability in low-latency scenarios. To overcome this limitation, non-autoregressive models emerge and markedly speed up speech synthesis. These models can be subdivided into Variational AutoEncoders (VAEs) [17], flow-based models [9,19,20], Generative Adversarial Networks (GANs) [10,12,22] and Denoising Diffusion Probabilistic Models (DDPMs) [1,5,11]. Among them,

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_26.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 397–408, 2025.
https://doi.org/10.1007/978-3-031-78172-8_26

diffusion models exhibit superior overall performance compared with the subpar generation quality of VAEs, the high design complexity of flow-based models, and the training instability of GANs. Notably, WaveGrad [1], the first diffusion model-based vocoder, stands out as the initial non-autoregressive vocoder that bridges the gap in speech quality with autoregressive vocoders.

Although diffusion model-based vocoders exhibit superior generating capabilities, their generation efficiency lags behind that of other non-autoregressive vocoders. It is noteworthy that these vocoders formulate the probability flow between noise distribution and data distribution with stochastic differential equation (SDE) or complex ordinary differential equation (ODE) [21], resulting in a highly curved generation path. Consequently, they necessitate traversing numerous steps to mitigate path distortion and, hence, ensure the quality of generated speech.

To improve the generation efficiency, the mainstream method is to find the optimal sampling schedule, which can reduce the generation steps while preserving the generation quality as much as possible. WaveGrad [1] utilizes grid search to find the optimal sampling schedule and achieves acceptable generation results with 6 steps. The concurrent work DiffWave [11] adopts a light model architecture, designs a fast sampling algorithm, and is also able to generate high-fidelity speech with 6 steps. Instead of designing manually, FastDiff [6] adpots a noise schedule predictor to obtain the optimal sampling schedule. Another efficient vocoder InferGrad [2], proposes to incorporate the inference process into training with an additional loss to further speed up speech synthesis, achieving comparable generation quality to WaveGrad with a 3× speedup. The main drawback of these models is their reliance on sampling schedules, necessitating meticulous design or time-consuming search. Moreover, none of these models focus on altering the curved generation path, which is the key factor of low generation efficiency. Intuitively, if the generation path is straight enough, it could be possible to generate high-fidelity speech with a single step.

Recently, rectified flow (reflow) [13] is dedicated to straightening the generation path. It reformulates the standard diffusion model and exploits linear interpolation to model the probability flow between noise distribution and data distribution, which leads to a much straighter generation path. Benefiting from the high straightness of the path, reflow demonstrates a significant improvement in sampling efficiency. Moreover, the sampling schedule generally follows the downsampling of the training schedule, which has been applied in recent studies to improve the efficiency of acoustic models [3,4]. Both Reflow-TTS [3] and VoiceFlow [4] effectively reduces the number of generation steps while maintaining speech quality comparable to GradTTS [18].

Motivated by these observations, we propose to integrate reflow into the vocoder, another component within the speech synthesis system, resulting in an efficient vocoder named SWave. SWave adopts the same network architecture as WaveGrad [1] but employs a completely different training workflow consisting of three stages: rectification, distillation, and fine-tuning. Rectification straightens the generation path by iteratively constructing data pairs and retraining. Distillation reduces the number of generation steps while preserving the generation

quality, and fine-tuning eliminates the noise introduced by data pair construction to further enhance generation quality. Experiments on the LJSpeech dataset [7] illustrate that SWave exhibits greater efficiency than other efficient vocoders like FastDiff and WaveGrad. Moreover, SWave can generate speech of comparable quality to the 25-step WaveGrad with only 2 steps. All the sampling schedules of SWave follow a simple approach, eliminating the need for exhaustive search procedures.

Our work makes the following contributions: (a) we propose a novel method to improve the efficiency of the vocoder by straightening the generation path through reflow; (b) we develop a 3-stage training workflow to facilitate the integration of reflow and obtain an efficient vocoder named SWave; (c) experimental results show that SWave outperforms other efficient vocoders, and its sampling schedules are more straightforward.

2 Related Work

Diffusion Model-based Vocoder: WaveGrad [1] is the first diffusion model-based vocoder and the initial non-autoregressive vocoder that bridges the gap in speech quality with autoregressive vocoders. It introduces a model conditioned on continuous noise levels to support a flexible sampling schedule. To accelerate sampling, it employs grid search to find a 6-step noise schedule capable of generating acceptable results. The concurrent work DiffWave [11] adopts a lighter model architecture than WaveGrad and also designs a fast sampling algorithm to generate high-fidelity speech with 6 steps. FastDiff [6] proposes an additional network to predict the sampling schedule to avoid meticulous design and can also generate acceptable speech within 6 steps. InferGrad [2] is a variant of WaveGrad, achieving comparable generation quality as WaveGrad with a 3× speedup by incorporating inference process into training. Specifically, after determining the range of inference schedules under a few steps, InferGrad generates the waveform from random noise following these schedules, and introduces an auxiliary loss to the training objective to minimize the gap between the generated waveform and the ground truth. Our work is also a variant of WaveGrad, but different from InferGrad, SWave enhances WaveGrad through a completely novel approach, with the focus on improving sampling efficiency by straightening the generation path, which also distinguishes SWave from other efficient vocoders.

Rectified Flow and Its Applications in Speech Synthesis: Rectified flow [13] is a reformulation of standard diffusion model. Instead of diffusing the data distribution through either SDE or intricate ODE [21], reflow transforms noise distribution into data distribution with a probability flow that is as straight as possible. Starting from any noise sample, it only takes a few Euler discretization steps to generate a data sample of acceptable quality. This mechanism has been utilized by recent studies to improve the efficiency of speech synthesis, such as voiceflow [4], which integrates reflow into the acoustic model and significantly reduces the number of generation steps while achieving similar speech quality to GradTTS [18]. However, the integration into another component within the

speech synthesis systems, namely the vocoder, remains inadequately explored. It is noteworthy that, compared with the acoustic model, the vocoder imposes higher demands on generation quality. The reason is that the acoustic models generally utilize the noise-robust HiFi-GAN [10] as their vocoder, mitigating the impact of noisy acoustic features on speech quality. In contrast, noise in vocoder generation manifests directly as perturbed speech. Our approach meets the high demand for generation quality and successfully improves the generation efficiency of vocoder.

3 Proposed Method

The training workflow of SWave is depicted in Fig. 1, which comprises three stages: rectification, distillation and fine-tuning. The specific details are described in the following sections.

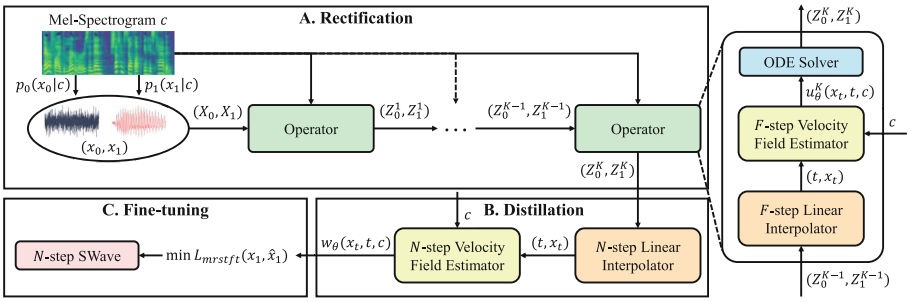


Fig. 1. Training Workflow of SWave. During rectification, we randomly sample some noises and speeches to construct the data pairs (X_0, X_1), and then apply the operator K times to straighten the generation path from noise to speech. During distillation, we utilize the data pairs constructed by the F-step VFE in the K-th operator to distill an N-step VFE. Finally, we fine-tune the N-step VFE with the ground truth and obtain N-step SWave. F is generally set to 1,000, and $N \ll F$.

3.1 Rectification

Rectification is the key stage in the training workflow. It applies repeat operations to continuously straighten the generation path.

Given a mel-spectrogram c, we denote the noise distribution as $p_0(x|c)$ and the data distribution as $p_1(x|c)$. We randomly sample $x_0 \sim p_0(x|c), x_1 \sim p_1(x|c)$ and construct a data pair (x_0, x_1). Then we establish a linear path pointing from x_0 to x_1:

$$x_t = x_0 + t(x_1 - x_0), \quad t \in [0, 1], \tag{1}$$

where x_t is the interpolation of x_0 and x_1. The linear path follows the ODE $\mathrm{d}x_t = (x_1 - x_0)\mathrm{d}t$, which is non-causal since updating x_t requires the information

of the destination x_1. By setting the velocity field $v(x_t, t, c)$ to drive the flow to follow the direction $(x_1 - x_0)$ as much as possible, the linear path follows a new ODE flow $\mathrm{d}x_t = v(x_t, t, c)\mathrm{d}t$, eliminating the dependency on x_1 and rendering it causal. The velocity field is fitted by a model known as the velocity field estimator (VFE) with an objective similar to that in reflow [13]:

$$\min_{\theta} \mathbb{E}_{t,x_0\sim p_0(x|c),x_1\sim p_1(x|c)}[\|(x_1 - x_0) - u_\theta(x_t, t, c)\|^2], \tag{2}$$

where $u_\theta(\cdot, \cdot, \cdot)$ denotes VFE and θ is the model parameters.

If we ensure that the set of data pairs, denoted as (X_0, X_1), keeps all the linear paths from crossing, then the directions of all $u_\theta(x_t, t, c)$ are uniquely determined. With such velocity fields, it only takes a single Euler discretization step to generate high-fidelity speech from any noise sample. However, due to the random matching of x_0 and x_1, the occurrence of intersections among linear paths is inevitable. When two paths intersect at x_τ, $u_\theta(x_\tau, \tau, c)$ is the expectation of these two directions. Intuitively, the final learned velocity fields rewire the individual trajectories passing through the intersection points to avoid crossing, while also inducing curvature in the generation path.

To further straighten the generation path, we design an operator and apply it recursively. The operator consists of three components: an F-step linear interpolator, an F-step VFE and an ODE solver, where F is generally set to 1,000. In the k-th operator, we refer to the F-step VFE as k-VFE, denoted as $\mathbf{Z}^k$, with its estimated velocity field represented as $u^k_\theta(x_t, t, c)$. With this design, the training of the VFE induced from (X_0, X_1) constitutes a part of the first operator. After this VFE converges, we utilize ODE solver to construct new data pairs. Specifically, we randomly sample some noises Z^1_0 and then generate their corresponding speeches Z^1_1 following the ODE $\mathrm{d}x_t = u^1_\theta(x_t, t, c)\mathrm{d}t$.

Due to rewiring, the new data pairs (Z^1_0, Z^1_1) have fewer intersections and hence further straighten the generation path of $\mathbf{Z}^2$. We measure the curviness of model $\mathbf{Z}^k$ [13] by

$$C(\mathbf{Z}^k) = \int_0^1 \mathbb{E}\left[\|(Z^k_1 - Z^k_0) - \dot{Z}^k_t\|^2\right] \mathrm{d}t, \tag{3}$$

where Z^k_0, Z^k_1 and $\dot{Z}^k_t$ denote the noise samples, the generated speeches and the estimated velocity fields at t of $\mathbf{Z}^k$, respectively. The smaller the value of $C(\cdot)$, the straighter the generation path and $C(\cdot) = 0$ means exact straightness. Upon recursive application of the operator, $C(\mathbf{Z}^k)$ will converge to 0 with a convergence rate of $\mathcal{O}(1/k)$. The proof is available in the appendix A.1.

3.2 Distillation

With a highly straight generation path, distillation can effectively reduce the number of sampling steps while maintaining the generation quality [14].

After applying the operator K times during rectification, we use the data pairs generated by K-VFE to distill a new VFE. During distillation, we set the

number of time steps to N ($N \ll F$) and the time increases linearly from 0 to 1, defined as $Linear(0, 1, N)$. We name the distilled VFE as N-step VFE and denote its estimated velocity field as $w_\theta(x_t, t, c)$.

3.3 Fine-Tuning

Although the rectification operator enhances the straightness of the waveform generation path, the constructed data pairs introduce noise into the generation results of the retrained model. The reason is that the speech samples generated by $\mathbf{Z}^k$ exhibit degraded quality compared with the ground truth, owing to various factors such as limited training data and constrained model capability. As the number of operations increases, the noise accumulates continuously, leading to a significant loss of quality in the final generation.

To mitigate the introduced noise, we ultimately fine-tune the N-step VFE by minimizing the Multi-Resolution Short-Time Fourier Transform (MRSTFT) loss [22] between the ground truth and the generated speech:

$$L_{\text{mrstft}}(x_1, \hat{x}_1) = L_{\text{sc}}(x_1, \hat{x}_1) + L_{\text{mag}}(x_1, \hat{x}_1), \tag{4}$$

where x_1 and $\hat{x}_1$ denote the ground truth and the generated speech, respectively; $L_{\text{sc}}(\cdot, \cdot)$ and $L_{\text{mag}}(\cdot, \cdot)$ denote *spectral convergence* and *log STFT magnitude* loss, respectively.

To avoid the increase in training costs caused by back-propagation through time (BPTT), we predict $\hat{x}_1$ at each time step separately:

$$\hat{x}_1 = x_t + (1 - t) \times w_\theta(x_t, t, c). \tag{5}$$

4 Experiments

4.1 Experimental Setup

Dataset: We utilize the LJSpeech dataset [7], consisting of 13,100 speech samples recorded at 22.05kHz from a single female speaker, with a total duration of approximately 24 h. The dataset is randomly divided into two parts: 13,000 samples for training and 100 samples for testing.

Implementation Details: During rectification, we apply the operator three times, taking into account both the training cost and the convergence rate of $C(\mathbf{Z}^k)$. Each k-VFE undergoes training for 120K iterations. We generate one million data pairs for each $\mathbf{Z}^k$ and each pair is generated in 1,000 steps to minimize generation errors. During distillation, We distill two models: a 2-step VFE and a 10-step VFE, each trained for 120K iterations. The final fine-tuning process takes 60K iterations. During inference, all the sampling schedules are $Linear(0, 1, N)$. 2-step generations are from the 2-step SWave, while 5-step and 10-step generations are from the 10-step SWave.

Evaluation Metrics: We conduct a comprehensive evaluation to assess the sample quality of SWave. For subjective evaluation, we use the Mean Opinion

Score (MOS) along with a 95% confidence interval. A test set is presented to 20 listeners who assess speech quality using a rating scale ranging from 1 to 5. Comparative MOS (CMOS), another subjective metric, is employed to indicate the quality gap between the speeches generated by different models. Specifically, 20 listeners rate the samples with scores ranging from -3 to 3. In the objective evaluation, we utilize mel cepstral distance (MCD), log-mel spectrogram mean squared error (LS-MSE), MRSTFT loss, and short-time objective intelligibility (STOI) [2]. Note that MCD is calculated with its official implementation (mcd-cli)[1] and all parameters are set to their default values. We use real-time factor (RTF) to evaluate the inference speed of vocoders on a single NVIDIA GeForce RTX 3090 GPU.

Comparative Models: As SWave adopts the identical network architecture as WaveGrad [1], we take WaveGrad as the primary comparative model. We follow the publicly available and widely used implementation[2] and train WaveGrad for 600K iterations. Additionally, we compare SWave with other diffusion model-based vocoders including DiffWave [11], FastDiff [6] and InferGrad [2]. For non-diffusion model-based vocoders, we report the comparison results with WaveNet [16], WaveGlow [19] and HiFi-GAN [10].

4.2 Results and Analyses

Overall Performance. The subjective and objective results are reported in Table 1 and Table 2 respectively. We obtain the 2-step and 6-step inference schedules for WaveGrad through grid searching and adopt a Fibonacci sequence as the 25-step schedule [1]. Table 1 indicates that the subjective quality of speech generated by SWave surpasses that of WaveGrad when the number of generation steps is the same or similar. Furthermore, the speech generated by SWave in 2 steps exhibits comparable quality to the speech generated by WaveGrad in 25 steps. So the generation efficiency of SWave is significantly higher than that of WaveGrad.

Table 1. Subjective evaluations of WaveGrad and SWave.

Model	Steps	MOS (↑)
WaveGrad	2 (Grid Search)	1.17±0.07
SWave	2	3.88± 0.06
WaveGrad	6 (Grid Search)	3.21± 0.09
SWave	5	4.15± 0.11
WaveGrad	25 (Fibonacci)	3.85± 0.07
SWave	10	4.20± 0.11
Ground Truth	-	4.54± 0.09

[1] https://github.com/jasminsternkopf/mel_cepstral_distance.git.
[2] https://github.com/ivanvovk/WaveGrad.git.

Table 2 demonstrates that even with just 2 steps, the objective quality of SWave's generated speech is remarkably high compared to other vocoders. Specifically, its quality rivals that of DiffWave with 50 steps, FastDiff with 6 steps, WaveGrad with 25 steps, InferGrad with 6 steps, and all the non-diffusion model-based vocoders. When we increase the inference steps to 5, SWave generates the highest quality speech. In terms of efficiency, the RTF of SWave with 2 steps is almost lowest among all the vocoders except HiFi-GAN. It is worth noting that although HiFi-GAN achieves the lowest RTF, the quality of its generation is relatively lower. Therefore, the overall performance of SWave exceeds all these vocoders.

Table 2. Objective evaluations of SWave and other conventional neural vocoders. These vocoders are subdivided into three groups: non-diffusion model-based vocoders, conventional diffusion model-based vocoders, and WaveGrad and its variants. * indicates that the results are from the original work. The best results are highlighted in **bold** and the second best results are underlined.

Model	Steps (↓)	MCD (↓)	LS-MSE(↓)	MRSTFT(↓)	STOI(↑)	RTF(↓)
WaveNet	-	4.00	0.544	1.565	0.907	95.757
WaveGlow	-	2.96	0.123	1.163	0.968	0.043
HiFi-GAN	-	3.44	0.187	1.229	0.913	**0.003**
DiffWave	6	4.17	0.400	1.201	0.969	0.133
	50	4.17	0.393	1.179	0.973	1.106
FastDiff	3	4.77	0.365	1.339	0.952	0.018
	6	4.09	0.246	1.208	0.975	0.034
WaveGrad	6	4.56	0.348	1.752	0.955	0.065
	25	3.35	0.270	1.289	0.973	0.269
InferGrad*	2	-	0.202	1.238	0.967	0.025
	6	-	**0.108**	1.060	0.976	0.069
SWave	2	2.58	0.126	1.064	0.975	0.021
	5	**2.40**	0.112	**1.048**	**0.979**	0.054

To intuitively demonstrate the advantages of SWave, we further visualize the mel-spectrogram of speech, as is shown in Fig. 2. The results show that the mel-spectrogram of SWave's generation is closer to the ground truth and exhibits finer details. More examples of mel-spectrogram are provided in appendix A.2.

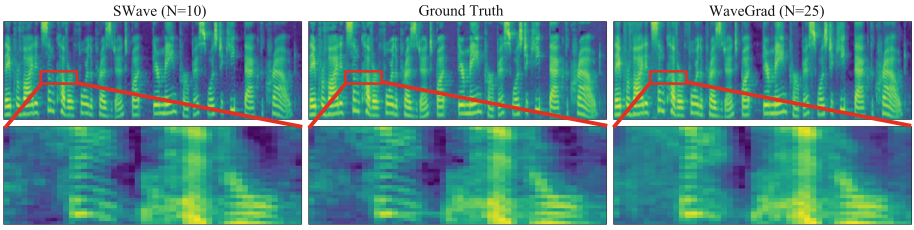


Fig. 2. Comparison of the visualization results of Mel-spectrogram for SWave and WaveGrad.

The key factor of superior performance is that the generation path is straightened. We demonstrate this by visualizing the generation paths of WaveGrad and SWave. Since the waveform of speech is a lengthy sequence, we sample only two time points to facilitate a neat visualization, and the final results are shown in Fig. 3, confirming the effectiveness of our proposed method.

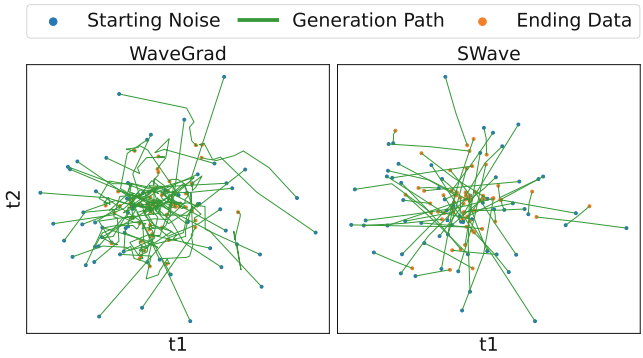


Fig. 3. Differences between generation paths of WaveGrad and SWave.

Effects of Rectification. We investigate the evolution of the straightness of generation paths and its accompanying effects on generations during rectification. We use MCD to quantify these effects. To amplify the effects, we generate all speech samples with 10 steps. The mean MCD of WaveGrad's generations serves as the baseline. The results are depicted in Fig. 4.

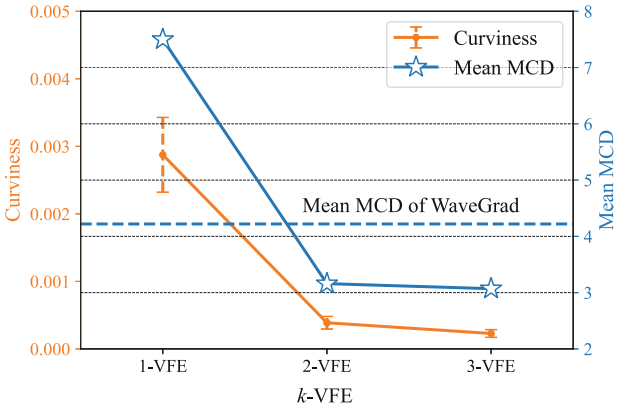


Fig. 4. Evolution of curviness of generation paths and mean MCD of generated samples during rectification. The number of generation steps is 10.

As the rectification progresses, both the mean and standard deviation of curviness decrease, indicating continuous straightening of the generation paths. The mean MCD of VFE exhibits a strong correlation with its curviness, and when $k \geq 2$, k-VFE generates speech samples with lower mean MCD than the baseline. Therefore, rectification straightens the generation path and hence effectively enhances the quality of the generated speech.

Effects of Distillation and Fine-Tuning. The models before and after distillation are 3-VFE and N-step VFE respectively. We demonstrate the effects of distillation by contrasting their generation quality, as shown in Table 3. The results show that 2-step VFE achieves comparable mean MCD to 3-VFE with 10 generation steps. Therefore, distillation improves the generation efficiency.

Table 3. Mean MCD of models before and after distillation. N is the number of generation steps.

Model	N=2	N=10
3-VFE	5.07	3.07
N-step VFE	3.57	2.87

To assess the efficacy of fine-tuning, we compare the speeches generated by N-step VFE and N-step SWave with CMOS. The results are present in Table 4. Without fine-tuning, the quality of both the 2-step generations and 10-step generations becomes slightly worse. Thus, fine-tuning enhances the generation quality.

Table 4. CMOS of models with and without fine-tuning.

Model	N=2	N=10
N-step SWave	0	0
N-step SWave w/o fine-tuning	-1.08±0.21	-1.19±0.22

One-Step Generation. We also study one-step generation and obtain 1-step SWave. Although the generated speech of 1-step SWave is quite clear, the presence of vibrato makes it sound less natural. We provide some samples at https://swave-demo.github.io/.

5 Conclusion

In this paper, we propose SWave, a novel vocoder based on rectified flow, which improves the generation efficiency by straightening the waveform generation path. We adopt the same network architecture as WaveGrad but employ a completely different training workflow comprising three stages: rectification, distillation and fine-tuning. Rectification straightens the generation path through iteratively constructing data pairs and retraining. Distillation reduces the number of generation steps while preserving the generation quality, and fine-tuning eliminates the noise introduced during data pair construction to further enhance generation quality. Experiments on the LJSpeech dataset demonstrate that with a straightforward $Linear(0, 1, N)$ sampling schedule, SWave exhibits superior generation efficiency than other efficient vocoders.

Acknowledgment. This work was sponsored by the National Key Research and Development Program of China (No. 2023ZD0121402, 2020YFB1708700) and National Natural Science Foundation of China (NSFC) grant (No. 62106143, 61922055, 61872233, 62272293).

References

1. Chen, N., Zhang, Y., Zen, H., Weiss, R.J., Norouzi, M., Chan, W.: WaveGrad: estimating gradients for waveform generation. arXiv preprint arXiv:2009.00713 (2020)
2. Chen, Z., et al.: InferGrad: improving diffusion models for vocoder by considering inference in training. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8432–8436. IEEE (2022)
3. Guan, W., Su, Q., Zhou, H., Miao, S., Xie, X., Li, L., Hong, Q.: ReFlow-TTS: a rectified flow model for high-fidelity text-to-speech. arXiv preprint arXiv:2309.17056 (2023)
4. Guo, Y., Du, C., Ma, Z., Chen, X., Yu, K.: VoiceFlow: efficient text-to-speech with rectified flow matching. arXiv preprint arXiv:2309.05027 (2023)
5. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Adv. Neural. Inf. Process. Syst. **33**, 6840–6851 (2020)

6. Huang, R., Lam, M.W., Wang, J., Su, D., Yu, D., Ren, Y., Zhao, Z.: FastDiff: a fast conditional diffusion model for high-quality speech synthesis. arXiv preprint arXiv:2204.09934 (2022)
7. Ito, K., Johnson, L.: The LJ speech dataset (2017)
8. Kalchbrenner, N., et al.: Efficient neural audio synthesis. In: International Conference on Machine Learning, pp. 2410–2419. PMLR (2018)
9. Kingma, D.P., Dhariwal, P.: Glow: generative flow with invertible 1x1 convolutions. Adv. Neural Inf. Process. Syst. **31** (2018)
10. Kong, J., Kim, J., Bae, J.: HiFi-GAN: generative adversarial networks for efficient and high fidelity speech synthesis. Adv. Neural. Inf. Process. Syst. **33**, 17022–17033 (2020)
11. Kong, Z., Ping, W., Huang, J., Zhao, K., Catanzaro, B.: DiffWave: a versatile diffusion model for audio synthesis. arXiv preprint arXiv:2009.09761 (2020)
12. Kumar, K., et al.: MelGAN: generative adversarial networks for conditional waveform synthesis. Adv. Neural Inf. Process. Syst. **32** (2019)
13. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022)
14. Liu, X., Zhang, X., Ma, J., Peng, J., Liu, Q.: InstaFlow: one step is enough for high-quality diffusion-based text-to-image generation. arXiv preprint arXiv:2309.06380 (2023)
15. Mehri, S., Kumar, K., Gulrajani, I., Kumar, R., Jain, S., Sotelo, J., Courville, A., Bengio, Y.: SampleRNN: an unconditional end-to-end neural audio generation model. arXiv preprint arXiv:1612.07837 (2016)
16. Oord, A.v.d., et al.: WaveNet: a generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)
17. Peng, K., Ping, W., Song, Z., Zhao, K.: Non-autoregressive neural text-to-speech. In: International Conference on Machine Learning, pp. 7586–7598. PMLR (2020)
18. Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., Kudinov, M.: Grad-TTS: a diffusion probabilistic model for text-to-speech. In: International Conference on Machine Learning, pp. 8599–8608. PMLR (2021)
19. Prenger, R., Valle, R., Catanzaro, B.: WaveGlow: a flow-based generative network for speech synthesis. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3617–3621. IEEE (2019)
20. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: International Conference on Machine Learning, pp. 1530–1538. PMLR (2015)
21. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. arXiv preprint arXiv:2303.01469 (2023)
22. Yamamoto, R., Song, E., Kim, J.M.: Parallel waveGAN: a fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6199–6203. IEEE (2020)

Outdoor Scene Relighting with Diffusion Models

Jinlin Lai[1], Anustup Choudhury[2(✉)], and Guan-Ming Su[2]

[1] University of Massachusetts Amherst, Amherst, MA, USA
jinlinlai@cs.umass.edu
[2] Dolby Laboratories Inc., Sunnyvale, CA, USA
aachou@dolby.com, guanmingsu@ieee.org

Abstract. Lighting is an environmental condition applied to objects in a scene, which creates complicated lighting effects including shading, shadow and so on. It is a challenging task to relight i.e., manipulate the lighting condition of an outdoor scene image. Recently, diffusion models have shown great success in generating photo-realistic images. Motivated by this, we introduce a novel approach to use diffusion models to relight an outdoor scene. Specifically, we use a conditional diffusion model where we condition on the output of two newly introduced encoders - Surface Encoder and Shadow Encoder. The Surface Encoder is used to extract latent features from the scene intrinsics except the shadow and the Shadow Encoder is used to extract latent features from the normal of the scene and the lighting for the shadow effect. We evaluate our technique both subjectively and objectively and show that while the proposed diffusion model based relighting framework is numerically comparable to the state-of-the-art, it has better visual quality and coherence in relighting typical outdoor scenes (esp. sky region) by rendering photo-realistic images.

Keywords: Diffusion model · Image synthesis · Relighting

1 Introduction

Lights are common phenomena in the real world that come from both nature and human activities. It is interesting yet challenging to understand how external light sources affect the appearances of real-world 3D objects. Studying the lighting effects is crucial in the industry of digital imaging. In practice, objects in a scene are decomposed into intrinsics for internal understanding and manipulation. Intrinsics are properties of scenes that follow an image formation model and typically include reflectance, shape, illumination and shading. The albedo of an object decides the perceived color under white light. In addition, the normal of a surface determines the shading, caused by the reflectance of the light. The overall environment and relative position of objects also decide indirect lighting

J. Lai: The author performed the work while interning at Dolby Labs.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 409–427, 2025.
https://doi.org/10.1007/978-3-031-78172-8_27

effects such as the shadow. To understand the impact of lighting on objects, all of the above factors need to be considered. Due to the difficulty of problems that involve light and illumination, researchers have recently been exploring neural rendering techniques [31] to improve the photo-realism of generated images under varying lighting conditions.

We address the problem of relighting i.e., rendering an outdoor 2D image under a new lighting condition. We assume that the lighting sources for an outdoor scene are infinitely distant and represent the lighting condition as spherical harmonic coefficients [4], which are common assumptions for diffused light source in the literature [37]. Outdoor scene relighting involves two steps - 1) First, albedo, normal, shadow and lighting are extracted from the image using an inverse renderer. 2) Then, under a novel lighting condition, the albedo and the normal are rendered back into a 2D image.

In this work, we use the inverse rendering network from [39] for the first step and focus on the second step which needs to generate lighting effects under the new lighting. Under the new lighting condition, it is challenging to generate the shadow and the sky that are coherent to the scene. Existing work of Yu et al. [37] use two separate neural networks to achieve this task: a shadow net and a sky GAN. However, the boundary between the scene and the generated sky is usually unsatisfactory. See Fig. 1 for a comparison between Yu et al. [37] and ours. We achieve similar relighting results with improved overall photorealism.

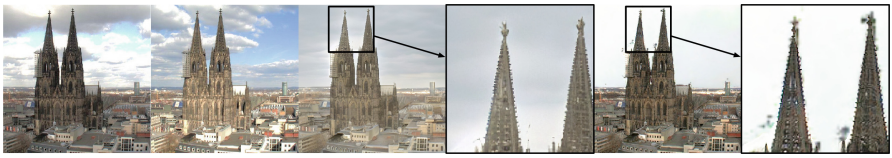


Fig. 1. Relighting results. From left to right: source image, target image from which lighting is used for relighting, our relighting result, zoomed-in region of our result, relighting results from Yu et al. [37], zoomed-in region of [37]'s result. Our generated sky is more coherent to the scene. It is not over-saturated and our result does not contain unwanted artifacts along the edge of the sky, that are present in [37].

Diffusion models (DMs) [9] are the state-of-the-art generative model for various computer vision tasks. Recent works reveal that DMs trained on 2D images can be used for 3D tasks [23]. Relighting is a task that also involves understanding 3D structure inside a 2D image. We propose to improve the existing relighting pipeline using a conditional diffusion model for the rendering step. To encode channels from the inverse renderer, we introduce a Surface Encoder to extract latent features from the albedo, normal, shading and the residual to include effects except the shadow. For the shadow effect, a separate Shadow Encoder is introduced to extract latent features from the normal and the lighting. The latent features act as conditions that are added to a pretrained stable diffusion model [24] using ControlNets [41]. The training of the diffusion model

with two conditions is performed in a progressive way. First, the Surface Encoder is trained with a diffusion model to render an image without shadow but with a coherent sky. Next, the diffusion model is trained using both the Surface Encoder and the Shadow Encoder to generate an image with shadow. Our contributions can be summarized as follows -

- To the best of our knowledge, we introduce the first approach to solve the outdoor scene relighting problem using a conditional diffusion model, which improves the global coherence of lighting effects generation, including sky generation. Compared with state-of-the-art methods, our relighting images are better in terms of IS [28] and FID [8] scores, implying that it has better realism.
- We introduce a novel Surface Encoder and a Shadow Encoder, to add conditions to a pre-trained diffusion model by the ControlNet mechanism.
- We introduce a novel progressive training strategy for our conditional diffusion model.

2 Related Work

2.1 Relighting

Relighting is a core technique in rendering, which can be further divided into face relighting, indoor object relighting and outdoor scene relighting. Previous works such as [17,21,36,40,45] have proposed various relighting techniques for faces and indoor objects . We focus on learning based outdoor scene relighting techniques [5,13,14,19,22,35,37,38], which has been explored under various settings. Philip et al. [22] perform high-quality daytime relighting of single images given multi-view images. The closest works to us are Yu and Smith [38] and Yu et al. [37], which model the lighting as spherical harmonics and perform single image relighting without further supervision. Kubiak et al. [13] formulate relighting as a style transfer problem. In their pipeline, lighting is not explicitly represented so it is not possible to relight with artificial lighting conditions. Griffiths et al. [5] focus on improving shadow prediction in relighting tasks conditioned on sun directions. Yang et al. [35] perform both inverse rendering and rendering by representing the intrinsics in a NeRF [20]. However, training a NeRF for each scene can be costly in some applications. Lyu et al. [19] use diffusion models for inverse rendering using the supervision of geometry. Differently, we learn the rendering step with diffusion models. Several existing works [14,26] propose to change viewpoints and lighting conditions together for outdoor scenes, which are outside the scope of our idea.

2.2 Diffusion Models

Relighting can be also interpreted as an image-to-image translation task, which are usually solved by conditional generative models. Existing works have explored generative adversarial network (GAN) based approaches [11,44] and

diffusion model based approaches [27] in image-to-image tasks under general settings. We take a step further by considering the intrinsics for the relighting task, which leads to a more interpretable procedure.

Conditional generation of diffusion models has also attracted much attention. [2,3,10] explore conditional generation in a pre-trained diffusion models without task specific fine-tuning. Zhang et al. [41] propose ControlNet, which is a mechanism to train a conditional diffusion model with a pre-trained diffusion model as a backbone. We utilize ControlNet for relighting tasks because it allows strong conditions in guided generation.

Diffusion models have been applied on face relighting problems. Papantoniou et al. [21] use diffusion models to learn the UV textures of faces, so that the 3D structure of a 2D face image can be obtained by inpainting. Zhang et al. [40] employ the pre-trained stable diffusion and a 2D GAN to construct relightable NeRF representations for faces. None of the existing works based on diffusion models tackle the problem of outdoor scene relighting. None of the existing work use diffusion models as the renderer in the relighting pipeline.

3 Technical Background

3.1 Relighting

In order to relight a 2D image, the 3D structure inside the image needs to be represented. Following Yu et al. [37], we assume a pixel m that corresponds to an object in the image $\mathbf{i}$ can be decomposed as ($\odot$ is element-wise multiplication) -

$$\mathbf{i}(m) = s(m)\boldsymbol{\alpha}(m) \odot \mathbf{Lb}(\mathbf{n}(m)), \tag{1}$$

where $\mathbf{i}(m) \in [0,1]^3$ is the color values of the three color channels. $s(m) \in [0,1]$ is the shadow which scales down the brightness of certain areas. $\boldsymbol{\alpha}(m) \in [0,1]^3$ is the albedo which is the basic color without any lighting effect. $\mathbf{L}$ is the lighting matrix with shape $(3,9)$ with 3 corresponding to the color channel and 9 corresponding to the spherical harmonics. $\mathbf{n}(m) \in [-1,1]^3$ is the normal which is the direction of the surface. $\mathbf{b}(\cdot) : [-1,1]^3 \rightarrow R^9$ is a fixed function that converts a normal vector into the values of spherical harmonics basis functions such that $\mathbf{h} = \mathbf{Lb}(\mathbf{n}(m))$ is the shading which means the color with the diffuse surface reflectance model under the lighting condition.

In practice, the inverse rendering is never perfect. Therefore, a residual term $\mathbf{r}(m) \in [-1,1]^3$ is introduced to capture the loss of information during the inverse rendering procedure. Then the decomposition becomes -

$$\mathbf{i}(m) = s(m)(\boldsymbol{\alpha}(m) \odot \mathbf{Lb}(\mathbf{n}(m)) + \mathbf{r}(m)). \tag{2}$$

Relighting can not be performed by naïvely replacing $\mathbf{L}$ with a novel lighting $\mathbf{L}'$. The shadow s and the sky should be predicted in relighting.

3.2 Diffusion Models

We use a pre-trained stable diffusion [24] as our backbone. Stable diffusion is a latent diffusion model which maps high dimensional images $\mathbf{i}$ to lower dimensional latents $\mathbf{x}$ via VQ-VAE [32] and learn the generating process of the latents from a unit Gaussian variable with a diffusion model. The denoising diffusion probabilistic model (DDPM) [9] is used to describe generating procedure. It can be interpreted as a sequence of denoising decoders approximated by a diffusion process via variational inference [12]. The observed variable $\mathbf{x}_0 = \mathbf{x}$ and the latent variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ make the backward denoising model -

$$p(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p(\mathbf{x}_{t-1}|\mathbf{x}_t). \tag{3}$$

The approximate posterior is a forward diffusing procedure -

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{4}$$

After properly setting the hyperparameters, the training objective is equivalent to minimizing the noise prediction error as shown in -

$$L_{\text{LDM}}(\theta) = \mathbb{E}_{\mathbf{i},\mathbf{x}_0=\mathcal{E}(\mathbf{i})}[\mathbb{E}_{t,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2]], \tag{5}$$

where $\mathcal{E}(\mathbf{i})$ is the encoder of VQ-VAE that maps the image to latent space, t is sampled uniformly from 1 to T and $\boldsymbol{\epsilon}_\theta$ is a U-Net [25] that predicts the noise injected to generate $\mathbf{x}_t$.

We utilize ControlNets [41] to add conditions to the stable diffusion model. ControlNet is a mechanism to add conditions to an existing neural network. Suppose there is a parameterized function $u = F(v; \theta)$ and a condition w that has the same shape as v needs to be added. ControlNet makes a copy of the parameter as θ_c and reformulate the output as -

$$\begin{aligned} u =& F'(v, w; \theta_c, \phi_1, \phi_2) \\ =& F(v; \theta) + Z(F(v + Z(w; \phi_1); \theta_c); \phi_2), \end{aligned} \tag{6}$$

where $Z(\cdot; \phi)$ is a zero convolution layer which initializes both the weight and the bias as zeros. The updated function F' is trained over $[\theta_c, \phi_1, \phi_2]$ while keeping the original parameter θ fixed. Before training, due to the zero convolution layers, F' is equivalent to F, ignoring the additional condition w. As the training proceeds, the network learns the contribution of w to the output, while keeping the original parameters intact. For the U-Net inside a diffusion model, the ControlNets are added by copying the down-sampling blocks and the mid blocks.

4 Method

4.1 Overview

The proposed relighting pipeline is shown in Fig. 2. Our diffusion model based relighting pipeline consists of a sky segmentation network [43], an inverse renderer [39], a pre-trained diffusion model [24] and two ControlNets [41] - a Surface

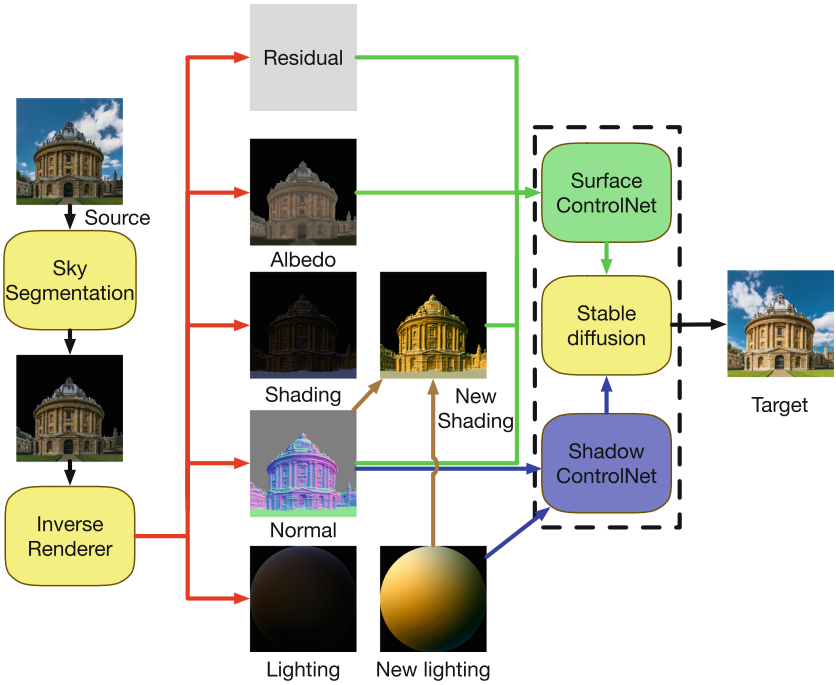


Fig. 2. Relighting pipeline with our diffusion model. From a source image, residual, albedo and normal channels are extracted using an inverse renderer [39] (red). Under a new lighting condition, new shading is calculated from the normal (brown). The residual, albedo, normal and new shading are fed into a Surface ControlNet to account for image synthesis without shadow (green). The normal and new lighting are fed into a Shadow ControlNet for shadow generation (blue). The yellow blocks are codes from existing works. The two ControlNets attached to the stable diffusion, shown in the dotted rectangle, are our contributions. (Color figure online)

ControlNet and a Shadow ControlNet. Sky is first removed from the source image using the sky segmentation network and then decomposed into the intrinsics by the Inverse Renderer. The two ControlNets are trained to add conditions to the diffusion model for relighting tasks. We describe the details of the conditional diffusion model in Sect. 4.2.

4.2 Conditional Diffusion Model

The details of the conditional diffusion pipeline are shown in Fig. 3. From a probabilistic perspective, if the conditional distribution $p(\mathbf{i}|\boldsymbol{\alpha}, \mathbf{n}, \mathbf{r}, \mathbf{L})$ can be learned, then relighting can be achieved by getting $\boldsymbol{\alpha}, \mathbf{n}, \mathbf{r}$ from inverse rendering and simulating $p(\mathbf{i}|\boldsymbol{\alpha}, \mathbf{n}, \mathbf{r}, \mathbf{L}')$ with the new lighting $\mathbf{L}'$. Thus, relighting can be formalized as a learning problem for a conditional diffusion model. The two main

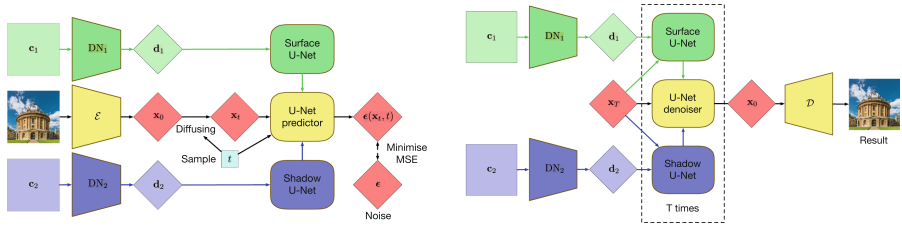
problems are shadow generation and sky generation. In this work, we decompose the simulation of this distribution into two steps -

- First, render an image with sky but without shadow
- Then, render an image with sky and shadow.

These 2 steps can be achieved by 2 ControlNets as follows -

- The first ControlNet, called Surface Encoder (indicated by green box in Fig. 3), takes $\boldsymbol{\alpha}$, $\mathbf{n}$, $\mathbf{r}$ and the shading $\mathbf{h} = \mathbf{Lb}(\mathbf{n}(m))$ as conditions. In the sky area, conditions are set to zeros, which informs the diffusion model to generate the sky region. The pipeline is trained in such a way that with the Surface Encoder, the diffusion model generates an image with the sky but without the shadow.
- The second ControlNet, called Shadow Encoder (indicated by blue box in Fig. 3), takes $\mathbf{n}$ and $\mathbf{L}$ as conditions. It is trained to add the shadow to the generating procedure considering the lighting matrix and the surface information.

Our pipeline adds the shadow on top of the shadow-free image, which is more interpretable than putting all conditions in a single ControlNet. In addition, separating sky and shadow generation also leads to better results with incremental learning.



(a) Training module (b) Inference module

Fig. 3. Pipeline of the proposed conditional diffusion model. Rectangles are image space variables. Rhombuses are latent space variables.The yellow blocks are codes from existing works. (Color figure online)

4.3 Architecture and Training Strategy

We denote the two conditions applied to the Surface Encoder and the Shadow Encoder by $\mathbf{c}_1$ and $\mathbf{c}_2$ respectively. $\mathbf{c}_1$ is formed by concatenating the four channels - $\mathbf{r}$, $\boldsymbol{\alpha}$, $\mathbf{n}$, and $\mathbf{h}$, which leads to the shape of $(512, 512, 12)$. However, in the stable diffusion model, the shape of the input to the down-sampling blocks in the U-Net is $(64, 64, 320)$. Since the ControlNet mechanism assumes the introduced condition has the same shape as the input, we use an additional down-sampling

network $\mathrm{DN}_1(\cdot)$ to convert the condition to match with the shape of the input. Different from Zhang et al. [41] which uses a small network with several convolution layers for down-sampling, we use a more complicated down-sampling network with ResNets [7] and self-attention layers [33]. The architectures of the down-sampling networks are similar to the encoder $\mathcal{E}$ in the VQ-VAE of the stable diffusion, except the first and the last convolution layers, whose channel numbers are determined by the input and output shapes. The output $\mathbf{d}_1 = \mathrm{DN}_1(\mathbf{c}_1)$, called as the Surface Feature, then acts as the first additional condition that is applied to the U-Net using the ControlNet mechanism. The copied U-Net for the Surface Feature is called the Surface U-Net.

For the Shadow Encoder, the condition $\mathbf{c}_2$ includes the normal, $\mathbf{n}$ with shape $(512, 512, 3)$ and the lighting, $\mathbf{L}$ with shape $(9, 3)$. To form a single variable, we reshape the lighting into a vector and replicate it on each pixel and get a matrix of shape $(512, 512, 27)$. Then it is concatenated with the normal to get the condition of shape $(512, 512, 30)$. In order to match the shape of $(64, 64, 320)$, we use a similar down-sampling network $\mathrm{DN}_2(\cdot)$ as the Surface Encoder to convert to shape $(64, 64, 320)$. The output $\mathbf{d}_2 = \mathrm{DN}_2(\mathbf{c}_2)$, called as the Shadow Feature, then acts as the second additional condition that is applied to the U-Net using the ControlNet mechanism. The copied U-Net for the Shadow Feature is called the Shadow U-Net.

In the latent space, outputs from both the ControlNets (Surface ControlNet and Shadow ControlNet) are added together before adding to the U-Net of stable diffusion. Our training pipeline shown in Fig. 3(a) is trained in two stages -

- Stage 1: Train the stable diffusion conditioned on the Surface Encoder.
- Stage 2: Train the stable diffusion conditioned on both the Surface Encoder and the Shadow Encoder.

At the first stage, the parameters in the Surface Encoder, θ_1 (including the down-sampling network, zero-convolution layers and the Surface U-Net) are trained by minimizing the noise prediction error conditioned on $\mathbf{c}_1$ as -

$$L_{\mathrm{SURFACE}}(\theta_1) = \mathbb{E}_{\mathbf{i}', \mathbf{x}_0 = \mathcal{E}(\mathbf{i}')}[\mathbb{E}_{t,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta_1}(\mathbf{x}_t, t, \mathrm{DN}_1(\mathbf{c}_1))\|_2^2]], \tag{7}$$

where $\mathbf{i}'$ is an image with removed shadow following Eq. 2. After the first stage is trained, the conditional diffusion model learns to generate an image with the sky and without shadow. Then the shadow is added to the generated image by optimizing the parameters in the Shadow Encoder, θ_2, with the noise prediction loss conditioned on $\mathbf{c}_1$ and $\mathbf{c}_2$. This loss is formulated as -

$$L_{\mathrm{SHADOW}}(\theta_1, \theta_2) = \mathbb{E}_{\mathbf{i}, \mathbf{x}_0 = \mathcal{E}(\mathbf{i})}[\mathbb{E}_{t,\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta_1,\theta_2}(\mathbf{x}_t, t, \mathrm{DN}_1(\mathbf{c}_1), \mathrm{DN}_2(\mathbf{c}_2))\|_2^2]]. \tag{8}$$

4.4 Relighting Pipeline

The high level pipeline for relighting/inference is shown in Fig. 2. The detailed steps inside our conditional diffusion model is demonstrated in Fig. 3 (b). To perform relighting of a $2D$ image $\mathbf{i}$ with a new lighting $\mathbf{L}'$ using our proposed method with a conditional diffusion model, the following steps are executed -

1. Extract shadow, albedo, normal, residual, lighting by utilizing an existing sky segmentation pipeline and inverse renderer as $(s, \boldsymbol{\alpha}, \mathbf{n}, \mathbf{r}, \mathbf{L}) = \mathrm{IR}(\mathbf{i})$.
2. Obtain shading $\mathbf{h}$ based on the formula $\mathbf{h} = \mathbf{L}'\mathbf{b}(\mathbf{n}(m))$.
3. Form condition $\mathbf{c}_1$ by concatenating $\boldsymbol{\alpha}$,$\mathbf{n}$,$\mathbf{r}$,$\mathbf{h}$ and converting into the Surface Feature $\mathbf{d}_1 = \mathrm{DN}_1(\mathbf{c}_1)$.
4. Form condition $\mathbf{c}_2$ by reshaping $\mathbf{L}'$ into a vector, expanding on each pixel and concatenating with $\mathbf{n}$. Then convert the condition into the shadow feature $\mathbf{d}_2 = \mathrm{DN}_2(\mathbf{c}_2)$.
5. Generate a random sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
6. Simulate $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by sampling until $\mathbf{x}_0$ is obtained, with the U-Net denoiser $\boldsymbol{\epsilon}_{\theta_1,\theta_2}(\mathbf{x}_t, t, \mathbf{d}_1, \mathbf{d}_2)$ and the two features $\mathbf{d}_1$ and $\mathbf{d}_2$.
7. Convert latent space image $\mathbf{x}_0$ into pixel space (Relighted image) by passing through the decoder of the fixed VQ-VAE in the stable diffusion by $\mathbf{i}' = \mathcal{D}(\mathbf{x}_0)$.

5 Experiments

We train our diffusion model on the training split of the MegaDepth dataset [16]. MegaDepth is split to use 149 scenes (107992 images) for training, 17 scenes (9544 images) for validation and 29 scenes (8984 images) for testing. To evaluate the performance of our approach, we use the test split of the MegaDepth dataset and the time-lapse images from the BigTime dataset [15].

Following Yu et al. [37], we use the predicted lighting from one image to relight the scene in another image. There are no ground-truth for relighting results in [16,26], so we mainly compare against the baselines qualitatively. We however report quantitative results on the BigTime dataset [15]. For that, we randomly choose 100 pairs of images from each of the 15 scenes of the dataset to relight the source image with estimated lighting from the target image. In total, the numerical results are reported on 1,500 pairs.

5.1 Implementation Details

We utilize a segmentation network [43] (PSPNet) and an inverse renderer [39] to preprocess the images from the datasets. For each image, we perform center cropping into a square and resize it into $(512, 512, 3)$ with linear interpolation. Then, the PSPNet [43] is used to generate the sky mask. With an image where the sky is masked out, we run the inverse renderer from Yu and Smith [39] to get the surface information $(s, \boldsymbol{\alpha}, \mathbf{n}, \mathbf{r}, \mathbf{L}) = \mathrm{IR}(\mathbf{i})$.

We execute all training stages on four A100s with 80GB GPU memory. For the training of the ControlNets in stage 1, we use the AdamW optimizer [18] for 50 epochs with the batch size as 16. The learning rate is 5×10^{-5} with a constant warmup strategy for the first 500 steps. The parameters are loaded from Stable Diffusion 1.5 [24] and $T = 1000$ is assumed. During training, the text prompt of the stable diffusion is randomly chosen from "A high quality, detailed and professional outdoor scene image without shadow" and an empty sequence. This

approach makes the learned ControlNets recognize semantics from the images instead of the prompts [41]. The setting in the second stage is the same, except that the training is performed for 100 epochs with halved batch size and the text prompt is randomly chosen from "A high quality, detailed and professional outdoor scene image with shadow" and an empty sequence.

The inference of our conditional diffusion model is performed with DDIM [29] for $T = 100$ timesteps. The text prompt is set to "A high quality, detailed and professional outdoor scene image with shadow" for the text embedding in the U-Net but the text guidance is disabled.

5.2 Cross Relighting

For each pair of images $\mathbf{i}_1$ and $\mathbf{i}_2$, we use the inverse renderer to get their lighting conditions $\mathbf{L}_1$ and $\mathbf{L}_2$. If the inverse renderer is perfect, then with $\mathbf{i}_1$ and $\mathbf{L}_2$, the ground-truth of relighting should be $\mathbf{i}_2$. However, no inverse renderer is perfect so the ground-truth can not be obtained. We compare with the results from Yu et al. [37]. We do not compare with [38] because [37] have shown better results.

Visual Evaluation: We show the effects of relighting on the MegaDepth dataset [16] in Fig. 4. We choose images from the test split of the MegaDepth dataset and relight the source images with the lighting conditions from other images. Both our diffusion model and [37] are able to relight the source images with the target lightings. But our generated images have better overall quality and sky regions that are more consistent with the scene. In the first example of Fig. 4, the sky segmentation is affected by the presence of tree branches, making [37] fail in rendering reasonable results in the area. Our method generates an image as a whole, compensating for the imperfectness from pre-processing. In the third example, [37] generates artifacts in the boundary of skies, while relighting image from our method is smoother between scene and sky. Our relighting result in the fourth example shows the effects of white lighting thus creating an almost clear sky and a bright image which looks much better than [37]. We include two failure cases in the last two rows, with the problem that the color themes may be off when relighting with diffusion models. In the last row, both methods adjust the color based on the lighting condition, but the relighting result from [37] is more aligned with the lighting sphere. Diffusion based relighting requires stronger condition of the normal to achieve better contrast in the images. The shadow rendering could be better which shows the limitation of the Shadow Encoder. Image rendering from diffusion models with ControlNets is still dependent on the fixed backbone, which could be improved in future works by also training the latent diffusion model.

With pairs of time-lapse images from the BigTime [15] dataset, it is possible to check the relighting performances with cross relighting i.e., to relight one image with the lighting condition from the other image. Our comparison results on the BigTime dataset are shown in Fig. 1 and Fig. 5. As shown in Fig. 1, we can see that our method does not have unwanted artifacts in the sky when compared

Fig. 4. Cross relighting results on the MegaDepth dataset [16]. The source images are relighted using lighting conditions extracted from target images.

Fig. 5. Cross relighting results on the BigTime dataset. The source images are relighted using lighting conditions extracted from target images.

with [37]. This happens due to the fact that we use a single generative model to relight the entire image as opposed to [37] which uses a SkyGAN. In the first and the third rows of Fig. 5, we can see that the estimated lighting has an orange-ish hue. The effects of the color of the lighting can be see in our final relighting result but is missing in [37]. We use the last three rows to demonstrate failure cases. Compared with [37], our method sometimes does not render with the

correct color theme that follows the shading. In the last row, the target lighting condition indicates a darker theme. Diffusion based relighting does produce a darker scene, but the influence of the lighting condition is not strong enough to render the target image. Even though ControlNets are usually strong conditions applied to diffusion models, we may need stronger conditions to do relighting in such cases.

Numerical Evaluation: We report numerical results using our method and [37] on the BigTime dataset in Table 1. We use pixel-level and patch-level error metrics (PSNR, L_1 error, SSIM [34], LPIPS [42]) as well as image-level quality metrics (IS [28], FID [8]). As ablation study, we tested two different variants of the stable diffusion model - one where all the conditions were included in a single ControlNet and our variant where we use two ControlNets. To be consistent with the evaluation procedure of Yu et al. [37], we show results while masking out the sky region. Additionally, we show results on the entire image as well. Furthermore, to negate the effects of bias, we repeat our experiment three times with three different random seeds and present the averaged numerical results. We observed that the standard deviation for each metric is below 0.5%.

From Table 1, we can see that our our approach of using two ControlNets is consistently better than the approach where all conditions are included in a single ControlNet. Comparing with [37], we find that the relighting images with our diffusion model have slightly worse error metrics (pixel-level and patch-level) on the scene areas (as indicated by the first three rows of Table 1 corresponding to the sky being masked out) but have an overall better quality metrics as whole images as indicated by the last two rows of Table 1 corresponding to the sky being included. This results from two factors - 1) Stable diffusion is not guaranteed to preserve the image details, which is a known issue in its image-to-image pipeline [6]. We will discuss more examples about this in Sect. 5.6. 2) The generated images from stable diffusion are closer to real images, especially in the sky area. This is evident with error metrics on the entire image (indicated by the last two rows of Table 1).

While we report scores using traditional metrics such as PSNR, SSIM and so on, it must be noted that to truly evaluate the effectiveness of generative models, scores such as IS and FID are more reliable because they correlate well with human perception [1]. Given the fact that our FID score is consistently lower (whether or not the sky region is included), we can conclude that the proposed method produces more realistic results compared to [37]. However, these metrics do not necessarily correlate with the relighting performance of the methods.

In order to truly understand the impact of relighting, we further evaluate the perceptual quality with a user study performed by 12 people on 50 images randomly chosen from the BigTime dataset. We presented to the participants the source image, the target illumination, relighting the source image with the target illumination using [37] and the proposed method. We asked the participants to compare the two relighting results and answer two questions: 1) Which image is relit more accurately according to the target illumination and 2) Which image's

Table 1. Averaged numerical results for cross relighting on the BigTime dataset using pixel-level metrics such as PSNR, L_1 error, SSIM, LPIPS, image-level metrics such as IS and FID, and results from our user-study. Considering the target illumination, the results from the user-study are indicated using RL-US for showing which method has better relighting capability and SK-US for showing which method has better sky generation (both in terms of percentage). Some images in the test set do not have sky and that is why SK-US does not add up to 100. The relighted images with our method is comparable to those with [37] in terms of closeness to the target images, has better visual quality measured by IS and FID, and based on the user study is comparable when it comes to relighting and better when it comes to sky generation.

Sky	Method	PSNR (↑)	L_1 error (↓)	SSIM (↑)	LPIPS (↓)	IS (↑)	FID (↓)	RL-US (↑)	SK-US (↑)
Masked	Yu et al. [37]	**20.21**	**0.065**	**0.75**	**0.17**	3.35	26.11	-	-
	One ControlNet	19.20	0.076	0.67	0.22	**3.65**	22.53	-	-
	Ours	20.02	0.066	0.70	0.18	3.57	**21.70**	-	-
Included	Yu et al. [37]	16.11	0.13	**0.68**	**0.31**	3.76	34.86	**50.83**	40.17
	Ours	**16.50**	**0.12**	0.62	**0.31**	**3.84**	**24.65**	49.17	**55.50**

sky has better quality taking into account the target illumination? The results of the user study are shown with the suffix '-US' in Table 1.

The user study reveals that while our method has comparable lighting consistency as compared to state-of-the-art [37], it has much better rendering of the sky region. We also find that the contrast of the images is better for [37] leading to a slightly higher perceptual score for relighting.

5.3 Sky Generation

The most direct benefit of relighting with a single generative model is that the sky generation will be more coherent to the scene. As shown in the first row of Fig. 4, isolated sky generation in [37] usually creates random artifacts at the boundary between sky and scene. The generated images from our diffusion model are much more coherent at the boundary which contributes to better IS and FID scores and validated by the user-study whose results are shown in Table 1. To measure the advantages with additional numerical results, we generate sky masks for the relighting images and compare with the original sky masks by intersection over union (IoU). Our method obtains an IoU of 0.9905 as compared to [37] which obtains an IoU of 0.9896. We note that the unwanted artifacts generated from the SkyGAN in [37] are usually small in size, so the difference in IoU is also small. But a difference of 0.001 implies more than 100 pixels for an 512×512 image, if the sky occupies 40% of the image.

5.4 Rendering with only Surface Encoder

To demonstrate the design of the two encoders in our pipeline, we execute our rendering with only the Surface Encoder. For that, during sampling, we use

Fig. 6. Rendering with Surface Encoder (w.o. shadow) and both Encoders (w. shadow).

Fig. 7. Relighting results with rotated lighting conditions.

the text prompt "A high quality, detailed and professional outdoor scene image without shadow" for instructing the sampling. This disables the Shadow Encoder and uses only the Surface Encoder. The results of using only the Shadow Encoder on some samples from the BigTime dataset are shown in Fig. 6. We can see that using only the Surface Encoder can render images without shadow and with a coherent sky. We observe similar trends across other images in the dataset.

Fig. 8. Relighting results with artificial lighting conditions.

5.5 Additional Relighting Results

The representation of spherical harmonics enables us to manipulate the lighting conditions. For an image from [15], we rotate the lighting condition around the z-axis by 10 evenly distributed degrees in $[0, 2\pi]$. The results are shown in Fig. 7. We can see that the shadow generation follows the lighting direction, and the sky is also coherent with the scene.

In addition, it is also possible to relight real 2D scene images with artificial lights. Figure 8 demonstrates relighting results under random sampled color lights on outdoor scene images from the dataset of Rudnev et al. [26]. It can be observed that the buildings and the skies follow the lighting conditions in general.

5.6 Discussion

It is known that the stable diffusion is not guaranteed to preserve details for image-to-image tasks [6]. This results from the compromise of representing images in the latent space in order to accelerate training and inference. We demonstrate this issue in Fig. 9. During relighting, some high frequency features of the original images, such as small rectangles, can be lost. Although visually plausible, this results in a gap in the error metrics in our comparison.

Recently, with the development of faster training and inference techniques, there has been increasing interest in high-resolution image-space diffusion models [30]. Image-space diffusion models usually have better generation qualities in the details so it will solve this problem in our pipeline. Our contribution is orthogonal

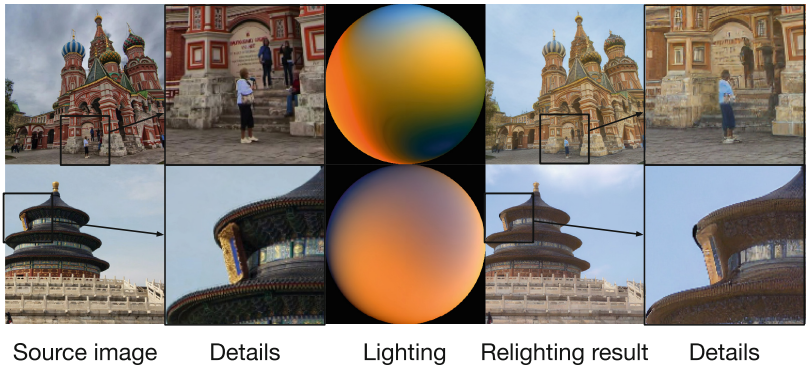


Fig. 9. Detail loss during relighting with our diffusion model.

to the choice of the backbone diffusion models, so we expect better results after integrating with more advanced diffusion models.

6 Conclusion

In this work, we propose a novel approach to perform the rendering step in the relighting pipeline for outdoor scenes with a conditional diffusion model. Our diffusion model based relighting is achieved by adding a Surface Encoder and a Shadow Encoder to the Stable Diffusion model using a ControlNet mechanism. The Surface Encoder is used to render with the scene intrinsics except the shadow and the Shadow Encoder is used to understand to generate the shadow based on the scene normals and lighting. These two encoders are trained in steps following the assumed rendering function. We found that our diffusion model generates relighting images quantitatively comparable to existing works, but with higher visual quality especially while generating the sky region. We demonstrate relighting results with real and artificial lighting conditions.

References

1. Borji, A.: Pros and cons of GAN evaluation measures. arXiv preprint arXiv:1802.03446 (2018)
2. Brooks, T., Holynski, A., Efros, A.: InstructPix2Pix: learning to follow image editing instructions. In: CVPR, pp. 18392–18402 (2023)
3. Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. NeurIPS **34**, 8780–8794 (2021)
4. Green, R.: Spherical harmonic lighting: the gritty details. In: Archives of the Game Developers Conference, vol. 56, pp. 4 (2003)
5. Griffiths, D., Ritschel, T., Philip, J.: Outcast: outdoor single-image relighting with cast shadows. In: Computer Graphics Forum, vol. 41, pp. 179–193. Wiley Online Library (2022)

6. Hao, S., Han, K., Zhao, S., Wong, K.Y.K.: ViCo: detail-preserving visual condition for personalized text-to-image generation. arXiv preprint arXiv:2306.00971 (2023)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
8. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. NeurIPS **30** (2017)
9. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. NeurIPS **33**, 6840–6851 (2020)
10. Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)
11. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.: Image-to-image translation with conditional adversarial networks. In: CVPR, pp. 1125–1134 (2017)
12. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. Mach. Learn. **37**, 183–233 (1999)
13. Kubiak, N., Mustafa, A., Phillipson, G., Jolly, S., Hadfield, S.: Silt: self-supervised lighting transfer using implicit image decomposition. In: BMVC (2021)
14. Li, Q., Guo, J., Fei, Y., Li, F., Guo, Y.: Neulighting: neural lighting for free viewpoint outdoor scene relighting with unconstrained photo collections. In: SIGGRAPH Asia, pp. 1–9 (2022)
15. Li, Z., Snavely, N.: Learning intrinsic image decomposition from watching the world. In: CVPR, pp. 9039–9048 (2018)
16. Li, Z., Snavely, N.: MegaDepth: learning single-view depth prediction from internet photos. In: CVPR, pp. 2041–2050 (2018)
17. Liu, Y., Neophytou, A., Sengupta, S., Sommerlade, E.: Relighting images in the wild with a self-supervised siamese auto-encoder. In: WACV, pp. 32–40 (2021)
18. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
19. Lyu, L., et al.: Diffusion posterior illumination for ambiguity-aware inverse rendering. ACM TOG **42**, 1–14 (2023)
20. Mildenhall, B., Srinivasan, P., Tancik, M., Barron, J., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (2021)
21. Papantoniou, F.P., Lattas, A., Moschoglou, S., Zafeiriou, S.: Relightify: Relightable 3D faces from a single image via diffusion models. In: CVPR, pp. 8806–8817 (2023)
22. Philip, J., Gharbi, M., Zhou, T., Efros, A., Drettakis, G.: Multi-view relighting using a geometry-aware network. ACM Trans. Graph. **38**(4) (2019)
23. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: DreamFusion: Text-to-3D using 2D diffusion. arXiv preprint arXiv:2209.14988 (2022)
24. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR, pp. 10684–10695 (2022)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. In: MICCAI, pp. 234–241 (2015)
26. Rudnev, V., Elgharib, M., Smith, W., Liu, L., Golyanik, V., Theobalt, C.: Nerf for outdoor scene relighting. In: ECCV, pp. 615–631. Springer (2022)
27. Saharia, C., et al.: Palette: Image-to-image diffusion models. In: SIGGRAPH, pp. 1–10 (2022)
28. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. NeurIPS **29** (2016)
29. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)

30. StabilityAI: IF by DeepFloyd Lab at StabilityAI. https://github.com/deep-floyd/IF (2023)
31. Tewari, A., et al.: Advances in neural rendering. In: Computer Graphics Forum, vol. 41, pp. 703–735. Wiley Online Library (2022)
32. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. NeurIPS **30** (2017)
33. Vaswani, A., et al.: Attention is all you need. NeurIPS **30** (2017)
34. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE TIP **13**(4), 600–612 (2004)
35. Yang, S., et al.: Complementary intrinsics from neural radiance fields and CNNs for outdoor scene relighting. In: CVPR, pp. 16600–16609 (2023)
36. Yi, R., Zhu, C., Xu, K.: Weakly-supervised single-view image relighting. In: CVPR, pp. 8402–8411 (2023)
37. Yu, Y., Meka, A., Elgharib, M., Seidel, H., Theobalt, C., Smith, W.A.: Self-supervised outdoor scene relighting. In: ECCV, pp. 84–101 (2020)
38. Yu, Y., Smith, W.A.: InverseRenderNet: learning single image inverse rendering. In: CVPR, pp. 3155–3164 (2019)
39. Yu, Y., Smith, W.A.: Outdoor inverse rendering from a single image using multi-view self-supervision. IEEE TPAMI **44**(7), 3659–3675 (2021)
40. Zhang, H., DAI, T., Xu, Y., Tai, Y.W., Tang, C.K.: FaceDNeRF: semantics-driven face reconstruction, prompt editing and relighting with diffusion models. NeurIPS **36** (2024)
41. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: ICCV, pp. 3836–3847 (2023)
42. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018)
43. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR, pp. 2881–2890 (2017)
44. Zhu, J.Y., Park, T., Isola, P., Efros, A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV, pp. 2223–2232 (2017)
45. Zhu, Z.L., Li, Z., Zhang, R.X., Guo, C.L., Cheng, M.M.: Designing an illumination-aware network for deep image relighting. IEEE TIP **31**, 5396–5411 (2022)

Matching Aggregate Posteriors in the Variational Autoencoder

Surojit Saha(✉), Sarang Joshi, and Ross Whitaker

Scientific Computing and Imaging Institute, The University of Utah, Salt Lake City, USA
surojit.saha@utah.edu, sjoshi@sci.utah.edu, whitaker@cs.utah.edu

Abstract. The variational autoencoder (VAE) [17] is a well-studied, deep, latent-variable model (DLVM) that optimizes the variational lower bound of the log marginal data likelihood. However, the VAE's known failure to match the aggregate posterior often results unacceptable latent distribution, e.g. with *pockets, holes, or clusters*, that fail to adequately resemble the prior. The training of the VAE under different scenarios can also result in *posterior collapse*, which is associated with a loss of information in the latent space. This paper addresses these shortcomings in VAEs by reformulating the objective function to match the *aggregate/marginal* posterior distribution to the prior. We use *kernel density estimation* (KDE) to model the aggregate posterior. We propose an automated method to estimate the kernel and account for the associated kernel bias in our estimation, which enables the use of KDE in high-dimensional latent spaces. The proposed method is named the *aggregate variational autoencoder* (AVAE) and is built on the theoretical framework of the VAE. Empirical evaluation of the proposed method on multiple benchmark datasets demonstrates the advantages of the AVAE relative to state-of-the-art (SOTA) DLVM methods. Here is the link to the code: https://github.com/Surojit-Utah/AVAE.

Keywords: Variational Autoencoders · Aggregate Posterior Matching · Non-parametric Density Estimation

1 Introduction

The development of DLVMs is an important topic of research that is widely used for generative modeling and representation learning. The VAE [17,31], a DLVM, learns a joint distribution distribution, $p_\theta(\mathbf{x}, \mathbf{z})$, that captures the relationship between a set of hidden variables, $\mathbf{z}$, and the observed variables, $\mathbf{x}$. VAEs model the data distribution, $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$, by optimizing the parameters, θ, typically a deep neural network known as the *generative model/decoder*. The VAE approximates the true posterior by a *surrogate* distribution, $q_\phi(\mathbf{z} \mid \mathbf{x})$, that informs the objective function to use a latent subspace

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_28.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 428–444, 2025.
https://doi.org/10.1007/978-3-031-78172-8_28

that is likely to maximize $p_\theta(\mathbf{x} \mid \mathbf{z})$. The parameters ($\phi$) of the surrogate posterior is another deep neural network known as the *inference model/encoder*. The encoder (ϕ) and decoder (θ) parameters of the VAE are jointly optimized to maximize the evidence lower bound (ELBO).

Despite strong theoretical foundations, the VAE fails in matching the *aggregate* posterior $q_\phi(\mathbf{z}) = \int q_\phi(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})d\mathbf{x}$ to the prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The mismatch between distributions results in *clusters* or *holes* in the latent space, indicating regions strongly supported under the prior may have low density under aggregate posterior [14,32] (and vice versa). The presence of *holes* increases the mismatch between the learned, $p_\theta(\mathbf{x})$, and real data distribution, $p(\mathbf{x})$, leading to the generation of low-quality samples. To alleviate this issue, methods in [2,42] use a flexible prior, and the authors of [4] match the prior in two stages. An additional regularization loss is added to the ELBO to match aggregate distributions that help in learning meaningful representations [43] and improved disentanglement of latent factors [16]. The estimation of $q_\phi(\mathbf{z})$ is challenging, thus the methods in [16,43] uses an adversarial classifier or a kernel method, e.g., MMD [9], to match the aggregate posterior to the prior, similar to the Wasserstein autoencoders [41].

The posterior distribution of the VAE might collapse to the prior for *a subset* or *all* of the latent dimensions during the training of VAEs. Under such scenarios, the representations produced by the encoder on the collapsed latent dimensions lack information that the decoder can use to reconstruct the input faithfully. This phenomenon is known as the *posterior collapse* or the *KL vanishing effect* [6,30,30]. We expect to encounter such degenerate solutions more often with the β-VAE [13] that advocates the use of higher β values for the improved disentanglement of the latent factors. The analysis in [14] explains the minimization of the mutual information $I(\mathbf{x}; \mathbf{z})$ between the latent ($\mathbf{z}$) and observed variables ($\mathbf{x}$) for higher β values. Several methods have been proposed to circumvent this issue, such as the KL annealing strategy [6,28], explicit inhibition of the distribution matching [30], use of complex priors [2,42], and special training policy [11].

In this work, we address the limitations of the VAE by matching the aggregate posterior to the prior in the ELBO framework derived from first principles. We use KDE in the latent space to model the aggregate posterior, $q_\phi(\mathbf{z})$. The use of KDE in the AVAE helps in a better estimate of differences between distributions relative to the adversarial training and kernel-based method used in [16,24,41,43]. In addition to improvement in the quality of the generated samples, matching the aggregate posterior to a prior finds potential application in the meaningful interpretation of the latent generative factors [16,40], outlier detection [35], and data completion [10,27]. Unlike other variants of the VAE that strive to match marginal posterior to the prior [16,40,43], the proposed method does not require additional regularization terms or hyperparameters to the objective function. Moreover, we propose a heuristic that automatically adjusts the β value during the training instead of empirically estimating the β for a dataset using cross-validation. The potential benefits of using KDEs for matching distributions have

been thoroughly studied in [35]. Though KDEs are used in [35] for matching the aggregate posterior distribution to the prior, the objective function is not derived in the general framework of DLVMs, and it is not well suited to high-dimensional latent spaces, e.g., ≥ 50, which restricts its application for modeling complex datasets, such as the CIFAR10 [18]. We correct the bias in KDE bandwidth estimation that qualifies the AVAE to use KDE in high-dimensional latent spaces (dimensions > 100).

The main contributions of this work are summarized as follows:

- Matching the aggregate posterior distribution to the prior in the VAE objective function using KDE without any modification of the ELBO.
- An automated method for estimating KDE bandwidth that allows using KDEs in high-dimensional latent spaces (dimensions > 100).
- Evaluations showing that the AVAE addresses the shortcomings in the formulation (*pockets/clusters*) and training (*posterior collapse*) of the VAE.
- The regularization scalar β is updated during training using the proposed heuristic. Thus, the AVAE is free from tuning the hyperparameter, β.
- Empirical evaluation of the proposed method using different efficacy measures on multiple benchmark datasets, producing results that compare favorably with state-of-the-art, likelihood-based, generative models.

2 Related Work

Several extensions to the formulation of the VAE address known limitations, such as alleviating posterior collapse [26,30], better matching of marginal posteriors [16,40], and reducing over-regularization [2,42]. Methods matching marginal posteriors are relevant to our work. These methods introduced an additional regularization term to the objective function [16,40] (along with a hyperparameter) to encourage statistical independence of latent factors. An interesting analysis of the VAE objective is done in [7] (RAE), which suggests that an autoencoder with regularized decoder parameters is as good as the VAE.

The generative adversarial network (GAN) is another popular generative model that implicitly matches distributions using a discriminator [8,29]. GANs produce novel, realistic examples, such as images with sharp and distinct features, which are difficult for even humans to identify as generated images [15]. Nevertheless, GANs do not produce a reliable matching form data samples into the latent space [5], and there are significant challenges in optimizing the objective function of a GAN [1,20,25]. GANs are very particular about the architecture of the discriminator, training strategy, and the associated hyperparameters [33,37]. The adversarial autoencoder (AAE) [24] is a likelihood-based generative model that implicitly matches the aggregate posterior in the latent space of an autoencoder to a prior with the help of a discriminator.

WAEs [41] is another likelihood-based generative model that explicitly matches the aggregate posterior to a prior in the latent space (unlike VAEs). In the WAE, the Wasserstein distance between the data and generated distribution is minimized by factoring the latent variable $\mathbf{z}$ in its formulation. The

regularization term in WAEs is computed using two different strategies. In one approach, a discriminator is used in the latent space, as in AAEs, and is known as the WAE-GAN. In the other approach, the maximum mean discrepancy (MMD) [9] is used to compute the divergence between distributions in the latent space, known as the WAE-MMD.

3 Method

3.1 Background

The goal of a DLVM is to learn the joint distribution of the latent variables, $\mathbf{z}$, and the observed variables, $\mathbf{x}$, such that the resulting (generative) distribution closely approximates the true but unknown data distribution, $p_{data}(\mathbf{x})$. In the DLVM, $p_\theta(\mathbf{x} \mid \mathbf{z})$ learns the mapping from the latent space to observed space using the samples generated by $p(\mathbf{z})$ and model parameters θ. This setup is used to generate new samples not present in the observed dataset. Thus, the aim is to determine the correct setting of the parameters, θ, such that the probability of each observed data, $p_\theta(\mathbf{x})$, is maximized. The objective function of the DLVM is defined as follows:

$$
\begin{aligned}
&\max_\theta \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \log p_\theta(\mathbf{x}) = \max_\theta \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \log \int p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\
&= \max_{\theta,q} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \log \int \left(p_\theta(\mathbf{x} \mid \mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} \right) q(\mathbf{z}) d\mathbf{z} \quad \begin{array}{l}\text{Expectation under the} \\ \text{proposal distribution, } q(\mathbf{z})\end{array} \\
&= \max_{\theta,q} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \log \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})} \left(p_\theta(\mathbf{x} \mid \mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} \right)
\end{aligned} \tag{1}
$$

by Jensen's inequality, we get

$$
\begin{aligned}
&\geq \max_{\theta,q} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})} \log \left(p_\theta(\mathbf{x} \mid \mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} \right) \\
&= \max_{\theta,q} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})} \left\{ \log \left(p_\theta(\mathbf{x} \mid \mathbf{z}) \right) - \log \left(\frac{q(\mathbf{z})}{p(\mathbf{z})} \right) \right\} \\
&= \max_{\theta,q} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \left\{ \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})} \log \left(p_\theta(\mathbf{x} \mid \mathbf{z}) \right) - \mathrm{KL}\left(q(\mathbf{z}) || p(\mathbf{z}) \right) \right\}.
\end{aligned} \tag{2}
$$

The objective function defined in 2 gives a lower bound on the data log likelihood and is known as the evidence lower bound (ELBO).

Use of $q(\mathbf{z} \mid \mathbf{x})$ as the proposal distribution in Eq. 1 gives us the objective function of the VAE [17,31]. The choice of the probability distribution for q is a modeling choice, and for VAEs, it is typically a Gaussian distribution [17,31]. The VAE uses an inference network (also called a recognition model), $q_\phi(\mathbf{z} \mid \mathbf{x})$, a deep neural network parameterized by ϕ that estimates the parameters of the Gaussian distribution for any input $\mathbf{x}_i$, $\phi : \mathbf{x}_i \rightarrow (\boldsymbol{\mu}_{\mathbf{x}_i}, \boldsymbol{\sigma}^2_{\mathbf{x}_i} \mathbf{I})$.

Matching the conditional distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ to $p(\mathbf{z})$ in VAEs often fails to match the aggregate posterior in the latent space [14,32]. The mismatch leads to, among other things, *holes* or *pockets* in the latent distribution that subsequently affects the quality of the generated samples. Increasing the strength of the regularization term in the objective function of VAEs does not help better match the aggregate posterior to the prior [13]. Instead, it results in a scenario known as *posterior collapse* [22,23], where the conditional distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$ matches to the prior $p(\mathbf{z})$ for a subset of the latent dimensions. Such degenerate solutions produce latent encodings that are no longer meaningful, and the decoder tends to ignore $\mathbf{z}$ in the reproduced observed data, resulting in poor reconstruction. This phenomenon is related to the identity, $\mathrm{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x})||p(\mathbf{z})\right) = I(\mathbf{x};\mathbf{z}) + \mathrm{KL}\left(q_\phi(\mathbf{z})||p(\mathbf{z})\right)$, where the $I(\mathbf{x};\mathbf{z})$ is the mutual information between the observed and latent variables. Thus, increasing the strength of the KL term would lead to better aggregate posterior matching but would lower the mutual information between the latent variables and the data. Several variants are proposed [4,16,40] to circumvent these issues encountered in the VAE that emphasizes matching the aggregate posterior to a prior.

3.2 Aggregate Variational Autoencoder (AVAE)

Instead of parametric distribution on the conditional probability, as used in VAEs, we propose to represent the aggregate distribution using kernel density estimation (KDE). KDE used to approximate the aggregate posterior distribution is defined as:

$$q(\mathbf{z}) = \frac{1}{m}\sum_{i=1}^{m} K\left(\frac{||\mathbf{z} - \mathbf{z}_i^{'}||}{h}\right). \tag{3}$$

Using KDEs, the probability at $\mathbf{z}$ for the proposal distribution is estimated using m KDE samples, $\mathbf{z}_i^{'}$, and the kernel, K, with an associated bandwidth, $h \in \mathbb{R}^+$. We use a random subset of the training data, $\mathcal{X}^{kde}$, that is shuffled every epoch to produce KDE samples in the latent space, $\mathbf{z}_i^{'} = \mathbf{E}_\phi(\mathbf{x}_i^{'})$, where $\mathbf{x}_i^{'} \in \mathcal{X}^{kde}$ and $\mathbf{E}_\phi$ is a deep neural network parameterized by ϕ, known as the *encoder*. We use a deterministic encoder (ignoring the variances along the latent axes), unlike VAEs. Through multiple empirical evaluations, we show that using a deterministic encoder in the AVAE does not rob it of expressive power compared to a regular VAE or its variants.

The ELBO objective function using KDE-based proposal distribution $q_\phi(\mathbf{z})$ is defined as follows:

$$\max_{\theta,\phi} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left\{\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z})} \log\left(p_\theta(\mathbf{x} \mid \mathbf{z})\right) - \mathrm{KL}\left(q_\phi(\mathbf{z})||p(\mathbf{z})\right)\right\}. \tag{4}$$

Equation 4 gives us the objective function of the AVAE. In comparison to the proposal distribution in VAEs, KDE-based approximation matches the aggregate posterior, $q_\phi(\mathbf{z})$, to the prior, $p(\mathbf{z})$, *without any modifications* to the ELBO formulation. Compared to the β-TCVAE [40], the AVAE does not have a mutual

information (MI) term in its objective function. The absence of the MI in the AVAE also reduces the number of hyperparameters.

The random (data) variable $\mathbf{x}$ typically exists in high dimensions, and thus the probability of $p_\theta(\mathbf{x} \mid \mathbf{z})$ is valid only for a small region in the latent space, i.e., $p_\theta(\mathbf{x} \mid \mathbf{z})$ is nonzero for a small region in the latent space. We use $\mathbf{z} = \mathbf{E}_\phi(\mathbf{x})$ as an estimate to maximize $\log p_\theta(\mathbf{x} \mid \mathbf{z})$ in Eq. 4. Considering this modeling choice, the objective function of the AVAE then becomes:

$$\max_{\theta,\phi} \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \left\{ \log \left(p_\theta(\mathbf{x} \mid \mathbf{E}_\phi(\mathbf{x})) \right) - \mathrm{KL} \left(q_\phi(\mathbf{z}) || p(\mathbf{z}) \right) \right\}. \tag{5}$$

We use the multivariate Gaussian distribution or Bernoulli distribution as the conditional likelihood distribution, $p_\theta(\mathbf{x} \mid \mathbf{z})$ in 5, depending on the dataset. The parameters of the chosen distribution are estimated using another neural network known as the *decoder*, $\mathbf{D}_\theta$, parameterized by θ. The objective function in 5 is optimized using the stochastic gradient descent (SGD) that jointly updates the encoder and decoder parameters, ϕ and θ, respectively. The first term in the objective function tries to reproduce the input as closely as possible using the corresponding latent statistics (reconstruction loss), while the KL term (matching the aggregate posterior to the prior) regularizes the model parameters.

The objective function of the AVAE is similar to that of WAEs, which have a reconstruction term and a divergence penalty on the aggregate distribution over latent representations. The divergence measure regulates the trade-off between the reconstruction and latent regularization loss. Similar to WAEs, the AVAE has the flexibility in choosing reconstruction cost terms by considering different distributions for $p_\theta(\mathbf{x} \mid \mathbf{z})$. The divergence penalty in the AVAE is the KL divergence, a particular case of the WAE. Nevertheless, the AVAE has provable statistical properties of the latent space, and the proposed method has empirically demonstrated its merit over the WAE under several evaluation metrics discussed in subsequent sections.

Training: The objective function of the AVAE defined in 5 has two terms: the reconstruction loss and KL-divergence-matching of the aggregate posterior to the prior. The aggregate posterior, $q_\phi(\mathbf{z})$, in the AVAE is represented using KDE. A random subset of the training data $\mathcal{X}^{kde} \in \mathcal{X}^{train}$ forms KDE samples that is shuffled after every epoch. Remaining samples $\mathcal{X}^{sgd} = \mathcal{X}^{train} - \mathcal{X}^{kde}$ are used for optimizing the objective function 5 using the SGD that updates the model parameters, ϕ and θ. Shuffling KDE samples in $\mathcal{X}^{kde}$ in every epoch changes the aggregate posterior, $q_\phi(\mathbf{z})$, used in the AVAE objective function. However, the evolving aggregate posterior does not impact (adversely) the training of the AVAE, and the loss curves on multiple datasets demonstrate the stable optimization of the AVAE objective function (refer to Fig. 1 in the supplementary). Moreover, an update of the $\mathcal{X}^{kde}$ and $\mathcal{X}^{sgd}$ in every epoch results in better performance of the AVAE (compared with a fixed $\mathcal{X}^{kde}$) under different metrics across datasets (refer to Table 1 in the supplementary).

Without any loss of generality, we use the isotropic Gaussian kernel in KDE for this work, which introduces a bandwidth parameter. There are many heuris-

Algorithm 1 : AVAE training using stochastic gradient descent

Input: Data $\mathcal{X}$, Latent dimensions l, KDE samples m.

Estimate the optimal kernel bandwidth $h_{\text{opt}}^{\text{corr}}$ given (l, m)
Split $\mathcal{X}$ into training, $\mathcal{X}^{train}$, and validation data, $\mathcal{X}^{val}$
Random samples for KDE, $\mathcal{X}^{kde}$, and SGD samples, $\mathcal{X}^{sgd} = \mathcal{X}^{train} - \mathcal{X}^{kde}$
Random initialization of the encoder (ϕ) and decoder (θ) parameters
Get KDE samples, $\mathbf{z}_i^{'} = E_\phi(\mathbf{x}_i^{'})$, where $\mathbf{x}_i^{'} \in \mathcal{X}^{kde}$ for the aggregate posterior, q_ϕ
Initialize $\beta \leftarrow \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \|\mathbf{x}_i^{''} - \hat{\mathbf{x}}_i^{''}\|_2$, where $\mathbf{x}_i^{''} \in \mathcal{X}^{val}$
for number of epochs **do**
  for number of minibatch updates **do**
    Sample a minibatch of size n_b from $\mathcal{X}^{sgd}$, $\mathcal{X}_b^{sgd} = \{\mathbf{x}_1^{''}, \ldots \mathbf{x}_{n_b}^{''}\}$
    Encode samples $\mathcal{Z}_b^{sgd} = \{\mathbf{z}_1^{''}, \ldots \mathbf{z}_{n_b}^{''}\}$, where $\mathbf{z}_i^{''} = E_\phi(\mathbf{x}_i^{''})$
    Update the encoder and decoder parameters, ϕ and θ, respectively, by optimizing the objective function in 5 using stochastic gradient descent
    Update KDE samples, $\mathbf{z}_i^{'} = E_\phi(\mathbf{x}_i^{'})$ using the current state of the encoder
  end for
  $\beta \leftarrow \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \|\mathbf{x}_i^{''} - \hat{\mathbf{x}}_i^{''}\|_2$, where $\mathbf{x}_i^{''} \in \mathcal{X}^{val}$
  New samples for $\mathcal{X}^{kde}$ chosen at random and update $\mathcal{X}^{sgd} = \mathcal{X}^{train} - \mathcal{X}^{kde}$
  Produce latent encoding $\mathbf{z}_i^{'} = E_\phi(\mathbf{x}_i^{'})$ for the aggregate posterior, q_ϕ, using KDE
end for

tics for estimating the kernel bandwidth used in KDE, and there is no established solution for unknown distributions. Furthermore, the estimation of KDE bandwidth is particularly challenging in high-dimensional latent spaces (dimensions > 50). We present a bandwidth estimation method in Sect. 3.4 that uses the knowledge of the prior distribution, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, to estimate KDE bandwidth for a given latent dimension and a given number of KDE samples. The proposed bandwidth estimation technique can even scale to higher dimensional latent spaces, i.e., dimensions > 100.

Several extensions of the VAE [6,28,34,38] propose automated ways to determine the hyperparameter β that balances the loss terms in the objective function. In a similar vein, we propose a data-driven technique to determine β that balances the loss terms in the AVAE objective function. An outline of the training of the AVAE is presented in Algorithm 1.

Estimation of β: The objective function of the standard VAE does not introduce a hyperparameter to weigh the loss terms. However, it is a common practice to assign weights to different terms in the objective functions [7,24,41] for various reasons, such as stability in optimization and application-specific trade-offs. Likewise, several variants of the VAE [13,40] use a hyperparameter, β, to control the contribution of the loss terms in the objective function. It is often challenging to decide the appropriate value of these hyperparameters for a particular model architecture, dataset, and other related settings for optimization. The widely used strategy under these circumstances is to set the hyperparameter value using cross-validation.

To alleviate these issues, methods proposed in [34,38], among others, have devised automated strategies to determine β. The method in [38] uses a PI controller that manipulates the value of β as the learning progresses. Assuming the decoder predicts the parameter of the multivariate Gaussian distribution, [34] presents two approaches to learning the Gaussian variance, σ (equivalent to learning β). In the first approach, an additional parameter is trained with the encoder-decoder parameters to learn the trade-off factor, σ. In another approach, the maximum likelihood estimate (MLE) determines the variance analytically.

Similar to these approaches, the proposed AVAE optimization sets beta β to weight the gradient of the regularization term relative to the reconstruction loss:

$$\beta \leftarrow \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \|\mathbf{x}_i^{''} - \hat{\mathbf{x}}_i^{''}\|_2, \tag{6}$$

where $\mathbf{x}_i^{''}$ is an example in the validation set, $\mathcal{X}^{val} \in \mathcal{X}$, and $\hat{\mathbf{x}}_i^{''}$ is the corresponding reconstructed sample produced by the decoder. Relative to [38], the proposed approach is simple yet effective, as demonstrated by the empirical evaluations. The update of β during the training of the AVAE on multiple datasets is reported in the supplementary (Fig. 1). Moreover, this formulation can be extended to any distribution chosen for the log conditional likelihood, $p(x \mid z)$, rather than being limited to only a Gaussian, as in [34].

3.3 Properties of the Aggregate Posterior of the AVAE

Considering the standard normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, as the prior distribution, $p(\mathbf{z})$, we analytically derive the expected aggregate posterior distribution of a trained AVAE. For a trained AVAE model, we assume the gradient of the objective function (5) w.r.t to latent encodings, $\mathbf{z}_i^{''}$'s (refer to Algorithm 1), is zero. In our analysis, we consider only the KL divergence term in the objective function. Setting the derivative of the $\mathrm{KL}(q_\phi(\mathbf{z}^{''})||p(\mathbf{z}^{''}))$ to 0, we derive the same expression as in equation 5 of [35]. Following the steps in [35], we prove the aggregate posterior distribution of the AVAE is $\mathcal{N}\left(\mathbf{0}, \mathbf{I}(1-h^2)\right)$, in expectation, where h is KDE bandwidth. The proof is also consistent with the known properties of KDEs generally—KDEs introduce a bias that is characterized by a convolution of the kernel with the underlying distribution.

3.4 KDE Bandwidth Estimate

Estimating KDE bandwidth can be challenging, and solutions in the literature are often related to particular applications. Many heuristics are proposed for bandwidth estimation under general circumstances [39]. However, here, we use the knowledge of the prior distribution, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, to our advantage for estimating KDE bandwidth used for modeling the aggregate distribution in the latent space, $q_\phi(\mathbf{z})$. We devise an objective function such that the empirical aggregate distribution, $q_\phi(\mathbf{z})$, in the latent space approaches the target distribution, $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, as the system converges. Thus, we set the kernel bandwidth

h_{opt} to minimize the KL divergence between the analytical prior distribution and KDE of a finite set of samples from the prior distribution, as follows:

$$h_{\text{opt}} = \min_{h} \text{KL}\left(p(\mathbf{z})||q_\phi(\mathbf{z})\right) = \max_{h} \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})} q_\phi(\mathbf{z}), \tag{7}$$

with latent dimension, l, and a number of KDE samples, m. In this optimization problem, we use samples from the $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ such that the probability of the samples is maximized w.r.t the aggregate posterior, $q_\phi(\mathbf{z})$. Table 1 reports the optimum bandwidth, h_{opt}, for different scenarios. We use gradient-based optimizers, such as Adam, to learn the single parameter, h_{opt} in 7. We observe in Table 1 that for higher latent dimensions with limited KDE samples (e.g., starting at $l = 40$ with $m = 500$), the optimal bandwidth is greater than the standard deviation of the prior distribution, $h_{\text{opt}} > 1.0$. Given the known bias KDE introduces in the AVAE optimization, optimizing the encoder under these conditions would degenerate to samples converging at the origin (posterior collapse).

Given h_{opt}, we know from Sect. 3.3 that the distribution in the latent space of the AVAE converges to $\mathcal{N}\left(\mathbf{0}, \mathbf{I}(1 - h^2_{\text{opt}})\right)$, where $(1 - h^2_{\text{opt}})$ is bias introduced by KDE. However, we could not consider $\mathcal{N}\left(\mathbf{0}, \mathbf{I}(1 - h^2_{\text{opt}})\right)$ as the target distribution, $p(\mathbf{z})$, for optimization of the objective function in 7, as h_{opt} is unknown. We hypothesize that this is one of the reasons for the optimal bandwidth h_{opt} to be greater than the standard deviation of the prior distribution in bigger latent spaces (Table 1). Thus, we must factor in the bias, $(1 - h^2_{\text{opt}})$, introduced by KDE to estimate the bandwidth. To this end, we propose to use a scaled version of the target distribution, $\mathcal{N}\left(\mathbf{0}, \alpha^2\mathbf{I}\right)$, for optimization of the objective function in 7, where the scaling factor α is unknown. We need to estimate the optimum bandwidth for $\mathcal{N}\left(\mathbf{0}, \alpha^2\mathbf{I}\right)$. Given h_{opt} as the optimum bandwidth for $\mathcal{N}(\mathbf{0}, \mathbf{I})$, the estimated bandwidth for $\mathcal{N}\left(\mathbf{0}, \alpha^2\mathbf{I}\right)$ is αh_{opt} by linear property of the Gaussian distribution. Moreover, we know (from Sect. 3.3) that with αh_{opt} as KDE bandwidth, the latent distribution of the AVAE would have a bias, $1 - (\alpha h_{\text{opt}})^2$, at convergence. We use this property to solve for the scaling factor, α, where we set the variance equal to the bias, $\alpha^2 = 1 - (\alpha h_{\text{opt}})^2$, to get the scaling that accounts for both the ideal optimal bandwidth and the bias:

$$\alpha^2 = \frac{1}{1 + h^2_{\text{opt}}}. \tag{8}$$

This simple but elegant strategy of handling the bias in KDE addresses the challenge of estimating KDE bandwidth in high-dimensional latent spaces. Notice that because $0 \leq \alpha \leq 1.0$, we avoid mode collapse because the system only degenerates ($h_{\text{opt}} \to \infty$) as the number of samples goes to zero or the dimensionality goes to infinity.

With the *bias scaling factor*, α, we get estimates of the *bias-corrected* KDE bandwidth ($h^{\text{corr}}_{\text{opt}} = \alpha * h_{\text{opt}}$) reported in Table 1, which are the scaled versions of the optimum bandwidth h_{opt} (Eq. 7). In the revised estimate, the optimal bandwidth is less than the standard deviation of the prior distribution, $h^{\text{corr}}_{\text{opt}} < 1.0$, for all dimensions in Table 1, as expected. The bias-corrected bandwidth

Table 1. Optimal bandwidths, h_{opt} (estimated using the objective function defined in 7) and corresponding bias-corrected estimations $h_{\text{opt}}^{\text{corr}}$ (h_{opt} scaled by the factor α) for a given latent dimension (l) and number of KDE samples (m). The estimated bandwidth increases with increasing dimensions (vertical) and decreases with increasing sample size (horizontal). For higher latent dimensions with limited KDE samples (e.g., starting at $l = 40$ with $m = 500$), $h_{\text{opt}} > 1.0$. However, the bias-corrected bandwidth $h_{\text{opt}}^{\text{corr}} < 1.0$.

	$m = 500$		$m = 1000$		$m = 2000$		$m = 5000$		$m = 10000$	
l	h_{opt}	$h_{\text{opt}}^{\text{corr}}$	h_{opt}	$h_{\text{opt}}^{\text{corr}}$	h_{opt}	$h_{\text{opt}}^{\text{corr}}$	h_{opt}	$h_{\text{opt}}^{\text{corr}}$	h_{opt}	$h_{\text{opt}}^{\text{corr}}$
10	0.74	0.60	0.70	0.58	0.67	0.56	0.63	0.53	0.60	0.51
20	0.89	0.67	0.86	0.65	0.84	0.64	0.80	0.62	0.78	0.61
40	> 1.0	0.72	> 1.0	0.71	0.98	0.70	0.95	0.69	0.93	0.68
50	> 1.0	0.73	> 1.0	0.72	> 1.0	0.71	0.99	0.70	0.98	0.70
70	> 1.0	0.74	> 1.0	0.74	> 1.0	0.73	> 1.0	0.73	> 1.0	0.72
100	> 1.0	0.76	> 1.0	0.75	> 1.0	0.75	> 1.0	0.74	> 1.0	0.74

encourages the use of KDEs in bigger latent spaces (e.g., dimensions ≥ 50) that makes the AVAE appropriate for modeling complex datasets (a limitation in the previous KDE-based aggregate matching [35]).

4 Experiments

4.1 Experimental Setup

Benchmark Methods: In comparisons, we consider the conventional VAE [17] and other variations of VAE that modify the original formulation in an attempt to match the aggregate posterior to the prior [16,40]. Among others, β-TCVAE [40] is the closest to the AVAE formulation, as the objective function does not introduce any additional, ad-hoc loss terms. The RAE [7] is chosen as one of the baseline models due to its performance on multiple benchmark datasets. Other maximum likelihood-based models such as the AAE [24] and WAE [41] match aggregate posterior in the latent space of a deterministic autoencoder. The AAE [24] implicitly matches aggregate distributions using a discriminator in the latent space. We use the WAE-MMD (with IMQ kernel) in our analysis due to the stability in training. We study the VAE [17], β-TCVAE [40], RAE [7], AAE [24], and WAE-MMD [41] as competing methods to the AVAE.

Evaluation Metrics: Ideally, the evaluation of a DLVM should include a comparison of the model's data distribution and that of the true data. Of course, this is infeasible because true data distribution is unknown. Many methods use the quality of the samples produced by the models in the observed space as a proxy for the actual distribution. In this work, we use the Fréchet Inception Distance (FID) [12] to quantify the quality of the samples. In addition, we evaluate the data distributions learned by different models using the precision-recall metric

Table 2. FID [12], and precision-recall [36] scores of competing methods. The **best** score is in **bold**, and the second best score is underlined.

Methods	MNIST ($l = 16$)			CelebA ($l = 64$)			CIFAR10 ($l = 128$)		
	FID ↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑	FID↓	Precision↑	Recall↑
VAE	28.78 ± 0.48	0.88 ± 0.04	0.97 ± 0.00	49.89 ± 0.57	0.79 ± 0.03	0.75 ± 0.03	147.74 ± 0.81	0.50 ± 0.03	0.47 ± 0.02
β-TCVAE	50.62 ± 1.19	0.82 ± 0.03	0.95 ± 0.01	50.14 ± 0.78	0.78 ± 0.02	0.70 ± 0.05	180.94 ± 1.16	0.30 ± 0.01	0.41 ± 0.03
RAE	18.79 ± 0.31	0.87 ± 0.01	0.95 ± 0.02	48.81 ± 1.02	0.81 ± 0.02	0.77 ± 0.04	94.34 ± 1.58	$\mathbf{0.74 \pm 0.02}$	0.47 ± 0.04
AAE	19.51 ± 1.77	0.85 ± 0.03	0.96 ± 0.01	49.32 ± 0.25	0.86 ± 0.01	0.75 ± 0.03	100.00 ± 1.40	0.71 ± 0.03	0.56 ± 0.04
WAE	25.42 ± 1.19	$\mathbf{0.92 \pm 0.03}$	0.92 ± 0.01	72.01 ± 2.26	0.64 ± 0.05	0.75 ± 0.02	140.49 ± 0.64	0.42 ± 0.01	0.31 ± 0.05
AVAE	$\mathbf{13.27 \pm 0.34}$	$\mathbf{0.92 \pm 0.02}$	$\mathbf{0.98 \pm 0.00}$	$\mathbf{46.0 \pm 0.42}$	$\mathbf{0.88 \pm 0.02}$	$\mathbf{0.85 \pm 0.02}$	$\mathbf{90.93 \pm 6.65}$	0.72 ± 0.05	$\mathbf{0.67 \pm 0.04}$

[36], where the precision evaluates the quality of the generated samples, and the recall assesses whether the model data distribution captures the variations present in the original but unknown data distribution. Besides the attributes of the model data distribution, we evaluate the properties of the latent representations of the competing methods. In particular, we are interested in the presence of holes in the latent distribution, and we use *entropy* of the aggregate posterior distribution as an indicator of holes/clusters. We train each method 5 times on a dataset for all empirical evaluations, initialized differently in every run.

Datasets: We use several popular benchmark datasets, MNIST [19], CelebA [21], and CIFAR10 [18] for empirical evaluation of different methods. To address the dataset's complexity, the size of the latent space, neural network architectures, model-specific hyperparameters, and other optimization parameters are altered accordingly. Details of the neural network architectures and other parameter settings for all the benchmark datasets used by the competing methods are reported in Sects. 2.1 and 2.2 in the supplementary material.

4.2 Results

Evaluation of the Model Data Distribution. We quantitatively evaluate the generated samples in this experiment using the FID scores [12] on multiple benchmark datasets. A lower FID score indicates better matching of the data distributions. Besides the FID metric, we evaluate the diversity and quality of the generated samples using the precision-recall metric [36]. A higher precision indicates good quality of the generated samples, and a higher recall suggests that the model data distribution covers the modes present in the true data distribution. Except for the RAE, all the methods considered in this experiment use $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as the prior distribution. For the RAE, we approximate the distribution in the latent space by the Gaussian distribution. Parameters of the Gaussian distribution derived from the latent representations are used to generate new data samples. We know that the latent distribution for the AVAE convergences to $\mathcal{N}\left(\mathbf{0}, \mathbf{I}(1-h^2)\right)$, where h is KDE bandwidth (Sect. 3.3). Therefore, we use samples drawn from the distribution, $\mathcal{N}\left(\mathbf{0}, \mathbf{I}(1-h^2)\right)$, to evaluate the generative capability of AVAEs. For a fair comparison, we have used the hyperparameter settings suggested by the author or recommended in the literature.

Table 3. Mean entropy of the $q_\phi{}^{\mathrm{w}}(\mathbf{z})$ produced by competing methods on the benchmark datasets. The **best** score is in **bold**, and the second best score is underlined. The entropy of the standard normal distribution is used as the ground truth.

Method	MNIST ($l = 16$) ↑	CelebA ($l = 64$) ↑	CIFAR10 ($l = 128$) ↑
VAE	4.71 ± 0.14	28.88 ± 0.03	29.56 ± 0.39
β-TCVAE	3.44 ± 0.39	28.42 ± 0.05	15.02 ± 0.69
RAE	4.89 ± 0.06	27.08 ± 0.09	49.92 ± 0.31
AAE	5.81 ± 0.18	30.49 ± 0.03	54.21 ± 0.36
WAE	4.67 ± 0.09	28.32 ± 0.05	48.10 ± 0.45
AVAE	$\mathbf{7.56 \pm 0.10}$	$\mathbf{30.96 \pm 0.02}$	$\mathbf{55.64 \pm 0.00}$
Standard Normal	8.00	32.00	64.00

The FID and precision-recall scores are reported in Table 2. The VAE does reasonably well for the MNIST and CelebA datasets. However, its performance drops significantly for the complex CIFAR10 dataset. Despite the importance given to match the aggregate posterior in the β-TCVAE ($\beta = 2$ for comparable reconstruction loss), it fails to address the shortcomings of the VAE. Furthermore, the performance of the β-TCVAE is poorer than the regular VAE. These results manifest the limitations in the formulation of the VAE (objective function and modeling assumptions) to model the data distributions. Other DLVMs (AAE, WAE, and AVAE) *matching the aggregate posterior* to the prior using a deterministic autoencoder do better than VAEs, in general. The AAE (aka WAE-GAN) closely follows the best performing methods under different evaluation metrics. We hypothesize that the kernel-based method used in WAE (WAE-MMD) to evaluate the mismatch between distributions is possibly leading to poor performance (justified by low entropy scores in Table 3), as the reconstruction error is comparable to all other methods (refer to the MSE per pixel in Table 5 of the supplementary material). The performance of the WAE gets worse for the CIFAR10 dataset using high-dimensional latent space $l = 128$. The performance of the RAE is promising across all datasets under different evaluation scenarios. The generative capability of the AVAE is the best among all the considered methods for all the benchmark datasets under different evaluation metrics studied in this work, except for the precision on the CIFAR10 dataset (the second best). It is important for any generative model to capture the modes present in a dataset, indicated by high recall scores. The AVAE consistently outperforms other methods under the recall metric, resulting in the best FID scores under all evaluation scenarios.

We investigate the poor performance of the VAE and β-TCVAE on the MNIST and CIFAR10 datasets. Other than the CelebA dataset, we observe the reconstruction loss of the VAE and β-TCVAE to be relatively higher than other methods (refer to Table 5 in the supplementary material). On further analysis, we discovered that both the VAE and β-TCVAE suffer from the posterior collapse when trained on the MNIST and CIFAR10 datasets (refer to Sect. 2.3 in

the supplementary). For the MNIST dataset, 4 and 7 (out of 16) latent dimensions collapsed for the VAE and β-TCVAE, respectively. Collapsed dimensions reduce the bottleneck capacity of a DLVM, resulting in higher reconstruction loss. The posterior collapse subsequently impairs the VAE and β-TCVAE to model the data distributions, leading to the worst FID scores for the β-TCVAE on the MNIST and CIFAR10 datasets, followed by the VAE.

Entropy of the Aggregate Posterior Distribution. In this experiment, we evaluate deviations of the resultant aggregate distribution, $q_\phi(\mathbf{z})$, beyond the second moment (other than the mean and covariance), as we would expect from holes or clusters in the distribution. For this, we use the entropy of the aggregate posterior distribution to quantify how close it is to Gaussian, *after whitening the distribution*, $q_\phi^{\mathrm{w}}(\mathbf{z})$, to remove the effects of the second moment mismatch. Because the Gaussian distribution has the maximum entropy (for a given mean and covariance), we use the entropy of the whitened data. Entropy is defined as

$$H(\mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim q_\phi^{\mathrm{w}}(\mathbf{z})}\left\{-\log\left(q_\phi^{\mathrm{w}}(\mathbf{z})\right)\right\} \approx \frac{1}{m}\sum_j \frac{1}{m-1}\sum_{i\neq j} K\left(\frac{||\mathbf{z}_j - \mathbf{z}_i^{'}||}{h}\right), \quad (9)$$

where $q_\phi^{\mathrm{w}}(\mathbf{z})$ is the aggregate posterior distribution over the whitened data. We use KDE (defined in 3) for estimating the density $q_\phi^{\mathrm{w}}(\mathbf{z})$ for all methods because it can, in principle model the deviations we are seeking to evaluate. The bandwidth h required in KDE for the latent dimensions $l = \{16, 64, 128\}$ (for different datasets) and KDE samples $m = 10K$ is derived using the strategy defined in Sect. 3.4. The entropy computation uses the held-out set of the datasets studied in this work. The entropy of the standard normal distribution (leaving out the constants) derived analytically serves as the ground truth.

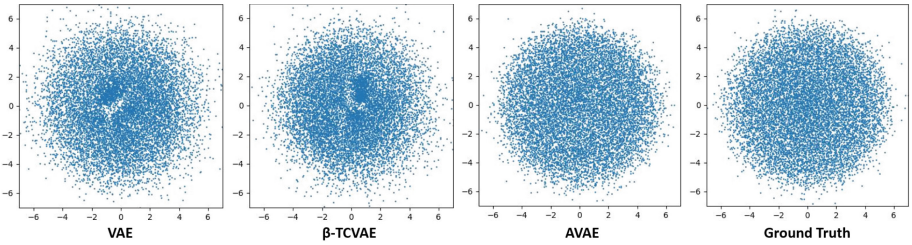


Fig. 1. The metric multidimensional scaling (mMDS) [3] plot in 2D of the latent representations ($\mathcal{Z} \in \mathbb{R}^{16}$) produced by the VAE [17], β-TCVAE [40] and the AVAE (proposed method) on the MNIST dataset [19]. Samples from the target distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, are used as the ground truth. The regions of low probability and unwanted aggregation of data points in different parts of the latent space of the VAE and β-TCVAE clearly show the mismatch with the ground truth. The AVAE closely matches the target distribution corroborated by empirical evaluations.

Table 4. Comparison of the performance of the AVAE with **different** number of KDE samples under multiple metrics for the MNIST and CAIFAR10 datasets.

KDE samples	MNIST ($l = 16$)				CIFAR10 ($l = 128$)			
	FID ↓	Precision↑	Recall↑	MSE↓	FID↓	Precision↑	Recall↑	MSE↓
1000	14.26 ± 0.23	0.89 ± 0.04	0.97 ± 0.00	0.0033 ± 0.0002	95.07 ± 5.52	0.75 ± 0.05	0.66 ± 0.03	0.0083 ± 0.0032
2000	13.09 ± 0.73	0.88 ± 0.03	0.97 ± 0.01	0.0034 ± 0.0002	99.88 ± 15.32	0.70 ± 0.11	0.67 ± 0.02	0.0064 ± 0.0002
5000	13.64 ± 1.24	0.89 ± 0.02	0.98 ± 0.01	0.0048 ± 0.0010	103.89 ± 5.21	0.59 ± 0.04	0.68 ± 0.04	0.0065 ± 0.0003
10000	13.27 ± 0.34	0.92 ± 0.02	0.98 ± 0.00	0.0041 ± 0.0004	90.93 ± 6.65	0.72 ± 0.05	0.67 ± 0.04	0.0062 ± 0.0002

From the results reported in Table 3, we observe that the entropy scores of the VAE and β-TCVAE are far off from the ground truth for the MNIST and CIFAR10 datasets even using the whitened latent representations. Low entropy scores of the VAE and β-TCVAE can be attributed to the formation of clusters as observed in Fig. 1 (refer to Sect. 2.3 in the supplementary for more results). Besides the posterior collapse, the entropy scores offer another perspective to explain the high FID scores of the generated samples produced by the VAE and β-TCVAE for the MNIST and CIFAR10 datasets. Poor FID scores of the WAE can be related to the low entropy values across datasets. The low entropy scores of the RAE are not surprising because it does not attempt to match any prior distribution in the latent space. However, the regularization approach in the RAE is more effective than the VAE. The AAE has entropy scores comparable to the AVAE, and it also helps us comprehend the consistent FID scores of the generated samples. The best entropy score of the AVAE for all the datasets indicates the close matching of the aggregate posterior to the prior, as shown in Fig. 1, where we do not observe clustering of the latent representations.

Ablation Study. In this experiment, we study the effect of the number of KDE samples on the performance of the AVAE under different evaluation metrics. The number of KDE samples used in the ablation study is $m = 1K, 2K, 5K$, and $10K$ for the MNIST and CIFAR10 datasets. We report the FID, precision-recall scores, and the reconstruction loss, i.e., the mean squared error (MSE) per pixel in Table 4. The AVAE produces comparable results with *a very few* KDE samples, $m = 1000$, even in high-dimensional latent space ($l = 128$ for the CIFAR-10 dataset). The stable optimization of the AVAE objective function with fewer KDE samples, such as $m = \{1K, 2K\}$ for the MNIST and CIFAR10 datasets, corroborates the accuracy and robustness of the proposed KDE bandwidth estimation technique. Overall, the performance of the AVAE under multiple metrics is slightly better with higher KDE samples. Therefore, we use $m = 10K, 20K$, and $10K$ for the MNIST, CelebA, and CIFAR10 datasets for all the evaluations reported in the paper.

5 Conclusion

We propose a novel algorithm, the aggregate VAE (AVAE), based on the framework of the VAE to match the aggregate posterior distribution to the prior

using KDE. Using the known properties of the prior distribution, we devised a method to estimate KDE bandwidth in high-dimensional latent spaces (dimensions > 100) that allows the modeling of complex datasets using the AVAE. The dynamic adjustment of the scaling factor, β, using the validation data avoids the hyperparameter tuning using cross-validation. The training of the AVAE does not suffer from the posterior collapse, as in VAEs and β-TCVAEs, and we avoid such failures without the modification of the ELBO formulation [2,30,42] and use of any complex training schedules [6,11,28]. We demonstrate the efficacy of the proposed method on multiple datasets, and the AVAE consistently outperforms the competing methods under different evaluation metrics. Close matching of the aggregate latent distribution to the prior with comparable reconstruction loss resulted in the best FID, precision, and recall scores for the AVAE. High entropy scores for the AVAE indicate that the latent representations are close to Gaussian and have a lower chance of encountering holes/clusters in the distribution. Through extensive empirical evaluation, we demonstrate the effectiveness of KDE in matching distributions in high-dimensional latent spaces compared to other methods, such as the kernel-based method used in the WAE-MMD and the discriminator in the AAE. In the AVAE, the cardinal latent axes do not represent the generative factors, unlike the regular VAE, due to matching the aggregate posterior to isotropic Gaussian, invariant to rotation. We plan to study this issue and devise a statistical method to identify the latent explanatory vectors.

References

1. Barnett, S.A.: Convergence problems with generative adversarial networks (GANs) (2018). preprint at https://arxiv.org/abs/1806.11382
2. Bauer, M., Mnih, A.: Resampled priors for variational autoencoders. AISTATS (2019)
3. Cox, T., Cox, M.: Multidimensional Scaling. Chapman Hall, Boca Raton, London (2001)
4. Dai, B., Wipf, D.: Diagnosing and enhancing VAE models. ICLR (2019)
5. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. In: ICLR (2017)
6. Fu, H., Li, C., Liu, X., Gao, J., Celikyilmaz, A., Carin, L.: Cyclical annealing schedule: a simple approach to mitigating kl vanishing. In: NAACL-HLT (2019)
7. Ghosh, P., Sajjadi, M.S.M., Vergari, A., Black, M., Scholköpf, B.: From variational to deterministic autoencoders. In: ICLR (2020)
8. Goodfellow, I., et al.: Generative adversarial nets. In: NeurIPS (2014)
9. Gretton, A., Borgwardt, K.M., Rasch, M.J., Scholkopf, B., Smola, A.: A kernel two-sample test. JMLR **13**, 723–773 (2012)
10. Harvey, W., Naderiparizi, S., Wood, F.: Conditional image generation by conditioning variational auto-encoders. In: ICLR (2022)
11. He, J., Spokoyny, D., Neubig, G., Berg-Kirkpatrick, T.: Lagging inference networks and posterior collapse in variational autoencoders. In: ICLR (2019)
12. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: NeurIPS (2017)

13. Higgins, I., et al.: β-VAE: learning basic visual concepts with a constrained variational framework. In: ICLR (2017)
14. Hoffman, M.D., Johnson, M.J.: ELBO surgery: yet another way to carve up the variational evidence lower bound. In: NeurIPS Workshop: Advances in Approximate Bayesian Inference (2016)
15. Karras, T., et al.: Alias-free generative adversarial networks. In: NeurIPS (2021)
16. Kim, H., Mnih, A.: Disentangling by factorising. In: ICML (2018)
17. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. ICLR (2014)
18. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009). https://www.cs.toronto.edu/~kriz/cifar.html
19. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist
20. Liu, S., Bousquet, O., Chaudhuri, K.: Approximation and convergence properties of generative adversarial learning. In: NeurIPS (2017)
21. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: ICCV (2015)
22. Lucasz, J., Tuckery, G., Grossez, R., Norouziy, M.: Don't blame the ELBO! a linear VAE perspective on posterior collapse. In: NeurIPS (2019)
23. Lucasz, J., Tuckery, G., Grossez, R., Norouziy, M.: Understanding posterior collapse in generative latent variable models. In: ICLR (2019)
24. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial autoencoders. In: ICLR (2016)
25. Mescheder, L., Geiger, A., Nowozin, S.: Which training methods for GANs do actually converge? In: ICML (2018)
26. Oord, A.v.d., Vinyals, O., Kavukcuoglu, K.: Neural discrete representation learning. In: NeurIPS (2017)
27. Peng, J., Liu, D., Xu, S., Li, H.: Generating diverse structure for image inpaintingwith hierarchical VQ-VAE. In: CVPR (2021)
28. Bowman, S.R., Vilnis, L.: Generating sentences from a continuous space. In: SIGNLL Conference on Computational Natural Language Learning (CoNLL) (2016)
29. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. In: ICLR (2016)
30. Razavi, A., Oord, A.v.d., Poole, B., Vinyals, O.: Preventing posterior collapse with δ-VAEs. In: ICLR (2019)
31. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: ICML, pp. 1278–1286 (2014)
32. Rosca, M., Lakshminarayanan, B., Mohamed, S.: Distribution matching in variational inference. arxiv preprint arxiv: abs/1802.06847 (2018)
33. Roth, K., Lucchi, A., Nowozin, S., Hofmann, T.: Stabilizing training of generative adversarial networks through regularization. In: NeurIPS (2017)
34. Rybkin, O., Daniilidis, K., Levine, S.: Simple and effective VAE training with calibrated decoders. In: ICML (2021)
35. Saha, S., Elhabian, S., Whitaker, R.: Gens: generative encoding networks. Mach. Learn. **111**, 4003–4038 (2022)
36. Sajjadi, M.S.M., Bachem, O., Lučić, M., Bousquet, O., Gelly, S.: Assessing generative models via precision and recall. In: Advances in Neural Information Processing Systems (NeurIPS) (2018)
37. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training GANs. In: NeurIPS (2016)

38. Shao, H., et al.: ControlVAE: controllable variational autoencoder. In: ICML (2020)
39. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)
40. Chen, R.T.Q., Li, X., Grosse, R., Duvenaud, D.: Isolating sources of disentanglement in VAEs. In: NeurIPS (2019)
41. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelköpf, B.: Wasserstein auto-encoders. In: ICLR (2018)
42. Tomczak, J.M., Welling, M.: VAE with a VampPrior. In: AISTATS (2018)
43. Zhao, S., Song, J., Ermon, S.: InfoVAE: balancing learning and inference in variational autoencoders. In: AAAI (2019)

Efficient Nonlinear DAG Learning Under Projection Framework

Naiyu Yin[1(✉)], Yue Yu[2], Tian Gao[3], and Qiang Ji[1]

[1] Rensselaer Polytechnic Institute, Troy, NY 12180, USA
{yinn2,jiq}@rpi.edu
[2] Lehigh University, Bethlehem, PA 18015, USA
yuy214@lehigh.edu
[3] IBM Research, Yorktown Heights, NJ 10598, USA
tgao@us.ibm.com

Abstract. Directed Acyclic Graphs (DAGs) are foundational in machine learning, causal inference, and probabilistic modeling. Recovering the underlying DAG structure from observational data is crucial in these areas. The DAG learning can be approached as a constrained optimization problem with a continuous acyclicity constraint, often solved iteratively through sub-problem optimization. A recent breakthrough has shown that the set of DAGs can be represented as the weighted gradients of graph potential functions. Hence, one may search for a DAG in the equivalent space, whereby the acyclicity constraint is guaranteed to be satisfied. However, the original work, DAG-NoCurl, is limited to (generalized) linear structural equation models (SEMs) where explicit weighted adjacency matrices are defined. Herein, we theoretically derive a nonlinear projection formulation and propose an efficient two-step nonlinear DAG learning method, which we coined DAG-NCMLP. The proposed approach first obtains a non-acyclic graph and then projects it to the equivalent space of DAGs to obtain the acyclic graph. Experimental studies on benchmark datasets demonstrate that our proposed method provides similar accuracy, if not better, compared to state-of-the-art non-parametric DAG learning methods with hard-constrained optimization, while substantially reducing the computational time.

Keywords: Causal Discovery · Structure Learning · Directed Acyclic Graphs

1 Introduction

Directed Acyclic Graphs (DAGs) are foundational in numerous fields, including machine learning [19,28], causal inference [20], and probabilistic modeling. Their acyclic nature provides a clear directionality, making them ideal for representing causal relationships among variables within a system. Learning the DAG

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78172-8_29.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 445–460, 2025.
https://doi.org/10.1007/978-3-031-78172-8_29

structure from observational data is crucial for uncovering causal mechanisms, making predictions, and understanding complex systems. However, the DAG learning problem is NP-hard, and the DAG space grows super-exponentially with the number of variables [1]. Zheng et al. (2018) [37] proposes a continuous DAG constraint, transforming the combinatorial optimization problem of DAG learning into a constrained continuous optimization problem. This formulation opens the door to employing various continuous optimization techniques from deep learning [12,13,15,16,32].

While achieving state-of-the-art accuracy on synthetic and real data, methods developed using the continuous optimization framework with the continuous DAG constraint face challenges in scaling to large datasets with thousands of variables due to their time-consuming nature. One of the primary reasons for this inefficiency is the use of the augmented Lagrangian method to enforce the continuous DAG constraint, as proposed by Zheng et al. (2018) [37]. This procedure transforms the constrained optimization problem into a sequence of soft-constrained optimization sub-problems, which are solved iteratively. To address this efficiency issue, Yu et al. (2021) [34] propose a novel approach that learns the DAG without any explicit acyclicity constraint. Their method projects the DAGs into an equivalent set and optimizes the solution for the DAG parameters within this admissible set. Consequently, the DAG learning problem can be formulated as a continuous optimization problem without an explicit acyclicity constraint, avoiding the need to directly solve the constrained optimization problem using the time-consuming augmented Lagrangian method.

While Yu et al. (2021) [34] demonstrates significant efficiency improvements, it is built upon the linear Structure Equation Model (SEM), where parameters are represented as a weighted adjacency matrix. This formulation cannot be directly applied to nonlinear SEMs with the non-parametric formulation, which uses a gradient-based adjacency matrix representation. Consequently, its performance in terms of accuracy may suffer when applied to complex nonlinear SEMs. To address this limitation, we propose applying the concept of DAG projection to nonlinear SEMs. Specifically, we theoretically establish that an equivalent set of gradient-based adjacency matrices exists and introduce a novel two-step approach to optimizing the solution within this equivalent set search space. Empirical studies demonstrate that our proposed approach achieves a significant efficiency gain over other state-of-the-art nonparametric DAG learning models.

Main Contributions. This paper presents three contributions. 1) We theoretically derive a non-parametric projection formulation for gradient-based adjacency matrices, thereby extending the projection framework's applicability beyond weighted adjacency matrix representation. 2) Building on this non-parametric projection formulation, we introduce a two-step DAG learning approach, referred to as DAG-NCMLP. 3) We empirically demonstrate the effectiveness of our proposed project-based nonparametric DAG learning algorithm on benchmark synthetic and real datasets. Our method significantly enhances computational efficiency while maintaining comparable accuracy to state-of-the-art DAG learning methods.

2 Related Work

The gold standard for establishing causality between variables in an intelligent system is intervention through controlled experiments. However, conducting such experiments is often impractical due to cost or feasibility constraints. As a result, recent studies have focused on recovering causal relationships solely from observational data. Causal discovery involves identifying causal relationships among a set of random variables in the form of DAGs using observational data.

The traditional causal discovery algorithms can be broadly categorized into two groups: constraint-based methods and score-based methods. Constraint-based methods estimate the DAG by conducting independent tests between variables. Popular algorithms in this category include PC [27], FCI [28,36], and IC [21]. On the other hand, score-based methods involve pre-defining a score function and searching the DAG space for a DAG with the optimal score. The differences among score-based methods lie in their search procedures, which can include hill-climbing [9,30], forward-backward search [1], dynamic programming [26], A^* [35], and integer programming [2,11]. Other widely used DAG learning methods include topological order-based search [4,6,25,29] and sampling [3,5,7,8,14,18,31].

Structure equation model-based methods encode statistical and causal dependencies through SEMs. Zheng et al. (2018) [37] introduced a continuous DAG constraint and the NOTEARS algorithm, which reformulates the original combinatorial DAG learning problem as a constrained continuous optimization. This conversion enables the use of continuous optimization techniques, as demonstrated in subsequent works such as [12,15], and [32]. Since then, several studies have extended the continuous DAG-constrained optimization formulation from linear models to nonlinear and nonparametric models [6,13,32,38]. To address the efficiency issues in these methods arising from the time-consuming augmented Lagrangian method used to enforce acyclicity, Ng et al. (2020) [16] and Yu et al. (2021) [34] have investigated learning frameworks that do not require an iterative process. Ng et al. (2020) [16] proposes training the framework with a soft acyclicity constraint, while Yu et al. (2021) [34] suggests projecting the DAG into an equivalent set that guarantees acyclicity. However, both works focus on the linear SEM setting. To the authors' best knowledge, this paper is the first attempt at developing an efficient continuous optimization approach without the iterative process for the nonlinear SEM setting.

3 DAG Projection Under Nonparameteric SEM

In this section, we provide the theoretical results of the DAG projection framework under the nonlinear SEM. These theoretical results will serve as the fundamental for developing the proposed algorithm in Sect. 4. With basic and necessary concepts introduced in Sect. 3.1, our theoretical contribution will be entailed in Sect. 3.2.

3.1 Preliminary

Nonlinear SEM. Let X denotes a set of d numbers of random variables, $X = (X_1, \cdots, X_d) \in \mathbb{R}^d$. The causal relations between a variable $X_j \in X$ and its parents can be modeled via SEM:

$$X_j = f_j(X_{\pi_j}) + E_j, j = 1, 2, \cdots, d \tag{1}$$

where $f_j(\cdot)$ is the nonlinear structural causal function. X_{π_j} are the parent variables of X_j. E_j is the exogenous noise variable corresponding to variable X_j. Together they account for the effects from all the unobserved latent variables and are assumed to be mutually independent [22].

DAG Learning under Nonlinear SEM. To learn a DAG $\mathcal{G}$ from a given joint distribution, X is modeled via SEMs defined by a set of continuous parameters $\boldsymbol{A} = (A_1, A_2, \cdots, A_d)$ that encode all the causal relations, as outlined in Eq. (2),

$$X_j = f_j(X; A_j) + E_j, j = 1, 2, \cdots, d \tag{2}$$

where A_j are the parameters in the nonlinear SEM for selecting parent variables X_{π_j} for variable X_j. Similar to prior works [13,38], we employ neural networks, in particular MLPs, to parameterize the nonlinear causal functions $f = (f_1, f_2, \cdots, f_d)$. For f_j, we have

$$f_j(X; A_j) = A_j^{(H)} \sigma \left(\cdots \sigma \left(A_j^{(2)} \sigma (A_j^{(1)} X) \right) \cdots \right) \tag{3}$$

where A_j^h represents the parameters for h^{th} layer in the MLP for X_j. We denote $A_j := (A_j^{(1)}, A_j^{(2)}, \cdots, A_j^{(h)}, \cdots, A_j^{(H)})$. Since $\boldsymbol{A}$ in the nonlinear SEM is not a weighted adjacency matrix with d by d dimensions, the DAG learning formulation that satisfies Eq. (2) and Eq. (3) is also known as **nonparametric SEM** according to [38]. We denote the $\boldsymbol{A}$ in the nonlinear SEM as the **gradient-based adjacency matrices**. We encode the causal dependencies in the first layers of MLPs, i.e., $A_1^{(1)}, A_2^{(1)}, \cdots, A_d^{(1)}$. We can obtain a weighted adjacency matrix $W(\boldsymbol{A}) \in \mathbb{R}^{d \times d}$ using the first layer weights, i.e., $W(\boldsymbol{A})[k, j] = \sqrt{\sum_b (A_j^{(1)}[b, k])^2}$. If there exists a causal link from variable X_k to X_j, then $W(\boldsymbol{A})[k, j] > 0$. Otherwise, we have $W(\boldsymbol{A})[k, j] = 0$ and equivalently $A_j^{(1)}[b, k] = 0$ for all b.

Given n observations of X, denoted as input data matrix $\boldsymbol{X}^{d \times n}$, the DAG learning problem can be formulated as follow

$$\begin{aligned} \boldsymbol{A}^* = \arg\min_{\boldsymbol{A}} & \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} \mathcal{L}\Big(X_j(i), f_j(\boldsymbol{X}(i); A_j)\Big) \\ & \text{subject to } h(W(\boldsymbol{A})) = 0 \end{aligned} \tag{4}$$

where $\boldsymbol{X}(i) \in \mathbb{R}^d$ is the i^{th} observation of variables X. $X_j(i)$ is the i^{th} observation of variable X_j. $h(W(\boldsymbol{A})) = tr(e^{W(\boldsymbol{A}) \circ W(\boldsymbol{A})}) - d = 0$ is the continuous acyclicity constraint following [38]. $\mathcal{L}(\cdot)$ is the least squared loss.

The SEM we employ in Eq. (2) has following assumptions: firstly, $f = (f_1, f_2, \cdots, f_d)$ represents a set of nonlinear causal functions; and secondly, $E_1, E_2, \cdots, E_d$ are independent noise variables. According to [23], given a distribution over random variables $p(X)$, a unique causal graph $\mathcal{G}$ can be identified.

We then briefly introduce notation for graph calculus in the following section.

Graph Calculus: Let $\widehat{\mathcal{G}} = (V, E)$ be a **complete undirected** graph where $V := \{1, \cdots, d\}$ is the set of vertices and E is the set of undirected edges. On each vertice, there is a real-valued function $f : V \to \mathbb{R}$, which is also known as the potential function. We denote the space of all potential functions as $L^2(V)$. We also define real-valued functions on edges $E = \{(i, j), i, j \in V\}$ with the requirement that these functions are alternating, i.e., $E[i, j] = -E[j, i]$. We denote the space of all alternating edge functions as $L^2_\wedge(E)$. Here we note that $p \in L^2(V)$ corresponds to a real vector $p = [p(1), \cdots, p(d)] \in \mathbb{R}^d$, and any $Y \in L^2_\wedge(E)$ corresponds to a skew-symmetric real matrix $Y \in \mathbb{R}^{d \times d}$ with $[Y]_{ij} = Y[i, j]$ and $Y = -Y^T$. We will use the same letter to denote a vector/matrix and the corresponding function on vertices/edges. We introduce graph calculus operators gradient, divergence, and the graph laplacian in **Definition** 1.

Definition 1. *The gradient (*$\text{grad} : L^2(V) \to L^2_\wedge(E)$*) is an operator defined on any function* p *on vertices:*

$$(\text{grad } p)[i, j] = p(j) - p(i), \quad \forall (i, j) \in E$$

The divergence ($\text{div} : L^2_\wedge(E) \to L^2(V)$*) is defined on any alternating function* Y *on edges:*

$$(\text{div } Y)(i) = \sum_{j=1}^{d} Y[i, j], \quad \forall i \in V.$$

The graph Laplacian ($\triangle_0 : L^2(V) \to L^2(V)$*) is an operator on any function* p *on vertices:*

$$(\triangle_0 p)(i) = -(\text{div grad } p)(i) = dp(i) - \sum_{j=1}^{d} p(j), \quad \forall i \in V.$$

Given a function $Y \in L^2_\wedge(E)$, with ReLU denoting the rectified linear unit function, we can find a weighted adjacency matrix $\text{ReLU}(Y) \in \mathbb{R}^{d \times d}$ as:

$$\text{ReLU}(Y)[i, j] = \begin{cases} Y[i, j], & \text{if } Y[i, j] > 0, \\ 0, & \text{else,} \end{cases}$$

We define a weighted directed graph $\mathcal{G}_{\text{ReLU}(Y)}$ from $\text{ReLU}(Y)$ in **Definition** 2:

Definition 2. *Consider a complete undirected graph* $\widehat{\mathcal{G}}(V, E)$ *and* $Y \in L^2_\wedge(E)$*, a directed graph* $\mathcal{G}_{\text{ReLU}(Y)}(V, E_{\text{ReLU}(Y)})$ *is defined such that there is a directed edge from vertex* i *to vertex* j *in* $\mathcal{G}_{\text{ReLU}(Y)}$ *if and only if* $Y[i, j] > 0$*, i.e., the set of directed edges* $E_{\text{ReLU}(Y)} = \{(i, j) | Y[i, j] > 0\}$*.* $\text{ReLU}(Y)$ *is a weighted adjacency matrix of* $\mathcal{G}_{\text{ReLU}(Y)}$*.*

Building on **Definition** 1 and **Definition** 2, [34] offers an equivalent representation of a DAG under linear SEM, whereby a DAG $\mathcal{G}$ with d nodes is characterized by a weighted adjacency matrix $W \in \mathbb{R}^{d\times d}$. This formulation is supported by **Theorem** 1 as presented in [33,34].

Theorem 1. *[33] For any weight matrix $S \in \mathbb{R}^{d\times d}$ and potential function $p \in L^2(V)$, $S \circ \mathrm{ReLU}(\mathrm{grad}(p))$ is the weighted adjacency matrix of a DAG. On the other hand, let $W \in \mathbb{R}^{d\times d}$ be the weighted adjacency matrix of any DAG with d nodes, then there exists a weight matrix $S \in \mathbb{R}^{d\times d}$ and a function $p \in L^2(V)$ such that $W = S \circ \mathrm{ReLU}(\mathrm{grad}(p))$. Hence, $\{\mathcal{G}_{S\circ \mathrm{ReLU}(\mathrm{grad}(p))}\}$ is equivalent to the DAG space.*

3.2 An Equivalent Model for DAG

Theorem 1 can only be applied to linear SEMs because it requires the usage of the square-weighted adjacency matrices. **Our key theoretical contribution is to derive the equivalent theorem in Theorem** 2 **for nonlinear (nonparametric) SEMs to remove this limitation and handle the gradient-based adjacency matrix representation in** Eq. (3).

Theorem 2. *The acyclicity holds for the neural network formulation in Eq.* (3) *if and only if there exists a function $p \in L^2(V)$ and weight matrices $S_j \in \mathbb{R}^{m_1\times d}$, $j = 1, \cdots, d$, such that*

$$A_j^{(1)}[b,k] = S_j[b,k]\,\mathrm{ReLU}(\mathrm{grad}(p))[k,j]. \tag{5}$$

Here m_1 is the number of hidden units in the first layer of MLP.

Proof. As shown in [38], $W(\boldsymbol{A})[k,j] = \sqrt{\sum_b (A_j^{(1)}[b,k])^2}$ encodes the dependency structure amongst the X_j and the neural network formulation in Eq. (3) satisfies the acyclicity constraint if and only if $W(\boldsymbol{A})$ is acyclic. Assuming $A_j^{(1)}$ satisfies Eq. (5) for all j, we note that

$$W(\boldsymbol{A})[k,j] = \sqrt{\sum_b (S_j[b,k])^2}\,\mathrm{ReLU}(p(j) - p(k)) = \tilde{S} \circ \mathrm{ReLU}(\mathrm{grad}(p)),$$

where $\tilde{S}[k,j] = \sqrt{\sum_b (S_j[b,k])^2}$ and $\tilde{S} \in \mathbb{R}^{d\times d}$. **Theorem** 1 then immediately indicates that $W(\boldsymbol{A})$ is acyclic. On the other hand, if $W(\boldsymbol{A})$ satisfies the acyclicity constraint, **Theorem 1** guarantees that one can find $\tilde{S} \in \mathbb{R}^{d\times d}$ and $p \in L^2(V)$ satisfying

$$W(\boldsymbol{A})[k,j] = \tilde{S}[k,j]\,\mathrm{ReLU}(p(j) - p(k)).$$

Notice that when $\mathrm{ReLU}(p(j) - p(k)) = 0$, we have $W(\boldsymbol{A})[k,j] = \sqrt{\sum_b (A_j^{(1)}[b,k])^2} = 0$ and hence $A_j^{(1)}[b,k] = 0$ for all $b \in \{1, \cdots, m_1\}$. Therefore Eq. (5) can be satisfied by setting

$$S_j[b,k] = \begin{cases} 0, & \text{if } p(j) \le p(k) \\ \frac{A_j^{(1)}[b,k]}{p(j)-p(k)}, & \text{if } p(j) > p(k). \end{cases} \tag{6}$$

The dependency structure $W(\boldsymbol{A})$ obtained by performing projection is a non-maximum acyclic graph that minimizes $\|W(\boldsymbol{A}) - \tilde{S}\|_2$.

Let $C(M)$ denote the connectivity matrix [17] of a directed graph M such that $[C(M)]_{ij} = 1$ only if a directed path exists from vertex i to vertex j. **Theorem** 3 provides an efficient approach to calculate p and S_j from $\boldsymbol{A}$:

Theorem 3. *Let $\boldsymbol{A}$ be a set of parameters in Eq. (3) which satisfying the acyclicity constraint, then*

$$p = -\triangle_0^{\dagger} \operatorname{div}\left(\frac{1}{2}(C(W(\boldsymbol{A})) - C(W(\boldsymbol{A}))^T)\right), \tag{7}$$

preserves the topological order in $W(\boldsymbol{A})$ such that $p(j) > p(i)$ if there is a directed path from vertex i to j. Here $\dagger$ denotes the Moore-Penrose pseudo-inverse. Moreover, with S_j defined in Eq. (6) we have

$$A_j^{(1)}[b,k] = S_j[b,k]\,\mathrm{ReLU}(\mathrm{grad}(p))[k,j]$$

We refer interested readers to Appendix A for a detailed proof.

Theorem 2 and **Theorem** 3 allow us to find the equivalent search space for the gradient-based adjacency matrix representations in Eq. (3). We introduce a two-step DAG learning algorithm that optimizes parameters within the equivalent search space, thereby circumventing the need for enforcing the computationally intensive acyclicity constraint.

4 Proposed Algorithm: DAG-NCMLP

Guided by theoretical insights from **Theorem** 2 and **Theorem** 3, we propose a nonparametric project-based DAG learning algorithm, named DAG-NCMLP, which employs MLP as the gradient-based weighted adjacency matrix representation. To avoid the strict enforcement of the DAG constraint, we propose learning the neural network parameters, $\boldsymbol{S}$ and the potential function p, instead of directly optimizing a gradient-based weighted adjacency matrix representation $\boldsymbol{A}$ that must satisfy the DAG constraint. Given the increasing complexity of optimizing both $\boldsymbol{S}$ and p, we employ a two-step procedure. In Step 1, we derive an initial solution $\hat{\boldsymbol{A}}$ without strictly adhering to the DAG constraint. This step aims to obtain a good initial solution from which a stable, informative estimate of the potential function, p^{pre}, can be extracted. In Step 2, we focus on optimizing $\boldsymbol{S}$ and p, guided by p^{pre}, to ultimately learn the optimal DAG. The algorithm is outlined in Algorithm 1. We will detail each step as follows.

Step 1. This step aims to yield an estimation of $\boldsymbol{A}$ that produces a stable and preferably informative potential function p. To obtain such an initial estimate, we propose to solve a penalized formulation of the original constrained optimization problem as shown in Eq. (10), by employing the standard augmented Lagrangian method and gradually increase the penalization parameter ρ. Instead of continuing the iterative procedure till convergence as in the original augmented

Lagrangian, here we only solve the sub-optimizations for a few iterations. As a result, the solution is not guaranteed to fully satisfy the acyclicity constraint. To be more specific, we denote the objective function in Eq. (10) as $L_\rho(\boldsymbol{A}, \alpha)$. Initially, we update the penalization parameter ρ by gradually increasing its value while holding $(\alpha, \boldsymbol{A})$ constant. Then, we update α using Eq. (8) for $K = 5$ iterations. For each pair of given ρ and α, we solve the sub-optimization problem in Eq. (9) for $T = 10d$ iterations[1]. Further details of the choices of K and T can be found in Sect. 5.

$$\alpha^{k+1} = \alpha^k + \rho_{k+1} h(W(\boldsymbol{A}^k)). \tag{8}$$

$$\boldsymbol{A}^{k+1} = \arg\min_{\boldsymbol{A}} L_{\rho_{k+1}}(\boldsymbol{A}^k, \alpha_{k+1}) \tag{9}$$

DAG-NCMLP utilizes a distinct Step 1 procedure compared to [34]. In [34], the optimization is solved with fixed values of α and ρ, akin to the augmented Lagrangian method with only one step of optimization. This approach is sufficient to yield a stable potential vector p under linear SEM with simple linear relationships between variables. However, for nonlinear models, solutions obtained with fixed coefficients are often inadmissible for estimating the potential function.

Step 2. This step aims to optimize the parameters $\boldsymbol{S}$ and p within the equivalent DAG space, using the potential function p^{pre} derived from the initial solution $\hat{\boldsymbol{A}}$. After Step 1, the resulting $W(\hat{\boldsymbol{A}})$ is typically non-acyclic since the DAG constraint is not satisfied. To obtain a DAG solution, we first approximate the potential function p^{pre} using Eq. (7) from **Theorem** 3. Next, we derive an initial graph solution $W(\boldsymbol{A}^{pre})$ in Eq. (12) by optimizing over $\boldsymbol{A}$ with p^{pre} fixed. Finally, we obtain the optimal DAG solution $W(\boldsymbol{A}^*)$ in Eq. (13) by jointly optimizing over $\boldsymbol{A}$ and p. Both in Step 1 and Step 2, we apply the standard thresholding procedure [37] to $W(\hat{\boldsymbol{A}})$ and $W(\boldsymbol{A}^*)$, respectively. The outcome of Step 1 directly impacts Step 2. A more accurate estimation of W^{pre} in Step 1 results in a better approximation of the potential function p^{pre}. This, in turn, encodes more accurate partial ordering information, aiding the algorithm in converging to an accurate estimation of $\boldsymbol{A}^*$ in Step 2c. Here we note that whether W^{pre} satisfies the acyclic constraint does not affect the algorithm's ability to obtain an effective p^{pre}, since p^{pre} can preserve the partial ordering information of a non-acyclic W^{pre}. Our proposed method involves optimizing over both $\boldsymbol{A}$ and p, with each affecting the estimation of the other during the optimization process. Step 2b simplifies the optimization process by fixing p to p^{pre}, allowing $\boldsymbol{A}$ to achieve a good initial estimation. The accuracy of $\boldsymbol{A}^*$ obtained by DAG-NCMLP is greatly compromised if Step 2b is omitted. We also point out that the objective functions in Eqs. (12) and (13) are non-convex. Consequently, only stationary solutions can be guaranteed, a characteristic shared with all continuous optimization-based DAG algorithms.

[1] "Solving the sub-optimization problem for $T = 10d$ iterations" means the optimizer stops when it performs $T = 10d$ gradient descent steps.

Algorithm 1. DAG-NCMLP Algorithm

Step 1: Within fixed numbers of iterations, solve for initialization $\hat{\boldsymbol{A}}$.

$$\begin{aligned}\hat{\boldsymbol{A}} = \quad & \underset{\boldsymbol{A}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\mathcal{L}\Big(X_j(i), f_j(\boldsymbol{X}(i), A_j)\Big) + \lambda||A_j^{(1)}||_{1,1} \\ & + \alpha h(W(\boldsymbol{A})) + \frac{\rho}{2}\|h(W(\boldsymbol{A})\|^2.\end{aligned} \tag{10}$$

Threshold $W(\hat{\boldsymbol{A}})$ to obtain W^{pre}.

Step 2: Obtain an acyclic graph solution $W(\boldsymbol{A}^*)$

2a) Obtain initial guess of potential vector p^{pre}:

$$p^{pre} = -\triangle_0^{\dagger}\,\mathrm{div}\left(\frac{1}{2}(C(W^{pre}) - C(W^{pre})^T)\right), \tag{11}$$

which preserves the variable ordering of W^{pre}.

2b) Solve for the initial guess of DAG $W(\boldsymbol{A}^{pre})$ with fixed potential vector p^{pre} and initialization $\hat{\boldsymbol{A}}$:

$$\boldsymbol{A}^{pre} = \underset{\{\boldsymbol{A},S\}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\mathcal{L}\Big(X_j(i), f_j(\boldsymbol{X}(i), A_j)\Big) + \lambda||A_j^{(1)}||_{1,1} \tag{12}$$

where $A_j^{(1)}[b,k] = S_j[b,k]\mathrm{ReLU}(p^{pre}(j) - p^{pre}(k))$.

2c) Solve for $W(\boldsymbol{A}^*), p^*$ with initialization $\boldsymbol{A}^{pre}$:

$$\boldsymbol{A}^*, p^* = \underset{\{\boldsymbol{A},S,p\}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\mathcal{L}\Big(X_j(i), f_j(\boldsymbol{X}(i), A_j)\Big) + \lambda||A_j^{(1)}||_{1,1} \tag{13}$$

where $A_j^{(1)}[b,k] = S_j[b,k]\mathrm{ReLU}(p^*(j) - p^*(k))$. Threshold $W(\boldsymbol{A}^*)$ to obtain W^{est} as output.

5 Experiments

We perform empirical evaluations on both synthetic and real data to demonstrate the effectiveness of our proposed DAG-NCMLP algorithm in improving efficiency while maintaining comparable accuracy.

Synthetic Datasets. We evaluate DAG-NCMLP on synthetic nonlinear datasets, generated using the same method as in prior work [38]. The ground truth DAGs are generated from Erdo-Renyi (ER) and Scale-Free (SF) graph models, with an expected edge degree set to 2 and 4. The synthetic data are generated from three-layer MLPs, which are universal nonlinear estimators, following the approach in [38]. To demonstrate the robustness of our proposed method across different data models, we also generate data using the Gaussian Process (GP) SEM. We create 10 graphs for each graph setting (ER2-MLP, ER4-MLP, SF2-MLP, SF4-MLP, ER2-GP, ER4-GP, SF2-GP, and SF4-GP), and test with varying numbers

of variables $d = 10, 20, 40, 50, 100$. For each setting, we simulate 10 trials with $n = 1000$ i.i.d. data observations.

Real Dataset. We further assess the performance of DAG-NCMLP using real-world flow cytometry data from Sachs et al. (2005) [24] for modeling protein signaling pathways. The dataset comprises continuous measurements of 11 phosphoproteins in individual T-cells. We specifically selected 853 observations corresponding to the first experimental condition outlined in Sachs et al. (2005) [24] as our dataset $\mathcal{D}$. For our reference graph (ground truth), we utilize the provided DAG, which consists of 11 nodes and 17 edges. It is important to note that this consensus graph may not provide a comprehensive or entirely accurate representation of the system under study.

Evaluation Metrics. We employ the Structural Hamming Distance (SHD) and runtime to evaluate the accuracy and efficiency of the estimated DAGs respectively. We report the average SHD with its standard deviation across 10 trials, and the average time (in seconds) with its standard deviation in Tables 1, 2, and 3. The SHD metric we use doesn't consider Markov Equivalence since the non-linear SEM in our formulation is fully identifiable.

Baselines. We mainly compare our method with following SEM-based baselines: GraN-DAG [13], DAG-GNN [32], GS-GES [10] and NOTEARS-MLP [38]. We use the default parameters for these baselines. For the baseline NOTEARS-MLP, we use the hyper-parameters that are reported in Zheng et al. (2020) [38]. The experiments for all the baselines and the proposed method, DAG-NCMLP are computed on a computing node with twenty 3.1 GHz CPU cores[2].To provide a more comprehensive comparison, we also compared causal discovery methods from different categories, including MMHC [30] and DAG-NoCurl [34] and show the empirical results in Appendix B.

The Choice of K and T. The hyperparameters K and T control the accuracy of the potential function p^{pre}, and consequently, the accuracy of the final output DAG. Ideally, we want to select relatively small values for K and T to enhance the algorithm's efficiency by reducing the number of optimization steps. However, K and T should also be large enough to allow p^{pre} to capture as much information as possible. A reasonable approach to selecting the hyper-parameters K, T is through empirical evaluation. The K, T are empirically selected on ER2 datasets when values of p do not change substantially (note that we do not use accuracy or SHD as the selection criterion). We observe that the algorithm performance is not sensitive to the values of K, T, Hence we fix the values of $K = 5$ and $T = 10d$.

[2] Due to the complexity of the neural networks used in methods like DAG-GNN and GraN-DAG, these models are typically run on a GPU to reduce runtime. However, to ensure a fair comparison of efficiency, we run experiments for these two baselines on a CPU, as with the other baselines. GPU acceleration is a standard technique and not a unique contribution of these two baselines; it can be applied to all the algorithms, including our DAG-NCMLP.

Table 1. Comparison of Different Algorithms on Nonlinear **Multi-Layer Perceptron** Synthetic datasets: results (mean ± standard error over 10 trails) on SHD and Run time(in seconds), where bold number s highlight the best method for each case.

ER2: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	15.0 ± 6.0	13.3 ± 5.5	10.5 ± 3.9	5.7 ± 3.2	$\mathbf{5.5 \pm 2.5}$
20	22.7 ± 1.8	25.7 ± 3.6	19.4 ± 5.6	$\mathbf{13.0 \pm 3.8}$	13.5 ± 4.0
40	57.5 ± 8.6	56.1 ± 6.7	40.5 ± 9.5	$\mathbf{27.7 \pm 5.1}$	27.8 ± 5.8
50	68.9 ± 13.3	65.8 ± 7.8	50.6 ± 8.4	$\mathbf{36.0 \pm 9.7}$	36.9 ± 10.3
100	$> 60h$	144.8 ± 7.1	$> 60h$	$\mathbf{77.3 \pm 4.0}$	80.5 ± 6.0
SF2: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	9.0 ± 4.5	9.5 ± 3.4	8.5 ± 3.2	$\mathbf{1.9 \pm 1.2}$	2.2 ± 1.1
20	19.7 ± 2.1	22.9 ± 3.4	22.7 ± 4.4	8.0 ± 3.2	$\mathbf{7.8 \pm 2.9}$
40	48.6 ± 4.6	52.4 ± 3.1	51.3 ± 7.0	18.9 ± 6.5	$\mathbf{18.5 \pm 6.2}$
50	52.3 ± 11.9	58.6 ± 6.5	65.1 ± 5.6	$\mathbf{24.5 \pm 6.2}$	25.5 ± 5.5
100	$> 60h$	149.2 ± 7.6	149.5 ± 7.2	81.4 ± 9.9	$\mathbf{79.0 \pm 7.1}$
ER4: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	22.4 ± 4.8	27.1 ± 3.4	24.10 ± 7.1	$\mathbf{8.0 \pm 1.9}$	9.9 ± 2.4
20	71.2 ± 16.2	65.5 ± 8.1	50.2 ± 9.7	$\mathbf{29.1 \pm 4.7}$	32.7 ± 7.1
40	96.7 ± 18.4	130.4 ± 10.2	87.7 ± 12.8	$\mathbf{47.7 \pm 9.3}$	55.0 ± 25.9
50	121.0 ± 16.9	161.1 ± 10.8	115.70 ± 21.8	$\mathbf{68.7 \pm 14.0}$	70.9 ± 15.3
100	$> 60h$	332.2 ± 12.6	$> 60h$	$\mathbf{134.5 \pm 13.4}$	144.1 ± 38.0
SF4: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	14.0 ± 1.5	18.1 ± 3.3	18.3 ± 5.6	$\mathbf{3.5 \pm 2.3}$	4.8 ± 2.9
20	29.7 ± 2.1	48.2 ± 5.5	44.3 ± 3.9	$\mathbf{12.4 \pm 4.2}$	$\mathbf{12.4 \pm 4.1}$
40	78.6 ± 4.2	119.9 ± 6.8	111.0 ± 7.2	$\mathbf{47.2 \pm 5.5}$	48.7 ± 6.5
50	132.3 ± 11.0	158.6 ± 6.5	148.3 ± 7.6	$\mathbf{62.1 \pm 21.2}$	77.7 ± 17.4
100	$> 60h$	323.1 ± 9.6	$> 60h$	211.1 ± 11.8	$\mathbf{202.0 \pm 10.8}$
ER2: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$2.6e3 \pm 5.2e3$	$6.6e2 \pm 2.3e2$	$3.1e2 \pm 5.3e1$	$3.1e2 \pm 1.1e2$	$\mathbf{1.2e2 \pm 6.0e1}$
20	$2.5e3 \pm 5.8e2$	$1.5e3 \pm 2.3e2$	$1.5e3 \pm 1.2e2$	$7.6e2 \pm 1.4e2$	$\mathbf{3.9e2 \pm 8.6e1}$
40	$8.6e3 \pm 1.8e3$	$8.0e3 \pm 1.9e2$	$6.8e3 \pm 1.4e3$	$1.6e3 \pm 2.2e2$	$\mathbf{1.1e3 \pm 2.1e2}$
50	$1.4e4 \pm 1.5e3$	$1.2e4 \pm 1.1e2$	$1.0e4 \pm 9.4e2$	$8.5e3 \pm 2.3e3$	$\mathbf{1.6e3 \pm 3.9e2}$
100	$> 60h$	$2.0e4 \pm 6.4e2$	$> 60h$	limit to $60h$	$\mathbf{6.5e3 \pm 1.3e3}$
SF2: Run Time					
d	GraN-DA	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$1.2e3 \pm 1.3e2$	$1.1e3 \pm 1.7e2$	$2.6e2 \pm 2.7e1$	$2.4e2 \pm 8.2e1$	$\mathbf{7.5e1 \pm 3.5e1}$
20	$4.6e3 \pm 1.7e3$	$1.2e3 \pm 2.2e2$	$1.1e3 \pm 1.4e2$	$1.8e3 \pm 3.9e2$	$\mathbf{3.4e2 \pm 5.9e1}$
40	$1.5e4 \pm 1.3e4$	$3.2e3 \pm 2.8e2$	$4.4e3 \pm 3.9e2$	$2.8e3 \pm 6.6e3$	$\mathbf{1.1e3 \pm 1.3e2}$
50	$2.2e4 \pm 5.2e2$	$2.1e4 \pm 1.5e2$	$7.0e3 \pm 5.0e2$	$4.5e3 \pm 1.2e3$	$\mathbf{1.8e3 \pm 5.3e2}$
100	$> 60h$	$> 60h$	$> 60h$	limit to $60h$	$\mathbf{9.3e3 \pm 2.7e3}$
ER4: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$1.2e3 \pm 2.3e2$	$8.6e2 \pm 8.2e1$	$6.7e2 \pm 3.6e2$	$1.2e3 \pm 4.4e2$	$\mathbf{1.5e2 \pm 9.2e1}$
20	$7.1e3 \pm 8.2e2$	$9.6e2 \pm 7.5e1$	$5.5e3 \pm 9.9e3$	$2.7e3 \pm 6.8e2$	$\mathbf{6.4e2 \pm 2.7e2}$
40	$7.6e3 \pm 1.0e3$	$7.2e3 \pm 9.7e2$	$9.6e3 \pm 2.1e3$	$7.4e3 \pm 1.6e3$	$\mathbf{1.5e3 \pm 2.5e2}$
50	$1.9e4 \pm 7.8e2$	$2.1e4 \pm 2.1e2$	$1.9e4 \pm 2.1e2$	$1.0e4 \pm 2.1e3$	$\mathbf{2.3e3 \pm 4.6e2}$
100	$> 60h$	$> 60h$	$> 60h$	limit to $60h$	$\mathbf{7.4e3 \pm 2.2e3}$
SF4: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$1.2e3 \pm 1.8e2$	$8.4e2 \pm 1.1e2$	$3.2e2 \pm 3.2e1$	$8.4e2 \pm 3.7e2$	$\mathbf{1.2e2 \pm 4e1}$
20	$1.3e3 \pm 2.7e2$	$8.2e2 \pm 1.5e2$	$1.3e3 \pm 1.8e2$	$1.5e3 \pm 4.4e2$	$\mathbf{3.6e2 \pm 9.9e1}$
40	$9.8e3 \pm 1.5e2$	$7.8e3 \pm 1.9e2$	$5.4e3 \pm 5.8e2$	$8.4e3 \pm 3.7e2$	$\mathbf{1.3e3 \pm 3.2e2}$
50	$2.2e4 \pm 6.8e3$	$1.6e4 \pm 2.0e3$	$9.1e3 \pm 1.9e3$	$6.8e3 \pm 3.3e3$	$\mathbf{2.7e3 \pm 6.7e2}$
100	$> 60h$	$> 60h$	$> 60h$	limit to $60h$	$\mathbf{7.1e3 \pm 8.6e2}$

Table 2. Comparison of Different Algorithms on Nonlinear **Gaussian Process** Synthetic datasets: results (mean ± standard error over 10 trails) on SHD and Run time(in seconds), where bold number s highlight the best method for each case.

ER2: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	12.3 ± 1.5	17.3 ± 0.9	9.6 ± 2.4	$\mathbf{7.2 \pm 2.1}$	7.5 ± 2.2
20	34.3 ± 8.8	36.0 ± 1.7	$\mathbf{19.3 \pm 6.5}$	30.0 ± 0.7	30.5 ± 10.8
40	48.4 ± 4.4	73.2 ± 2.1	$\mathbf{33.7 \pm 10.0}$	43.2 ± 7.2	42.7 ± 7.9
50	71.2 ± 12.4	93.1 ± 3.1	$\mathbf{47.8 \pm 8.2}$	62.1 ± 9.6	62.2 ± 8.9
100	$> 60h$	185.6 ± 3.7	$\mathbf{94.5 \pm 7.3}$	125.7 ± 2.5	128.3 ± 3.3
SF2: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	16.4 ± 2.0	15.4 ± 0.9	11.4 ± 3.7	$\mathbf{7.5 \pm 1.9}$	7.7 ± 2.3
20	33.1 ± 5.4	34.8 ± 1.2	31.2 ± 3.9	$\mathbf{28.6 \pm 2.7}$	29.1 ± 4.4
40	62.3 ± 6.5	72.4 ± 1.6	64.8 ± 6.7	$\mathbf{58.6 \pm 5.0}$	58.7 ± 4.5
50	94.8 ± 10.8	91.7 ± 2.3	82.1 ± 8.2	$\mathbf{77.0 \pm 4.4}$	79.0 ± 5.0
100	$> 60h$	185.8 ± 1.4	$> 60h$	$\mathbf{171.2 \pm 2.1}$	173.0 ± 3.3
ER4: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	20.8 ± 3.8	38.3 ± 1.1	32.3 ± 2.6	$\mathbf{19.3 \pm 3.4}$	21.1 ± 2.7
20	68.1 ± 9.7	78.7 ± 1.0	60.5 ± 3.6	56.4 ± 4.2	$\mathbf{56.3 \pm 4.3}$
40	154.9 ± 7.0	140.8 ± 4.7	$\mathbf{119.0 \pm 7.2}$	138.1 ± 6.0	138.2 ± 9.0
50	186.6 ± 16.2	195.7 ± 1.7	$\mathbf{154.9 \pm 5.6}$	188.3 ± 12.7	189.5 ± 15.4
100	$> 60h$	391.0 ± 2.7	$\mathbf{309.0 \pm 9.9}$	350.7 ± 2.1	354.3 ± 4.2
SF4: SHD					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	27.1 ± 4.4	28.8 ± 1.0	24.3 ± 2.7	$\mathbf{15.1 \pm 3.1}$	16.1 ± 3.9
20	63.1 ± 1.9	68.2 ± 1.2	61.0 ± 4.0	60.7 ± 2.3	$\mathbf{60.5 \pm 2.4}$
40	139.9 ± 6.1	145.8 ± 1.7	133.4 ± 2.8	$\mathbf{129.4 \pm 4.0}$	131.1 ± 4.3
50	184.6 ± 4.2	185.1 ± 1.4	$\mathbf{169.8 \pm 6.5}$	171.8 ± 4.5	170.4 ± 4.4
100	$> 60h$	379.2 ± 3.2	$> 60h$	$\mathbf{351.7 \pm 4.1}$	356.0 ± 5.0
ER2: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$5.3e2 \pm 7.5e1$	$5.3e2 \pm 3.7e1$	$1.8e2 \pm 1.8e1$	$1.7e2 \pm 7.8e1$	$\mathbf{6.5e1 \pm 1.6e1}$
20	$7.6e2 \pm 7.7e1$	$5.5e2 \pm 4.7e1$	$8.0e2 \pm 9.0e1$	$1.1e3 \pm 2.7e2$	$\mathbf{7.8e1 \pm 1.9e1}$
40	$2.1e3 \pm 1.4e2$	$6.6e2 \pm 3.2e1$	$4.1e3 \pm 3.6e2$	$2.7e3 \pm 7.5e2$	$\mathbf{5.4e2 \pm 4.9e1}$
50	$2.4e3 \pm 2.1e2$	$7.6e2 \pm 4.6e1$	$5.5e3 \pm 5.2e2$	$4.1e3 \pm 8.8e2$	$\mathbf{7.0e2 \pm 4.4e1}$
100	$> 60h$	$3.6e3 \pm 4.1e2$	$2.1e4 \pm 1.1e3$	limit to $60h$	$\mathbf{1.2e3 \pm 3.1e1}$
SF2: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$4.2e2 \pm 3.4$	$4.3e2 \pm 4.0e1$	$1.3e2 \pm 1.4e1$	$1.6e2 \pm 4.7e1$	$\mathbf{2.1e1 \pm 4.5}$
20	$9.1e2 \pm 1.4e2$	$5.3e2 \pm 2.3e1$	$4.9e2 \pm 6.3e1$	$5.4e2 \pm 1.7e2$	$\mathbf{8.6e1 \pm 1.4e1}$
40	$2.0e3 \pm 1.7e2$	$6.8e2 \pm 5.5e1$	$2.3e3 \pm 2.5e2$	$1.5e3 \pm 4.2e2$	$\mathbf{5.5e2 \pm 6.2e1}$
50	$2.8e3 \pm 5.2e2$	$8.3e2 \pm 8.8e1$	$7.0e3 \pm 5.0e2$	$2.8e3 \pm 7.0e2$	$\mathbf{6.6e2 \pm 3.2e1}$
100	$> 60h$	$2.3e3 \pm 1.8e2$	$> 60h$	limit to $60h$	$\mathbf{1.2e3 \pm 3.0e1}$
ER4: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$3.4e2 \pm 4.4e1$	$5.0e2 \pm 2.1e1$	$2.5e2 \pm 4.0e1$	$1.5e2 \pm 2.5e1$	$\mathbf{6.6e1 \pm 1.1e1}$
20	$6.8e2 \pm 9.8e1$	$5.5e2 \pm 3.5e1$	$9.6e2 \pm 8.5e1$	$5.8e2 \pm 1.4e2$	$\mathbf{2.1e2 \pm 2.3e1}$
40	$1.8e3 \pm 1.2e2$	$6.9e2 \pm 9.7e1$	$4.0e3 \pm 1.7e2$	$2.5e3 \pm 5.5e2$	$\mathbf{6.3e2 \pm 1.4e2}$
50	$2.3e3 \pm 1.7e2$	$1.2e3 \pm 3.0e2$	$6.3e3 \pm 3.6e2$	$3.6e3 \pm 6.2e2$	$\mathbf{1.1e3 \pm 2.3e2}$
100	$> 60h$	$4.2e3 \pm 3.1e2$	$> 60h$ [a]	$6.5e3 \pm 2.3e2$	$\mathbf{3.3e3 \pm 2.0e2}$
SF4: Run Time					
d	GraN-DAG	DAG-GNN	GS-GES	NOTEARS-MLP	DAG-NCMLP
10	$3.4e2 \pm 3.5e1$	$5.0e2 \pm 3.4e1$	$1.3e2 \pm 1.2e1$	$2.3e2 \pm 4.8e1$	$\mathbf{4.9e1 \pm 9.5}$
20	$7.0e2 \pm 1.1e2$	$5.7e2 \pm 4.2e1$	$5.2e2 \pm 7.3e1$	$2.7e2 \pm 9.6e1$	$\mathbf{1.9e2 \pm 3.3e1}$
40	$1.8e3 \pm 1.7e2$	$6.9e2 \pm 3.3e1$	$2.9e3 \pm 2.4e2$	$1.4e3 \pm 2.7e2$	$\mathbf{6.5e2 \pm 4.3e1}$
50	$2.6e3 \pm 4.3e2$	$9.6e2 \pm 2.0e2$	$4.7e3 \pm 3.5e2$	$2.1e3 \pm 3.7e2$	$\mathbf{8.6e2 \pm 4.6e1}$
100	$> 60h$	$1.8e3 \pm 4.2e2$	$> 60h$	$3.9e3 \pm 1.4e2$	$\mathbf{1.3e3 \pm 5.7e2}$

[a] We allow for $60h$ on 10 graphs. If the average runtime is longer than $2.16e4$ seconds, then we will mark the runtime as $> 60h$ in the table. For example, the average runtime for GS-GES on ER4 graphs is $2.8e4 \pm 2.2e3$.

5.1 Empirical Results on Synthetic Data

In Tables 1 and 2, the top four sub-tables present the accuracy results in terms of the SHD. The bottom four sub-tables display the computational efficiency measured in CPU runtime in seconds. Given the complexity of the data, we imposed a 60-hour time limit for each method and then evaluated the intermediate or final learned DAGs. The data is generated under a non-linear SEM assumption, rendering linear SEM-based methods ineffective in capturing the complex non-linear relationships present in the data. Consequently, we compare our DAG-NCMLP only with baselines developed under non-linear SEMs.

Table 1 demonstrates that NOTEARS-MLP consistently outperforms other advanced methods across most settings, aligning with previous observations. Our proposed DAG-NCMLP method shows significant accuracy improvements compared to the baselines (GraN-DAG, GS-GES, and DAG-GNN) across all graph settings. While DAG-NCMLP's accuracy is comparable to NOTEARS-MLP, it surpasses NOTEARS-MLP in accuracy in 6 out of 20 graph settings and falls slightly behind within an acceptable range of differences in the remaining settings. In terms of efficiency, DAG-NCMLP requires significantly less computational time compared to the baselines, particularly NOTEARS-MLP. It typically completes computations in approximately half to 10% of the time required by NOTEARS-MLP.

Despite the universal nonlinear estimation capability of the 3-layer MLP model used to generate the synthetic data in Table 1, we aim to demonstrate the effectiveness of our proposed methods across different nonlinear SEM assumptions. Therefore, we present empirical evaluation results on Gaussian Process data in Table 2. Table 1 showcases DAG-NCMLP outperforming GraN-DAG and DAG-GNN, achieving results comparable to NOTEARS-MLP. However, in contrast to the results in Table 1, GS-GES outperforms NOTEARS-MLP in 8 out of 20 graph settings, achieving the highest accuracy. The differences between the accuracy of NOTEARS-MLP and DAG-NCMLP are minimal, with SHDs of DAG-NCMLP typically within a 2.6% variation of those of NOTEARS-MLP, except in extreme cases. In terms of efficiency, DAG-NCMLP is significantly more computationally efficient, requiring only 15.97% to 70.37% of the time required by NOTEARS-MLP, with greater gains for larger d. This observation in Table 2 aligns with the findings in Table 1, demonstrating that DAG-NCMLP substantially improves efficiency while maintaining comparable accuracy compared to NOTEARS-MLP. Additionally, DAG-NCMLP outperforms other state-of-the-art nonlinear SEM-based methods in terms of accuracy. Comparing the runtime of DAG-NCMLP in both tables, it is faster on GP data in Table 2 than on MLP data in Table 1. This difference is due to the simpler data generation process for GP data, which uses fewer parameters. As a result, DAG-NCMLP finds it easier to model the data distribution of GP data compared to MLP data.

Empirical results in Appendix B indicate that although some popular causal discovery methods have good efficiency, however, they suffer from poor accuracy issues. Our proposed DAG-NCMLP achieves good accuracy as the nonlinear SEM-based baselines while significantly improving the efficiency.

5.2 Empirical Results on Real Data

Table 3 presents the results of applying the DAG-NCMLP and 4 other baseline methods on the real dataset. The table reports the accuracy of the SHD, the number and the ratio of correctly estimated edges, and the computational efficiency in terms of the runtime in seconds. Table 3 shows that NOTEARS-MLP achieves an SHD of 15 in $4.4e2$ seconds, while DAG-NCMLP achieves an SHD of 15 in $1.5e2$ seconds. Two methods correctly estimate the same number of edges. Hence, on the real dataset, DAG-NCMLP can achieve a comparable accuracy with substantially reduced efficiency compared to NOTEARS-MLP. This is consistent with our observation on synthetic datasets.

Table 3. Comparison of different algorithms on Real Data: results on SHD, number of edges, and runtime.

Dataset	SHD	# Correct Edges	Ratio of Correct Edges	Runtime
GraN-DAG	**13**	6/17	0.353	$6.1e2$
DAG-GNN	19	**8/17**	**0.471**	$5.3e2$
GS-GES	17	6/17	0.353	$5.0e2$
NOTEARS-MLP	15	7/17	0.412	$4.4e2$
DAG-NCMLP	15	7/17	0.412	**1.5e2**

6 Conclusion

In this paper, we introduce an efficient DAG learning algorithm that utilizes a projection formulation on nonlinear SEMs, enabling better capture of complex nonlinear relationships between variables. We theoretically derive nonlinear projection formulations for gradient-based adjacency matrix representations. Leveraging these formulations, we propose a novel nonlinear DAG learning algorithm, DAG-NCMLP, designed to efficiently solve the unconstrained optimization problem inherent in the formulation and learn the DAG structure. Our empirical results demonstrate that DAG-NCMLP significantly enhances computational efficiency, particularly in scenarios with a large number of variables. Importantly, DAG-NCMLP achieves comparable accuracy to state-of-the-art nonparametric or nonlinear DAG learning methods. We believe that DAG-NCMLP presents a promising framework for DAG learning.

Acknowledgements. This work was supported in part by the National Science Foundation award IIS 2236026 and in part by IBM through the IBM-Rensselaer Future of Computing Research Collaboration. Y. Yu is grateful for support from the AFOSR grant FA9550-22-1-0197.

References

1. Chickering, D.M.: Optimal structure identification with greedy search. J. Mach. Learn. Res. **3**(Nov), 507–554 (2002)
2. Cussens, J.: Bayesian network learning with cutting planes. In: UAI (2011)
3. Eaton, D., Murphy, K.: Bayesian structure learning using dynamic programming and MCMC. arXiv preprint arXiv:1206.5247 (2012)
4. Friedman, N., Koller, D.: Being Bayesian about network structure. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 201–210. Morgan Kaufmann Publishers Inc. (2000)
5. Friedman, N., Koller, D.: Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks. Mach. learn. **50**(1-2), 95–125 (2003)
6. Gao, M., Ding, Y., Aragam, B.: A polynomial-time algorithm for learning nonparametric causal graphs. arXiv preprint arXiv:2006.11970 (2020)
7. Grzegorczyk, M., Husmeier, D.: Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. Mach. Learn. **71**(2–3), 265 (2008)
8. He, R., Tian, J., Wu, H.: Structure learning in Bayesian networks of a moderate size by efficient sampling. J. Mach. Learn. Res. **17**(1), 3483–3536 (2016)
9. Heckerman, D., Geiger, D., Chickering, D.M.: Learning Bayesian networks: the combination of knowledge and statistical data. Mach. Learn. **20**(3), 197–243 (1995)
10. Huang, B., Zhang, K., Lin, Y., Schölkopf, B., Glymour, C.: Generalized score functions for causal discovery. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1551–1560 (2018)
11. Jaakkola, T., Sontag, D., Globerson, A., Meila, M.: Learning Bayesian network structure using LP relaxations (2010)
12. Kalainathan, D., Goudet, O., Guyon, I., Lopez-Paz, D., Sebag, M.: Sam: structural agnostic model, causal discovery and penalized adversarial learning (2018)
13. Lachapelle, S., Brouillard, P., Deleu, T., Lacoste-Julien, S.: Gradient-based neural DAG learning. arXiv preprint arXiv:1906.02226 (2019)
14. Madigan, D., York, J., Allard, D.: Bayesian graphical models for discrete data. Int. Stat. Rev./Rev. Int. de Stat. 215–232 (1995)
15. Ng, I., Fang, Z., Zhu, S., Chen, Z., Wang, J.: Masked gradient-based causal structure learning. arXiv preprint arXiv:1910.08527 (2019)
16. Ng, I., Ghassami, A., Zhang, K.: On the role of sparsity and DAG constraints for learning linear DAGs. Adv. Neural Inf. Process. Syst. **33** (2020)
17. Nievergelt, J., Hinrichs, K.H.: Algorithms and Data Structures: With Applications to Graphics and Geometry. Prentice-Hall Inc, USA (1993)
18. Niinimaki, T., Parviainen, P., Koivisto, M.: Partial order MCMC for structure discovery in Bayesian networks. arXiv preprint arXiv:1202.3753 (2012)
19. Ott, S., Imoto, S., Miyano, S.: Finding optimal models for small gene networks. In: Pacific Symposium on Biocomputing (2004)
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, Inc., 2 edn. (1988)
21. Pearl, J.: Causality: models, reasoning, and inference. Economet. Theor. **19**(46), 675–685 (2003)
22. Peters, J., Janzing, D., Scholkopf, B.: Causal inference on discrete data using additive noise models. IEEE Trans. Pattern Anal. Mach. Intell. **33**(12), 2436–2450 (2011)

23. Peters, J., Mooij, J.M., Janzing, D., Schölkopf, B.: Causal discovery with continuous additive noise models. J. Mach. Learn. Res. **15**(1), 2009–2053 (2014)
24. Sachs, K., Perez, O., Peer, D., Lauffenburger, D.A., Nolan, G.P.: Causal protein-signaling networks derived from multiparameter single-cell data. Science **308**(5721), 523–529 (2005)
25. Scanagatta, M., de Campos, C.P., Corani, G., Zaffalon, M.: Learning Bayesian networks with thousands of variables. In: Advances in Neural Information Processing Systems, pp. 1864–1872 (2015)
26. Silander, T., Myllymaki, P.: A simple approach for finding the globally optimal Bayesian network structure. In: UAI (2006)
27. Spirtes, P., et al.: Causation, Prediction, and Search. MIT press (2000)
28. Spirtes, P., Meek, C., Richardson, T.: Causal inference in the presence of latent variables and selection bias. In: UAI (1995)
29. Teyssier, M., Koller, D.: Ordering-based search: a simple and effective algorithm for learning Bayesian networks. arXiv preprint arXiv:1207.1429 (2012)
30. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Mach. Learn. **65**(1), 31–78 (2006)
31. Viinikka, J., Hyttinen, A., Pensar, J., Koivisto, M.: Towards scalable Bayesian learning of causal DAGs. arXiv preprint arXiv:2010.00684 (2020)
32. Yu, Y., Chen, J., Gao, T., Yu, M.: DAG-GNN: dag structure learning with graph neural networks. arXiv preprint arXiv:1904.10098 (2019)
33. Yu, Y., Gao, T.: DAGs with no curl: efficient DAG structure learning. In: Advances in Neural Information Processing Systems (NeurIPS) Workshop on Causal Discovery and Causality-Inspired Machine Learning (2020)
34. Yu, Y., Gao, T., Yin, N., Ji, Q.: DAGs with no curl: an efficient DAG structure learning approach. In: International Conference on Machine Learning, pp. 12156–12166. Pmlr (2021)
35. Yuan, C., Malone, B.: Learning optimal Bayesian networks: a shortest path perspective. J. Artif. Intell. Res. **48**, 23–65 (2013)
36. Zhang, J.: On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. Artif. Intell. **172**(16–17), 1873–1896 (2008)
37. Zheng, X., Aragam, B., Ravikumar, P.K., Xing, E.P.: DAGs with no tears: continuous optimization for structure learning. In: Advances in Neural Information Processing Systems, pp. 9472–9483 (2018)
38. Zheng, X., Dan, C., Aragam, B., Ravikumar, P., Xing, E.P.: Learning sparse nonparametric DAGs. In: International Conference on Artificial Intelligence and Statistics (2020)

GCompletor: A Graph-Based Deep Learning Method for Traffic State Imputation on Urban Road Networks

Kaijie Li[1,2], Juanjuan Zhao[2(✉)], Li Yan[3], Xitong Gao[2], Ye Li[4], and Kejiang Ye[2]

[1] Southern University of Science and Technology, Shenzhen, China
likj2023@mail.sustech.edu.cn
[2] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
{jj.zhao,xt.gao,kj.ye}@siat.ac.cn, liye@szbit.cn
[3] The School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, China
li.yan.88@xjtu.edu.cn
[4] Shenzhen Insititute of Beidou Applied Technology Co., Ltd., Shenzhen, China

Abstract. Complete traffic data is the premise for traffic strategy making. However, due to the constraints of data collection, communication failures and other reasons, we may collect incomplete traffic states inevitably. The common idea of existing completion methods is to learn the latent representation reflecting the spatiotemporal correlation in the traffic data. However, due to insufficient influencing factors considered and limited capability of spatiotemporal correlation modeling, existing methods need further improvement, especially for the scenes with high missing rates. In this paper, we propose a novel traffic state imputation method GCompletor using Graph-based Encoder-Decoder framework, which enriches the features of each road by considering the physical features (road grade, direction, etc.), and organize all traffic features into a graph-based sequence. Then the sequence is fed into a novelly designed Encoder-Decoder component, where the spatiotemporal dependencies of each road is learned through extended GAT and BiGRU-CNN hybrid method. Experimental results demonstrate that GCompletor achieves better imputation performance than the state-of-the-art approaches. The source code is available at https://github.com/zfrInSIAT/GCompletor.

Keywords: Imputation Method · ITS · Missing Traffic Data

1 Introduction

Nowadays, with the development of sensing technology, we can collect the urban road traffic states through various traffic sensing devices [1]. For example, using ground loops [2] or cameras [3], we can collect the speed, flow, density on roads.

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 461–477, 2025.
https://doi.org/10.1007/978-3-031-78172-8_30

Additionally, GPS positioning equipment allows us to track vehicle speeds on these roads [4]. However, in real-world situations, factors like cost limitations, hardware or software errors, and communication disruptions prevent us from collecting traffic data on all roads. Incomplete traffic data reduces the accuracy of analysis and hampers the effectiveness of traffic management systems [1]. This paper aims to estimate the missing traffic states in urban road networks.

Existing traffic data imputation methods which extract the low-dimensional latent factors that contain spatiotemporal correlation of the traffic data have been proved to be effective. One main category is based on matrix/tensor factorization, such as PPCA [5], Tucker decomposition [6,7]. However, they are essentially linear model, which can not capture the complex spatial dependency of traffic data. Recently, deep generative models have been applied to address missing data issues, including traffic state imputation. These models use autoencoder-based approaches to capture the nonlinear spatial-temporal dependencies in traffic flow data. However, further improvements are necessary as these current methods severely overlook key influencing factors and inadequately model spatiotemporal correlations. The details of the limitation are as follows:

First, traffic states result from vehicle movements influenced by road attributes (e.g. number of lanes, lane width and so on) and traffic flow transitions. However, existing methods often overlook these factors. *Second*, current methods fail to fully capture spatiotemporal traffic correlations. Traffic states exhibit local (among adjacent roads or recent time slots) and global (over long-term periods) dependencies. Most existing deep generative models organize data into structures, which can not effectively support local spatiotemporal correlation modelling.

In this paper, we propose a novel Graph based Encoder-Decoder framework GCompletor for traffic state imputation in complex urban road traffic networks. Our contributions are as follows:

- To enable GCompletor to learn the spatiotemporal correlation of traffic states among roads sufficiently, we enrich the features of each road by considering both of the physical attributes and the observed incomplete traffic states, and organize the features across observed time slots into a graph sequence.
- GCompletor completes unobserved road traffic states by modelling both of the local-global spatial and temporal dependency based on the graph sequence. It uses Extended-GAT to learn the spatial dependency among roads by considering road attributes, and uses BiGRU to learn the traffic state evolution and anti-evolution patterns from temporal perspective through making full use of the traffic data before and after the missing time slot.
- We conduct extensive experiments of traffic state imputation on real-world data in the city of Shenzhen, China. The results show that GCompletor performs better than all other methods, particularly in scenarios with high rates of missing data, including random and continuous missing patterns.

2 Related Work

Existing traffic state imputation methods can be divided into four categories: neighbor based methods, prediction based methods, matrix/tensor decomposition based methods and autoencoder based methods.

Neighboring imputation methods are initial methods, which fill the missing data with a weighted average of the corresponding observed data, and can be divided into temporal neighbor methods, spatial neighbor methods and pattern neighbor methods. The temporal neighbor methods complete the missing data of a road based on the traffic states of consecutive time intervals [8]. The spatial neighbor methods complete the missing data of a road based on the traffic data of adjacent roads at the same time [9]. The pattern neighbor methods complete the missing data of a road by using the observed data of the road on historical days with the same evolution pattern [10]. All these methods can only model the local and stable spatiotemporal correlation between traffic states, and rely on the premise that the historical information from neighbors is complete.

The prediction based imputation methods use statistics or machine learning techniques, including ARMA [11], Bayesian Network [12], Feedforward Neural Network [13], etc. Recently, deep learning has changed traffic prediction architecture dramatically and brings more opportunities [14,15]. The RNN and its variants such as LSTM and GRU [15,16] are used to model the temporal correlation. GCN and its variants are used to model spatial correlation among roads. However, the prediction models are based on the precondition that the historical observed data is complete. Furthermore, they can not make full use of the traffic states after the time of missing data, but the information is important due to the local temporal correlation of traffic data.

The main idea of matrix/tensor decomposition based methods is that the traffic state matrices or tensors are low rank due to the intrinsic spatiotemporal correlation. Probabilistic PCA (PPCA) [5] and Bayesian PCA (BPCA) were proposed to project the incomplete traffic flow to low dimensional latent factors. Chen et al. [17], Ran et al. [7] introduced various tensor based models to impute traffic data by considering time dimensions(day of week and hour of day). However, matrix/tensor decomposition based methods are essentially linear models. They can not effectively support local spatiotemporal correlation modelling. Although some methods use local smoothing methods to capture the local correlation, they can not distinguish local correlations between roads under various spatiotemporal contexts.

The above mentioned methods belong to Discriminant Models. The performance largely depends on the amount of information embedded in the observation data, and the degradation is obvious when the data missing rate is very high. As a branch of self supervised learning technology, autoencoder based methods can capture the intrinsic features of data, and have shown their superiority in traffic tasks. This kind of methods contains two steps. It first learns the lower dimensional feature representations at the encoder component by some bottleneck layers, and then fills the missing data directly in decoder component by some reconstruction layers. Some studies have applied stacked encoder (SAE) [18], denoising stacked autoencoders [19] to estimate the missing data.

Although the imputation performance has been improved to a certain extent, imputation methods need further improvement due to the limitations as mentioned in Sect. 1, including the insufficient influencing factors considered and limited spatiotemporal correlation modelling of traffic data. In this paper, a graph based Encoder-Decoder model is proposed to complete the traffic state. This model captures the spatiotemporal dependencies with the data, including road physical attributes and observed sparse traffic states in an urban road network.

3 Overview

In this section, we first introduce some basic definitions (Table 1) and then formulate the traffic data imputation problem. Finally, we briefly describe the architecture of our method GCompletor.

3.1 Definitions

Definition 1 *(Road Network).* A road Network is represented by $G=(R, E, F)$. The $R = \{r_1, r_2, \cdots, r_N\}$ is the collection of roads. $F \in R^{N \times D_v}$ is the collection of physical attributes of all roads (e.g., road grade, speed limit and road length) where v corresponds to the road physical attributes and D_v is the total number of them. The ith row of F denoted as f_i is the attributes of road r_i. The $E \in R^{N \times N}$ is the collection of the edges in the graph, represented by adjacency matrix. If the roads r_i and r_j are connected, they are neighbors and the value $(E)_{i,j} = e_{ij}$ is 1, otherwise the value is 0.

Definition 2 *(Traffic State).* We divide a day into T time slots with the same time interval. The traffic states with missing data in a road network G on a day are denoted as matrix $X \in R^{T \times N}$, where $(X)_{t,i} = x_{t,i}$ represents the traffic state of the road r_i at the time slot $t \in [1, T]$. A mask matrix $\Omega \in R^{T \times N}$ with the same shape as X is used to mark the locations of the missing elements. The value $(\Omega)_{i,j}$ is 1 if $x_{t,i}$ is collected, otherwise the value is 0. In this paper, we use a partially observed vector $x_{(t)} = \{x_{t,1}, \cdots, x_{t,N}\}$ to represent the traffic states of all roads at the time slot t.

Problem Definition: Given a road network G on a day, the task of traffic data imputation is to estimate the missing traffic data in the traffic matrix X. It can be formulated as $\hat{X} = f_c(X, G, \Omega)$, where f_c is the imputation model and $\hat{X}$ is the completed or reconstructed traffic matrix.

3.2 Framework

Figure 1 presents the framework of our method GCompletor. It consists of two components: Traffic Graph Sequence Construction and Encoder-Decoder based Traffic Data Imputation. The former component aims to organize the observed traffic states along the observed time slots, road physical attributes, and relations between roads into a graph sequence. The latter component aims to impute

Table 1. Notations in This Paper

Symbol	Description
G	Road Network
R	Collection of roads
r_i	A road in R
F	Collection of physical properties of all roads
f_i	Properties of road r_i
X	Observed traffic states of all roads
$x_{t,i}$	Traffic state of road v_i at time t
Ω	Mask matrix of the missing elements in X
u_t^i	Local spatial representation of node r_i at time t
w_t^i	Local spatiotemporal representation of node r_i at time t
$x_{(t)}$	The traffic states of all roads at time t

the missing traffic data through learning the latent and compact spatiotemporal representation hidden in the graph sequence of observed traffic data. More specifically, in the encoding step, at each time slot, it first learn each road's local spatiotemporal representation through extended GAT-based attention mechanism and BiGRU-CNN based on the traffic data before and after the time slot. Then the representation sequence of each road along all observed time slots is fed into multiple dense bottleneck layers to learn the road's global latent representation. In the decoding step, it uses reconstruction layers to reconstruct the inputs through dimension ascending. Finally, the reconstructed traffic flow matrix is used to impute the missing data.

4 Traffic Graph Sequence Construction

The correlation of the traffic states between roads is related to their geographical connection, road physical attributes and the traffic transition between roads. In

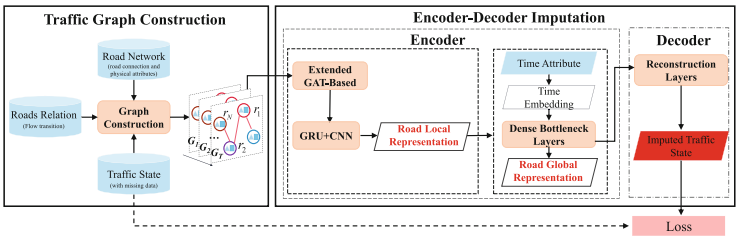


Fig. 1. Framework of GCompletor

this section, we organize all these traffic data over observed time slots into a graph-based sequence, where the neighborhood between roads is determined by their geographical physical connection and time-dependent traffic transition.

More specifically, for each time slot t of a day, we construct a time-dependent traffic graph $G_t = \left(R, E, F, F_E^t, F_V^t\right)$, where R, E and F refer to the collection of roads, edges, and road properties as Definition 1. F_V^t and F_E^t are time-dependent (dynamic) features of roads and edges described as follows.

Dynamic features of nodes $F_V^t \in R^N$: is the $x_{(t)}$ of traffic matrix X, referring to the observed traffic states of all roads at the time slot t. The ith item of F_V^t is the traffic state of road r_i. Note that F_V^t is sparse since some traffic states are not observed. There is no standard approach for using sparse data as input of neural networks. Most of the existing studies deal with sparse inputs by pre-computing an estimate of the missing values. In our case, since the traffic states have obvious periodicity, we use the average value of the traffic states at the same time slot on history days to pre-fill the missing data.

Dynamic features of edges $F_E^t \in R^{N \times N}$: refers the attributes of the flow transition between two connected roads at the time slot t. That can help the model to capture the local spatial correlation between roads even with limited traffic data. However, we can not obtain the flow transition information. Instead, we use the similarity of traffic state evolution trends between two roads to indirectly express their transition relationship. The intuition is that the greater the proportion the traffic states of two connected roads both rise or fall is, the stronger the correlation of their traffic states may be. The feature $f_{i,j}^t$ of edge $e_{i,j}$ is set to the the similarity of traffic state evolution trends between two roads r_i and r_j, which is defined by Equ(1).

$$Sim(r_i, r_j) = e_{i,j} \times \left(\frac{\mathbf{Cnt}(v_i^t \geq \hat{v}_i^t, v_j^t \geq \hat{v}_j^t) + \mathbf{Cnt}(v_i^t < \hat{v}_i^t, v_j^t < \hat{v}_j^t)}{Tcn_{i,j}^t} \right) \tag{1}$$

where $\mathbf{Cnt}(v_i^t \geq \hat{v}_i^t, v_j^t \geq \hat{v}_j^t)$ and $\mathbf{Cnt}(v_i^t < \hat{v}_i^t, v_j^t < \hat{v}_j^t)$ represent the number of times that the traffic states of two roads rise or fall compared with history average speeds $\hat{v}_i^t$ and $\hat{v}_j^t$ at time slot t on history days. $Tcn_{i,j}^t$ represents the total number of observations. In practice, if the traffic data on history days at time slot t is insufficient, the traffic states near the time slot t can be used together due to the strong temporal correlation of recent traffic states.

Based on the above process, we construct a sequence of traffic state graphs for T time slots of a day, denoted as $\mathcal{G} = (G_1, G_2, \cdots G_T)$. This will be fed into the encoder-decoder based traffic data imputation component.

5 Encoder-Decoder Based Traffic Data Imputation

In the previous section, we construct a sequence of traffic state graphs for all roads over T time slots of a day. In this section, based on the underlying idea of general autoencoder, the encoder-decoder component first learns the lower dimensional representation of the traffic data for each road in the encoding step,

and then reconstructs the traffic data in the decoder step. The major difference of encoder-decoder component with general autoencoder lies in that it sufficiently captures the local spatiotemporal correlation through combining the physical properties of each road (node features) and the flow transition between roads (edge features).

5.1 Local Spatiotemporal Dependency Representation

Encoder-decoder based traffic data imputation first learns the local spatial embedding representation of each road at each time slot based on a novelly designed GAT-based component through aggregating the traffic states of neighbor roads. Then the BiGRU-CNN component is used to learn the traffic's local temporal dependency of each road at each time slot with the local traffic information before and after the time slot.

Local Spatial Dependency Learning. Given a time slot t and traffic graph $G_t = \left(R, E, F, F_E^t, F_V^t\right)$, we use the idea of GAT mechanism to extract the spatial representation for each node r_i by aggregating the traffic states of the neighbor nodes $N_i = \{r_i | e_{i,j} = 1\}$.

Since GAT mechanism learns the attention coefficient between nodes only based on node features, we extend it by using the features of nodes and edges together.

Specifically, as shown in the Fig. 2, the features of node r_i and node r_j in G_t are obtained through combining the physical and dynamic features. That is $h_i{=}(F)_i||(F_V^t)_i$ and $h_j{=}(F)_j||(F_V^t)_j$ respectively. The edge features between the two nodes are $h_{ij} = \left(F_E^t\right)_{ij}$. Based on that, we transform the features into compact features through a learnable linear transformation W, including $z_i{=}\,\tilde{\mathrm{W}}h_i$, $z_{\mathrm{j}}{=}\,\tilde{\mathrm{W}}h_j$ and $z_{\mathrm{i,j}}{=}\,\hat{\mathrm{W}}h_{ij}$. The transformation matrix $\tilde{\mathrm{W}}$ and $\hat{\mathrm{W}}$ are the parameters to be learned. On this basis, we perform operations $w_{ij} = a(z_i, z_j, z_{ij})$ to calculate the importance of node r_i to node r_j. The operation a consists of a single-layer feedforward neural network and leakyReLu nonlinear activation function. In order to make the coefficients easy to compare on different nodes, we use softmax normalization function to regularize all neighbor nodes by Equ(2).

$$\alpha_{ij} = \mathbf{Softmax_j}(w_{ij}) = \frac{\exp(w_{ij})}{\sum_{k \in N_i} \exp(w_{ik})} \tag{2}$$

In conclusion, the attention coefficient can be expressed by Equ(3), where $.^T$ and $||$ represent the transposition and connection operation. After a convolution operation, we get a vector $h_i' = \sigma(\sum_{j \in N_i} a_{ij} \mathrm{W} \boldsymbol{h}_j)$.

$$\alpha_{ij} = \frac{\exp\left(\mathbf{LeakyReLU}\left(\boldsymbol{a}^T[z_i||z_j||z_{ij}]\right)\right)}{\sum_{k \in N_i} \exp\left(\mathbf{LeakyReLU}\left(\boldsymbol{a}^T[z_i||z_k||z_{ik}]\right)\right)} \tag{3}$$

Since the traffic states of one-hop neighbor nodes may be missing, the aggregated information might not correctly reflect the real-time information. We use

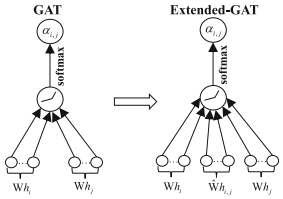


Fig. 2. Extended-GAT

two ways to alleviate that. First, we use multi-layer graph attention mechanism to aggregate the information of multi-hop neighbor nodes. Second, we use the multi-head mechanism to stabilize the learning process of self attention. It learns the local spatial representation of nodes in multiple views at each time slot through multiple convolution kernels W. If there is K head attentions, we need to merge the vectors together to obtain the local spatial representation of node r_i at the time slot t. That is $u_t^i = \mathop{\Big\|}_{k=1}^{K} (\sigma(\sum_{j \in N_i} a_{ij}^k \mathrm{W}^k h_j))$.

Local Temporal Dependency Representation. Based on above analysis, for each road r_i, we can get the local spatial representation sequence $\{u_1^i, u_2^i \cdots u_T^i\}$ among the T time slots. For a road, the traffic state at a time slot t is evolved or anti-evolved from the traffic states at the time slots before or after. Moreover the traffic state at a time slot is more related with its adjacent time slots.

Accordingly, for each time slot t, we use BiGRU and CNN to capture the temporal correlation. Specifically, given the spatial feature representation sequence of road r_i among the T time periods $\{u_1^i, u_2^i \cdots u_T^i\}$, the BiGRU learns the traffic evolution and anti-evolution patterns from the two sequences in positive order $X^l = \{u_1^i, u_2^i \cdots u_T^i\}$ and reversed order $X^r = \{u_T^i, u_{T-1}^i \cdots u_1^i\}$ respectively. The details of BiGRU model refer to the work [20]. For each time slot t, the BiGRU outputs two vectors representing the evolution and anti-evolution pattern of the road r_i. They are concatenated to get a vector for road r_i at time slot t, which is denoted as br_t^i.

In addition, CNN(Convolutional Neural Network) is used to model the strong traffic dependencies between a given time slot and its neighbor time slots. It captures the recent dependencies by aggregating the representation of traffic states at neighbor time slots. We use multi-channel convolution operation and the step size is set to 1. In order to obtain the output with the same length as the input sequence $\{u_1^i, u_2^i \cdots u_T^i\}$, we choose to use values at near time slots to fill the input data. Through CNN layers, we can get a vector for each time slot t of the road r_i, denoted as cr_t^i.

The final local spatiotemporal representation of road r_i at time t is obtained by combining the two vectors br_t^i and cr_t^i, and denoted as a vector $w_t^i = br_t^i || cr_t^i$.

5.2 Global Dependency Learning and Data Imputation

For each road r_i, we can get the local spatiotemporal representation sequence $\{w_1^i, w_2^i \cdots w_T^i\}$ over T time slots. In order to learn the global correlation of traffic data, we use multi-narrow hidden layers, same as the idea in the autoencoder, to learn a dense and low-dimensional vector. Since the traffic state has obvious periodicity, we use the embedding of time attributes together to generate the global representation Y_i.

More specifically, we have one embedding table for each time factor, including time of day, day of the week. These embeddings are concatenated to the local spatiotemporal representation sequence, and then passed through narrow hidden layers to obtain the final low-dimensional vector.

In the decoding step, we use multiple wide fully connected layers to reconstruct the traffic states by $\hat{a}_i = f_d(X_i W + b)$. The GCompletor uses neural network based framework. It is trained by using stochastic gradient descent and requires a loss function, which is to evaluate how well a specific algorithm models the given data. If the output deviates too much from actual results, the loss function would generate a very large error. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction. The loss function of GCompletor is defined as Equ(4).

$$L = \left\| (\hat{X} - X) \cdot ((1 - \Omega)) \right\|_F^2 \tag{4}$$

Through optimization, we minimize the gap between observed and reconstructed values. At the same time, we can obtain a reconstructed traffic flow data matrix $\hat{X}$, where the missing data are also constructed. Based on the matrix $\hat{X}$, the complete traffic state is obtained by $\hat{X}_{raw} = X \odot \Omega + \hat{X} \odot (1 - \Omega)$.

6 Experiment

6.1 Experiment Settings

Datasets. We evaluate our method's performance through extensive experiments on real-world datasets from urban road network in Shenzhen, China. The datasets are from Shenzhen Transportation Agencies and some sample data can be downloaded on the data open platform of Shenzhen municipal government, comprising two types of data sources. One is the static information of road network, including the topology and road's physical attributes[1] (as shown in the Table 2). Another is the traffic state (speed) of each road at each time slot[2]. The whole data contains 6 months ranging from Jan to June in 2018, including 4248 roads. Each day is divided into 288 time slots with an interval of 5 min.

[1] https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_00403588.
[2] https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_00403590.

Table 2. Road Attributes

Attribute	Examples
Road grade	Expressway, Secondary, Branch Road, etc.
Speed limit	30 km/h, 40 km, 50 km/h, etc.
Road length	800 m, 1 km, etc.
Direction	South-North, North-South, East-West, etc.

Missing Mode Setting. We consider two missing modes on traffic data: missing completely at random (MR), and missing at continuity (MC).

MR: The missing values may occur because of communication failure or sparsity of data acquisition (vehicle trajectories). These are some isolated points randomly scattered and independent of each other completely.

MC: The missing values may occur due to a physical damage or maintenance backlog. These values are related to their temporal or spatial neighboring readings. Thus, this mode appears to be some sequential points at the same sensor.

Model Setting. In experiments, we set the parameters of our model as follows. The numbers of attention heads and graph convolution layers in the local spatial dependency learning component are set to 4 and 2 respectively, based on best settings from validation data. The layers of BiGRU and CNN of the local temporal learning dependency component is set to 2. In addition, stochastic gradient descent with momentum is used as the optimizer. We evaluated our model for different momentum values from 0 to 0.9. The learning rate is tuned within 0.01−− 0.001 and set 0.008 after testing.

Evaluation Metrics. The constructed values for the missing traffic states are compared with the real values to evaluate each method. The closer the completed values are to the real ones, the better the model is. We use two most widely utilized evaluation RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error) to evaluate our method. They are defined by Eq. (5) and Eq. (6), where N is the total number of missing items in the test data, and y_i^{real} and y_i^{impu} are the i_{th} elements of the real value and estimated value respectively.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\left|y_i^{real} - y_i^{impu}\right|\right)^2} \tag{5}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i^{real} - y_i^{impu}}{y_i^{real}}\right| \times 100\% \tag{6}$$

Baselines. We compare our approach with baselines, which can be categorized as prediction based methods (ARIMA, STGCN, GAMAN), neighbor based methods(Historical Average, KNN), matrix/tensor decomposition based methods (BTRMF, BGCP), and deep generative based methods (SA, VA).

Prediction Based Methods: They use traffic data before the missing time slot as input to estimate the missing data. They predict the missing value based on the time order, considering the previously predicted value as real and then predict subsequent missing values. These methods take a time series as input.

ARIMA [21]: Auto-Regressive Integrated Moving Average is a well known model for predicting time series, which fits the observed time series into a parametric model to predict the future traffic data.

STGCN [22]: Spatio-Temporal Graph Convolutional Network consists of graph convolutional layers and 1D convolution layers. It can capture both spatial and temporal dependencies in traffic prediction.

GAMAN [23]: Graph Multi-Attention Network uses an encoder-decoder architecture with spatiotemporal attention mechanism to predict traffic states.

Interpolation Based Methods: They fill the missing data with a weighted average of corresponding observed data.

HA: Historical Average method estimates missing traffic data by averaging the traffic states of the target road at the same time slot on the past days.

KNN: K-Nearest Neighbor method, also known as mode interpolation, estimates the missing traffic state of a road by averaging the observed traffic data from historical days with similar traffic transition patterns [10].

Matrix/tensor Decomposition Methods: These methods use the idea that the traffic data matrix or tensor has the characteristics of low rank due to the intrinsic spatiotemporal correlation.

BTRMF [24]: It completes the missing traffic states by integrating low-rank matrix decomposition and vector autoregressive (VAR) process into a single probabilistic graphical model, characterizing consistencies in large-scale time series data. It uses a matrix representation of road segments and time slots as input.

BGCP [25]: It extends the Bayesian Probabilistic Matrix Decomposition model to higher-order tensors and apply it to spatiotemporal traffic data imputation tasks. It uses third-order tensor representation of $road \times day \times timeslot$ as input.

Deep Generative Based Methods: The autoencoder based methods can capture the intrinsic features of data. They estimate the missing data by constructing an encoder-decoder based framework to learn the data distribution and the global spatiotemporal correlation between traffic data.

SA [18]: Stacked Autoencoder uses two stacked autoencoders to impute the missing values, which can simultaneously consider the spatial and temporal dependencies. It uses a matrix of $road \times timeslot$ as input.

VA [26]: Variational Denoising Autoencoder corrupts the input by adding random noise, which can enable our model to better learn the relationship between data. It uses the traffic matrix of $road \times timeslot$ as input.

6.2 Results and Analysis

In this section, extensive experiments are conducted to answer the following research questions. i) What is the performance of our model compared with other baselines on the two missing data modes (MR and MC) where the missing rates vary from 20% to 80%. ii) Does each of the components of GCompletor and the road attributes make a contribution to improving the model performance?

Performance Comparison with Baselines. Two metrics, i.e., RMSE and MAPE, are used to evaluate the performance of our proposed model and all baseline models mentioned above. The results are summarized in Tables 3 and 4.

We observe that our model achieves the best performance in both MR and MC cases with different missing rates. The interpolation methods perform the weakest. Because these methods are only based on the traffic states of periodicity and road proximity correlation, and use the simple average to perform traffic data imputation. This is only suitable for traffic state imputation in stable scenarios. The prediction based methods are worse than matrix/tensor decomposition based method. Although the prediction methods can capture the spatiotemporal nonlinear correlation of traffic states by using GCN and CNN, they do not make full use of the traffic data after the time slots. Furthermore, most prediction methods are based on the assumption that the input data is complete, so the advantages of deep learning cannot be fully realized. The tensor based methods

Table 3. Comparison of different methods in missing mode MR

Methods	20%		40%		60%		80%	
	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)
HA	3.034	29.375	3.136	30.284	3.413	31.481	3.938	36.032
KNN	2.987	30.403	3.093	31.557	3.350	33.543	3.812	35.865
ARIAM	2.688	26.882	2.819	28.676	3.054	30.683	3.456	34.126
STGCN	2.351	26.932	2.595	28.011	2.775	29.979	3.112	33.452
GAMAN	2.187	23.013	2.463	26.323	2.687	28.402	2.978	34.123
BTRMF	2.101	21.997	2.323	25.497	2.539	28.227	3.001	32.621
BGCP	1.997	20.876	2.135	24.162	2.356	26.873	2.603	30.537
SA	2.082	21.125	2.221	22.995	2.326	25.573	2.683	29.157
VA	2.130	21.942	2.247	23.346	2.333	25.798	2.687	29.339
GCompletor	**1.884**	**19.093**	**2.017**	**21.124**	**2.125**	**23.213**	**2.335**	**25.934**

Table 4. Comparison of different methods in missing mode MC

Methods	20%		40%		60%		80%	
	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)
HA	3.373	32.351	3.487	33.439	3.733	34.806	4.253	40.051
KNN	3.274	33.512	3.355	34.831	3.743	35.162	4.073	39.775
ARIAM	2.927	29.474	3.137	31.513	3.342	33.777	3.792	37.791
STGCN	2.644	29.462	2.822	30.779	3.063	32.985	3.432	36.960
GAMAN	2.463	25.013	2.755	28.762	2.951	31.241	3.283	37.841
BTRMF	2.379	23.831	2.575	27.794	2.876	31.051	3.347	36.096
BGCP	2.271	20.485	2.459	26.233	2.633	29.432	2.93	33.643
SA	2.339	22.803	2.373	23.774	2.571	26.760	2.830	30.980
VA	2.301	22.602	2.452	24.259	2.519	27.003	2.894	31.128
GCompletor	**2.152**	**20.595**	**2.311**	**22.931**	**2.455**	**25.313**	**2.651**	**28.503**

are slightly better than the matrix based methods. The reason is that the tensor based methods can utilize the spatiotemporal information of multiple modes simultaneously, while matrix decomposition based method can only mine the data correlation in a single traffic mode.

The two autoencoder based methods perform better than interpolation based methods and matrix/tensor decomposition based methods due to the better capacity of learning the complex spatiotemporal dependencies in traffic state sequences. However, it's worse than our method GCompletor. For example, our model reduce the Mean Absolute Percentage Error(MAPE) from 29.157% to 25.934%, from 30.980% to 28.503% on the MR and MC missing modes when the missing ratio is higher than 80%. This is because, compared with them, we consider multi-view attributes of roads and use BiGRU and CNN to learn the local spatiotemporal correlation of traffic states, which is helpful for our model to comprehensively capture the correlation relationship between data, so as to obtain a more effective low-dimensional feature representation.

As compared to MR cases, we can observe that all methods in MC cases suffer much more severe degradation, which suggests that the traffic states are highly related to the traffic states of the spatial and temporal neighboring points. Compared with other methods, GCompletor can achieve better results for continuous missing cases, because it can better model the local spatial correlation of traffic states by using road attributes and the flow transition between roads. These spatial relationships, strengthened by prior knowledge, can better help the model identify and capture the local spatial correlation of traffic data.

6.3 Model Ablation Analysis

As mentioned above, compared with the existing autoencoder based methods, GCompletor considers more road attributes and uses two components to capture

Table 5. Model Ablation Results in MR Mode

Methods	20%		40%		60%		80%	
	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)
GCompletor$_g$	2.020	20.628	2.141	22.665	2.256	24.863	2.519	27.743
GCompletor$_s$	2.013	20.279	2.144	22.046	2.243	24.729	2.485	27.668
GCompletor$_d$	1.969	19.985	2.108	22.091	2.228	24.287	2.446	27.316
GCompletor$_l$	1.954	19.692	2.087	21.643	2.181	23.814	2.439	26.572
GCompletor$_{LSC}$	2.078	20.984	2.189	22.709	2.255	24.897	2.615	28.267
GCompletor$_{LTC}$	2.109	21.447	2.125	22.967	2.306	25.515	2.627	28.823
GCompletor	**1.882**	**19.095**	**2.016**	**21.125**	**2.123**	**23.211**	**2.338**	**25.939**

Table 6. Model Ablation Results in MC Mode

Methods	20%		40%		60%		80%	
	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)	RMSE	MAPE(%)
GCompletor$_g$	2.275	22.128	2.343	23.457	2.534	26.077	2.765	30.069
GCompletor$_s$	2.234	21.503	2.349	23.402	2.517	25.996	2.744	29.612
GCompletor$_d$	2.229	21.553	2.342	23.203	2.493	25.049	2.724	29.264
GCompletor$_l$	2.196	21.147	2.326	23.15	2.473	25.674	2.686	29.037
GCompletor$_{LSC}$	2.373	23.063	2.459	24.514	2.643	26.745	2.794	30.435
GCompletor$_{LTC}$	2.304	22.408	2.366	23.617	2.555	26.604	2.798	30.514
GCompletor	**2.156**	**20.593**	**2.319**	**22.932**	**2.458**	**25.312**	**2.653**	**28.508**

the intrinsic spatiotemporal correlation in traffic data, so as to achieve better traffic data completion accuracy. This section further analyzes the effectiveness of road physical features and the GCompletor components respectively.

Road Physical Attributes. In order to verify the effectiveness of road physical features, including road grade(f_g), speed limit(f_s), direction(f_d), road length(f_l), and we remove each of them and input the remaining features into GCompletor. The results are shown in Tables 5 and 6 for two missing modes. We observe that the prediction performance decreases no matter which feature is removed. The results indicate that each of the features makes a contribution to GCompletor. Moreover, as we can see in the two tables, their effects are different, the road grade is the most important feature, followed by speed limit and direction.

Different Components. We also test the effectiveness of each model component of GCompletor, including local spatial component (LSC) and local temporal component (LTC). The results are shown in Table 5 and Table 6. No matter which component is removed, the model performance becomes worse. This

demonstrates that they are both necessary to improve prediction accuracy. Moreover, the LSC component plays a more significant role in MC mode than MR mode, because when the traffic states of a road are continuously missing, it is difficult to estimate the missing data by capturing the temporal correlation within the time series of that road's traffic states. In contrast to MR mode, MC mode depends more on the traffic states of neighboring roads. This limitation can be better addressed by learning the local spatial correlations between roads.

7 Conclusion

This paper addresses traffic state imputation in intricate urban road networks with sparse traffic data and physical attributes. We design a novel model called GCompletor with two key components: traffic graph sequence construction and encoder-decoder based traffic data imputation. The first one organizes the observed traffic states and the road physical attributes into a graph sequence by considering their physical connection and similarity of traffic evolution trends. The second one then imputes the missing traffic data through learning the latent and compact spatiotemporal representations hidden in the observed traffic data through extended GAT-based attention mechanism and BiGRU-CNN. More importantly, we implement and evaluate GCompletor on real-world datasets in the city of Shenzhen, China. The experiments show that the proposed model outperforms all baseline methods in both MR and MC missing modes across 4 different missing rates.

Acknowledgement. This work is supported by National Natural Science Foundation of China (No. 62372443, No. 62376263), Guangdong Basic and Applied Basic Research Foundation (No. 2023B1515130002), Natural Science Foundation of Guangdong (2024A1515011970, 2024A1515030209)

References

1. Seo, T., Bayen, A.M., Kusakabe, T., Asakura, Y.: Traffic state estimation on highway: a comprehensive survey. Annu. Rev. Control. **43**, 128–151 (2017)
2. Nayak, P., Garetto, M., Knightly, E.W.: Modeling multi-user wlans under closed-loop traffic. IEEE/ACM Trans. Networking **27**(2), 763–776 (2019)
3. Chakraborty, P., Adu-Gyamfi, Y.O., Poddar, S., Ahsani, V., Sharma, A., Sarkar, S.: Traffic congestion detection from camera images using deep convolution neural networks. Transp. Res. Rec. **2672**(45), 222–231 (2018)
4. Altintasi, O., Tuydes-Yaman, H., Tuncay, K.: Detection of urban traffic patterns from floating car data (FCD). Transp. Res. Procedia **22**, 382–391 (2017)
5. Qu, L., Li, L., Zhang, Y., Hu, J.: PPCA-based missing data imputation for traffic flow volume: a systematical approach. IEEE Trans. Intell. Transp. Syst. **10**(3), 512–522 (2009)
6. Goulart, J.D.M., Kibangou, A., Favier, G.: Traffic data imputation via tensor completion based on soft thresholding of tucker core. Transp. Res. Part C Emerg. Technol. **85**, 348–362 (2017)

7. Ran, B., Tan, H., Feng, J., Wang, W., Cheng, Y., Jin, P.: Estimating missing traffic volume using low multilinear rank tensor completion. J. Intell. Transp. Syst. **20**(2), 152–161 (2016)
8. Li, L., He, S., Zhang, J., Ran, B.: Short-term highway traffic flow prediction based on a hybrid strategy considering temporal-spatial information. J. Adv. Transp. **50**(8), 2029–2040 (2016)
9. Yin, W., Murray-Tuite, P., Rakha, H.: Imputing erroneous data of single-station loop detectors for nonincident conditions: comparison between temporal and spatial methods. J. Intell. Transp. Syst. **16**(3), 159–176 (2012)
10. Ni, D., Leonard, J.D., Guin, A., Feng, C.: Multiple imputation scheme for overcoming the missing values and variability issues in its data. J. Transp. Eng. **131**(12), 931–938 (2005)
11. Xu, J.R., Li, X.Y., Shi, H.J.: Short-term traffic flow forecasting model under missing data. J. Comput. Appl.**30**(4), 1117–1120 (2010)
12. Ni, D., Leonard, J.D.: Markov chain Monte Carlo multiple imputation using Bayesian networks for incomplete intelligent transportation systems data. Transp. Res. Rec. **1935**(1), 57–67 (2005)
13. Karlaftis, M.G., Vlahogianni, E.I.: Statistical methods versus neural networks in transportation research: differences, similarities and some insights. Transp. Res. Part C Emerg. Technol. **19**(3), 387–399 (2011)
14. Wang, J., Jiang, J., Jiang, W., Li, C., Zhao, W.X. (eds.): LibCity: an open library for traffic prediction. In: ACM (2021)
15. Ye, J., Zhao, J., Ye, K., Xu, C.: How to build a graph-based deep learning architecture in traffic domain: a survey. IEEE Trans. Intell. Transp. Syst. **23**(5), 3904–3924 (2020)
16. Ye, J., Zhao, J., Ye, K., Xu, C.: Multi-STGCnet: a graph convolution based spatial-temporal framework for subway passenger flow forecasting. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
17. Chen, X., He, Z., Wang, J.: Spatial-temporal traffic speed patterns discovery and incomplete data recovery via SVD-combined tensor decomposition. Transp. Res. part C Emerg. Technol. **86**, 59–77 (2018)
18. Han, L., Zheng, K., Zhao, L., Wang, X., Wen, H.: Content-aware traffic data completion in its based on generative adversarial nets. IEEE Trans. Veh. Technol. **69**(10), 11950–11962 (2020)
19. Duan, Y., Lv, Y., Liu, Y.L., Wang, F.Y.: An efficient realization of deep learning for traffic data imputation. Transp. Res. part C Emerg. Technol. **72**, 168–181 (2016)
20. BiGRU: Bigru. https://github.com/topics/bigru
21. Xu, D.W., Wang, Y.D., Jia, L.M., Qin, Y., Dong, H.H.: Real-time road traffic state prediction based on ARIMA and kalman filter. Front. Inf. Technol. Electron. Eng. **18**(2), 287–302 (2017)
22. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI, pp. 3634–3640 (2018)
23. Zheng, C., Fan, X., Wang, C., Qi, J.: GMAN: a graph multi-attention network for traffic prediction. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI, pp. 1234–1241 (2020)
24. Chen, X., Sun, L.: Bayesian temporal factorization for multidimensional time series prediction. IEEE Trans. Pattern Anal. Mach. Intell. **44**(9), 4659–4673 (2021)

25. Chen, X., He, Z., Sun, L.: A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation. Transp. Res. Part C Emerg. Technol. **98**, 73–84 (2018)
26. Chen, Y., Lv, Y., Wang, F.Y.: Traffic flow imputation using parallel data and generative adversarial networks. IEEE Trans. Intell. Transp. Syst. **21**(4), 1624–1630 (2019)

Author Index

A. Antonacopoulos et al. (Eds.): ICPR 2024, LNCS 15306, pp. 479–480, 2025.
https://doi.org/10.1007/978-3-031-78172-8