

Mufti Mahmud · Maryam Daborjeh ·  
Kevin Wong · Andrew Chi Sing Leung ·  
Zohreh Daborjeh · M. Tanveer (Eds.)

LNCS 15286

# Neural Information Processing

31st International Conference, ICONIP 2024  
Auckland, New Zealand, December 2–6, 2024  
Proceedings, Part I

1  
Part I

**ICONIP**  
**2024**



Springer

## Founding Editors

Gerhard Goos

Juris Hartmanis

## Editorial Board Members

Elisa Bertino, *Purdue University, West Lafayette, IN, USA*

Wen Gao, *Peking University, Beijing, China*

Bernhard Steffen , *TU Dortmund University, Dortmund, Germany*

Moti Yung , *Columbia University, New York, NY, USA*

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.

Mufti Mahmud · Maryam Doborjeh ·  
Kevin Wong · Andrew Chi Sing Leung ·  
Zohreh Doborjeh · M. Tanveer  
Editors

# Neural Information Processing

31st International Conference, ICONIP 2024  
Auckland, New Zealand, December 2–6, 2024  
Proceedings, Part I

*Editors*

Mufti Mahmud   
King Fahd University of Petroleum  
and Minerals  
Dhahran, Saudi Arabia

Kevin Wong   
Murdoch University  
Murdoch, WA, Australia

Zohreh Doborjeh   
Auckland University of Technology  
Auckland, New Zealand

Maryam Doborjeh   
Auckland University of Technology  
Auckland, New Zealand

Andrew Chi Sing Leung   
City University of Hong Kong  
Kowloon, Hong Kong

M. Tanveer   
Indian Institute of Technology Indore  
Indore, Madhya Pradesh, India

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-981-96-6575-4

ISBN 978-981-96-6576-1 (eBook)

<https://doi.org/10.1007/978-981-96-6576-1>

© The Editor(s) (if applicable) and The Author(s), under exclusive license  
to Springer Nature Singapore Pte Ltd. 2025

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.  
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

If disposing of this product, please recycle the paper.

## Preface

Welcome to the 31st International Conference on Neural Information Processing (ICONIP 2024) of the Asia-Pacific Neural Network Society (APNNS), held in Auckland, New Zealand, December 2–6, 2024.

The mission of the Asia-Pacific Neural Network Society is to promote active interactions among researchers, scientists, and industry professionals who are working in neural networks and related fields in the Asia-Pacific region. APNNS has Governing Board Members from 13 countries/regions - Australia, China, Hong Kong, India, Japan, Malaysia, New Zealand, Singapore, South Korea, Qatar, Taiwan, Thailand, and Turkey. The society's flagship annual conference is the International Conference on Neural Information Processing (ICONIP). The ICONIP conference aims to provide a leading international forum for researchers, scientists, and industry professionals who are working in neuroscience, neural networks, deep learning, and related fields to share their new ideas, progress, and achievements.

ICONIP 2024 received 1301 papers, of which 318 papers were accepted for publication in Lecture Notes in Computer Science (LNCS), representing an acceptance rate of 24.44% and reflecting the increasingly high quality of research in neural networks and related areas. The conference focused on four main areas, i.e., “Theory and Algorithms”, “Cognitive Neuroscience”, “Human-Centered Computing”, and “Applications”. All the submissions were rigorously reviewed by the conference Programme Committee (PC), comprising 767 PC members, and they ensured that every paper had at least three high-quality single-blind reviews. Each paper was further reviewed by at least one chair.

We would like to take this opportunity to thank all the authors for submitting their papers to our conference, and our great appreciation goes to the PC members and the reviewers who devoted their time and effort to our rigorous peer-review process; their insightful reviews and timely feedback ensured the high quality of the papers accepted for publication. We hope you enjoyed the research program at the conference.

March 2025

Mufti Mahmud  
Maryam Doborjeh  
Kevin Wong  
Andrew Chi Sing Leung  
Zohreh Doborjeh  
M. Tanveer

# **Organisation**

## **Honorary Chairs**

Nikola Kasabov

Derong Liu

Seiichi Ozawa

Jonathan Chan

Auckland University of Technology, New Zealand  
Southern University of Science and Technology,  
China

Kobe University, Japan

King Mongkut's University of Technology  
Thonburi, Thailand

## **Conference Chairs**

Maryam Daborjeh

Mufti Mahmud

Michael Witbrock

Auckland University of Technology, New Zealand  
King Fahd University of Petroleum and Minerals,  
Saudi Arabia

University of Auckland, New Zealand

## **Program Chairs**

Kevin Wong

Andrew Chi Sing Leung

Zohreh Daborjeh

M. Tanveer

Murdoch University, Australia

City University of Hong Kong, China

University of Auckland, New Zealand

IIT Indore, India

## **Finance Chairs**

Saori Tanaka

Terry Brydon

University of Electro-Communications, Japan

Auckland University of Technology, New Zealand

## **Competition Chairs**

Paul S. Pang

Goh Wen Bin Wilson

Federation University Australia, Australia

Nanyang Technological University, Singapore

## Special Session Chairs

Tingwen Huang  
Wei-Neng Chen  
Edmund Lai

Texas A&M University at Qatar, Qatar  
South China University of Technology, China  
Auckland University of Technology, New Zealand

## Tutorial and Workshop Chairs

Alex Sumich  
Kenneth Johnson  
Marzia Hoque Tania  
Akbar Ghobakhloo

Nottingham Trent University, UK  
Auckland University of Technology, New Zealand  
University of New South Wales, Australia  
Auckland University of Technology, New Zealand

## Award Chairs

Kenji Doya  
Tanja Mitrovic

Okinawa Institute of Science and Technology,  
Japan  
University of Canterbury, New Zealand

## Keynote Chairs

Zeng-Guang Hou  
Amir H. Gandomi  
Stephen Thorpe  
Leanne Bint  
Jing Ma  
Mahsa Mohaghegh  
Wei Qi Yan  
Amir Hussain  
Roopak Sinha

Chinese Academy of Sciences, China  
University of Technology Sydney, Australia  
Auckland University of Technology, New Zealand  
Edinburgh Napier University, UK  
Deakin University, Australia

## Publicity Chairs

El-Sayed M. El-Alfy  
David Brown  
Siddhartha Bhattacharyya

King Fahd University of Petroleum and Minerals,  
Saudi Arabia  
Nottingham Trent University, UK  
VSB Technical University of Ostrava, Czech  
Republic

Hyeyoung Park	Kyungpook National University, South Korea
Qinmin Yang	Zhejiang University, China
Chu Kiong Loo	University of Malaya, Malaysia
Cosimo Ieracitano	Mediterranean University of Reggio Calabria, Italy
Bernhard Pfahringer	University of Waikato, New Zealand

## Local Organising Chairs

Tek Tjing Lie	Auckland University of Technology, New Zealand
Felix Tan	Auckland University of Technology, New Zealand
Minh Nguyen	Auckland University of Technology, New Zealand
Yvonne Chan Cashmore	Auckland University of Technology, New Zealand
Peter Chong	Auckland University of Technology, New Zealand
Reza Enayatollahi	Toi Ohomai Institute of Technology, New Zealand
Selina Nihalani-Sharma	Auckland University of Technology, New Zealand
Janie van Woerden	Auckland University of Technology, New Zealand

## International Advisory Committee

Giancarlo Fortino	University of Calabria, Italy
Hamido Fujita	Iwate Prefectural University, Japan
Shariful Islam	Deakin University, Australia
Tianzi Jiang	Institute of Automation, CAS, China
M. Shamim Kaiser	Jahangirnagar University, Bangladesh
Joarder Kamruzzaman	Federation University Australia, Australia
Francesco Carlo Morabito	University Mediterranea of Reggio Calabria, Italy
Stefano Panzeri	University Medical Center Hamburg-Eppendorf, Germany
Hanchuan Peng	SEU-Allen Institute for Brain & Intelligence, China
Nelishia Pillay	University of Pretoria, South Africa

## Technical Program Committee

Hamid Abbasi	University of Auckland, New Zealand
Abdullah Abdullah	Universiti Teknologi PETRONAS, Malaysia
Hazel Abraham	Auckland University of Technology, New Zealand
Sonali Agarwal	IIIT Allahabad, India

Nehal Ahmad	Indian Institute of Technology Indore, India
Saad Bin Ahmed	RPTU, Germany
Toshiaki Aida	Okayama University, Japan
Aisha Ajmal	Open Polytechnic, New Zealand
Mushir Akhtar	IIT Indore, India
Zaid Al-Tameemi	Toi Ohomai Institute of Technology, New Zealand
Shafiq Alam	Massey University, New Zealand
Wajahat Ali	Aligarh Muslim University, India
Jumabek Alikhanov	HumbleBeeAI, South Korea
Fares Alkhawaja	Concordia University, Canada
Nathan Allen	Auckland University of Technology, New Zealand
Fahad Alvi	University of York, UK
Goran Andonovski	University of Ljubljana, Slovenia
Yuki Aoki	Osaka University, Japan
Sabri Arik	Istanbul University, Turkey
Tetsuya Asai	Hokkaido University, Japan
Mohammad Asif	IIIT Allahabad, India
G. Avinash	National Institute of Occupational Health, India
Helena Bahrami	Auckland University of Technology, New Zealand
Muhammad Zeeshan Baig	Macquarie University, Australia
Tao Ban	National Institute of Information and Communications Technology, Japan
Sourasekhar Banerjee	Umeå University, Sweden
Qiming Bao	University of Auckland, New Zealand
Soubhagya Barpanda	VIT-AP University, India
Arindam Basu	University of Canterbury, New Zealand
Boris Bačić	Auckland University of Technology, New Zealand
Aymen Ben Said	University of Regina, Canada
Krishan Berwal	Military College of Telecommunication Engineering, India
M. Agni Catur Bhakti	Sampoerna University, Indonesia
Raghavendra Bhalerao	IITRAM, India
Siddhartha Bhattacharyya	RCC Institute of Information Technology, India
Dasari Bhulakshmi	VIT, India
Ying Bi	Victoria University of Wellington, New Zealand
Jacques Blanc-Talon	DGA TA, France
Christian Bohn	University of Wuppertal, Germany
Phil Bones	University of Canterbury, New Zealand
Yee Ling Boo	RMIT University, Australia
Larbi Boubchir	University of Paris 8, France
Sally Britnell	Auckland University of Technology, New Zealand
Chenyang Bu	Hefei University of Technology, China

Sugam Budhraja	Auckland University of Technology, New Zealand
Michael Bunn	Auckland University of Technology, New Zealand
Jérémie Cabessa	Panthéon-Assas University Paris II, France
Wanqiang Cai	Nanjing University of Finance & Economics, China
Weidong Cai	University of Sydney, Australia
Cristian S. Calude	University of Auckland, New Zealand
Yuan Cao	Beijing University of Posts and Telecommunications, China
Elisa Capecci	Auckland University of Technology, New Zealand
Debasrita Chakraborty	Indian Statistical Institute, Kolkata, India
Srinivasa Chakravarthy	IIT Madras, India
Stephan Chalup	University of Newcastle, Australia
Victor Chan	Tsinghua University, China
Ritesh Chandra	IIT Allahabad, India
Siripinyo Chantamunee	Walailak University, Thailand
Gu Chaochen	Shanghai Jiao Tong University, China
Nisha Chauhan	Indian Institute of Technology Roorkee, India
Zaineb Chelly Dagdia	Université Paris-Saclay, France
Bang Chen	Ningbo University, China
Dongjie Chen	University of Chinese Academy of Sciences, China
Jianyong Chen	Shenzhen University, China
Jinpeng Chen	Beijing University of Posts & Telecommunications, China
Kecheng Chen	University of Science and Technology of China, China
Ling Chen	Southwest University, China
Lyu Chen	Shandong Normal University, China
Wei-Neng Chen	South China University of Technology, China
Xiaoqing Chen	University of Chinese Academy of Sciences, China
Xinrui Chen	Tsinghua University, China
Zhewei Chen	Australian National University, Australia
Zhi Chen	University of Electronic Science and Technology of China, China
Long Cheng	Institute of Automation, CAS, China
Xingfu Cheng	Qufu Normal University, China
Ziqi Cheng	Chongqing University, China
Sung-Bae Cho	Yonsei University, South Korea
Chak Fong Chong	Macao Polytechnic University, China
Peter Han Joo Chong	Auckland University of Technology, New Zealand
Eashita Chowdhury	Indian Institute of Technology Kharagpur, India

Mamadou B. H. Cissoko	University of Strasbourg, France
Raphael Couturier	Marie and Louis Pasteur University, France
Stephen Cox	Residio, New Zealand
Brian Cusack	Auckland University of Technology, New Zealand
Jianhua Dai	Hunan Normal University, China
Shaoxiang Dang	Nagoya University, Japan
Anik Das	St. Francis Xavier University, Canada
Santosh Das	OmDayal Group of Institutions, India
Subham Das	Indian Institute of Technology Madras, India
Sudhansu Bala Das	NIT Rourkela, India
Marcílio De Souto	University of Orléans, France
Neda Deljavan	Institute for Advanced Studies in Basic Sciences, Iran
Jeremiah D. Deng	University of Otago, New Zealand
Nagaraj V. Dharwadkar	National Institute of Technology, Warangal, India
Kumar Dheenadayalan	Qualcomm, USA
Johannes Dimyadi	University of Auckland, New Zealand
Yuxin Ding	Harbin Institute of Technology, China
Maryam Doborjeh	Auckland University of Technology, New Zealand
Zohreh Doborjeh	University of Auckland, New Zealand
Bin Dong	Ricoh Software Research Center, China
Ninghua Dong	Beijing Jiaotong University, China
Ruiqi Dong	South China Normal University, China
Jordan Douglas	University of Auckland, New Zealand
Kenji Doya	OIST, Japan
Haiwen Du	Harbin Institute of Technology, China
Fuqing Duan	Beijing Normal University, China
Piotr Duda	Częstochowa University of Technology, Poland
Thao Duong	Murdoch University, Australia
Varun Dutt	Indian Institute of Technology Mandi, India
Pratik Dutta	Indian Institute of Technology Patna, India
Mansoor Ebrahim	Iqra University, Pakistan
El-Sayed M. El-Alfy	King Fahd University of Petroleum and Minerals, Saudi Arabia
Reza Enayatollahi	Toi Ohomai Institute of Technology, New Zealand
Farnoush Falahatraftar	Polytechnique Montréal, Canada
Chenyou Fan	South China Normal University, China
Jiangtao Fan	Durham University, UK
Junyuan Fang	City University of Hong Kong, China
Sen Fang	University of Victoria, Australia
Yifei Fang	University of Chinese Academy of Sciences, China

Zhiyuan Fang	University of Waikato, New Zealand
Muhammad Farhan	Glocal University, India
Jiaxuan Feng	Southwest University of Science and Technology, China
Yong Feng	Chongqing University, China
Jaco Fourie	Lincoln Agritech Ltd, UK
Frieyadie Frieyadie	STMIK Nusa Mandiri, Indonesia
Junjie Fu	Southeast University, China
Taoran Fu	Hunan University, China
Xiping Fu	University of Otago, New Zealand
Yulin Fu	University of Auckland, New Zealand
Fumiyo Fukumoto	University of Yamanashi, Japan
Haitao Gan	Hubei University of Technology, China
Mudasir Ganaie	Indian Institute of Technology Indore, India
Varun Ganjigunte Prakash	CogniAble, India
Qian Gao	Qilu University of Technology, China
Terry Gao	Moreton Bay City Council, Australia
Xinyi Gao	Auckland University of Technology, New Zealand
Xizhan Gao	University of Jinan, China
Chandan Gautam	I2R, A*STAR, Singapore
Liang Ge	Chongqing University, China
Mengmeng Ge	University of Canterbury, New Zealand
Trevor Gee	University of Auckland, New Zealand
Mingyang Geng	National University of Defense Technology, China
Thanawit Gerdprasert	Yamaguchi University, Japan
Akbar Ghobakhloo	Auckland University of Technology, New Zealand
Hamid Gholamhosseini	Auckland University of Technology, New Zealand
Ashish Ghosh	Indian Statistical Institute, Kolkata, India
Tripti Goel	National Institute of Technology Silchar, India
Zbigniew Gomolka	University of Rzeszow, Poland
Jiaying Gong	Virginia Tech, USA
Rui Gong	Chinese Academy of Sciences, China
Yingjie Gong	Zhejiang University, China
Maanvik Gounder	University of the South Pacific, Fiji
Richard Green	University of Canterbury, New Zealand
Jessica Gu	University of Auckland, New Zealand
Tamil Selvan Gunasekaran	University of Auckland, New Zealand
Long Guo	Sichuan University, China
Ping Guo	Beijing Normal University, China
Qin Guo	Peking University, China
Deepak Gupta	MNNIT Allahabad, India

Harshit Gupta	IIIT Allahabad, India
Tomasz Hachaj	Pedagogical University of Krakow, Poland
Katsuyuki Hagiwara	Mie University, Japan
Masafumi Hagiwara	Keio University, Japan
Luke Hallum	University of Auckland, New Zealand
Chansu Han	National Institute of Information and Communications Technology, Japan
Gao Han	Xidian University, China
Zhang Haoyu	Chinese Academy of Sciences, China
Md. Tarek Hasan	United International University, Bangladesh
Tatsuhiro Hasegawa	University of Fukui, Japan
Hao He	Chinese Academy of Sciences, China
Xing He	Southwest University, China
Xinyang He	UCAS, China
Yangliu He	Beijing University of Posts and Telecommunications, China
Yuting He	University of Nottingham Ningbo, China
Mahshid Helali Moghadam	RISE Research Institutes of Sweden, Sweden
Hugo Hernault	Playtika, Japan
Akira Hirose	University of Tokyo, Japan
Yvonne Hong	Victoria University of Wellington, New Zealand
Yin Hongwei	Huzhou University, China
Adrian Horzyk	University of Krakow, Poland
Amanda Horzyk	University of Edinburgh, UK
Md Zakir Hossain	Australian National University, Australia
Zengguang Hou	Chinese Academy of Sciences, China
Menghao Hu	Pengcheng Laboratory, China
Yan Hu	University of New South Wales, Australia
Zhongyun Hua	Harbin Institute of Technology, China
He Huang	Soochow University, China
Kaizhu Huang	Duke Kunshan University, China
Sheng Huang	Chongqing University, China
Siyu Huang	Nanjing University of Aeronautics and Astronautics, China
Zhen Huang	Shenyang Institute of Computing Technology, China
Carolynne Hultquist	Pennsylvania State University, USA
Aarij Hussaan	Iqra University, Pakistan
David Iclanzan	Sapientia University, Romania
Cosimo Ieracitano	University “Mediterranea” of Reggio Calabria, Italy
Kazushi Ikeda	Nara Institute of Science and Technology, Japan

Radu Tudor Ionescu	University of Bucharest, Romania
Noriyuki Iwane	Hiroshima City University, Japan
Sana Jabbar	Lahore University of Management Sciences, Pakistan
Sapna Jaidka	University of Waikato, New Zealand
Debesh Jha	Northwestern University, USA
Ningning Jia	Beijing University of Posts and Telecommunications, China
Yan Jia	National University of Defense Technology, China
Peng Jiang	Wuhan University, China
Xianwei Jiang	Southwest University of Science and Technology, China
Xuesong Jiang	Qilu University of Technology, China
Chen Jiaqi	Dongguan University of Technology, China
Jin	Xi'an Jiaotong Liverpool University, China
Chunzhen Jin	Northeastern University, USA
Xiaozheng Jin	Qilu University of Technology, China
Vijay John	RIKEN, Japan
Kenneth Johnson	Auckland University of Technology, New Zealand
Seul Jung	Chungnam National University, China
Akbar K.	BITS Pilani, Goa Campus, India
Sweta Kaman	IIT Jodhpur, India
Keiji Kamei	Nishinippon Institute of Technology, Japan
Keisuke Kameyama	University of Tsukuba, Japan
Yoshimi Kamiyama	Aichi Prefectural University, Japan
Joarder Kamruzzaman	Federation University Australia, Australia
Bhavik Kanekar	IIT Mandi, India
Tomoyuki Kaneko	University of Tokyo, Japan
Jun-Su Kang	Kyungpook National University, South Korea
Shin'Ichiro Kanoh	Shibaura Institute of Technology, Japan
Nikola Kasabov	Auckland University of Technology, New Zealand
Arshpreet Kaur	NIT Jalandhar, India
Yoshinobu Kawahara	Osaka University/RIKEN, Japan
Hideaki Kawano	Kyushu Institute of Technology, Japan
Kostiantyn Khabarlak	Dnipro University of Technology, Ukraine
Mehshan Khan	Deakin University, Australia
Atikant Khanna	Auckland University of Technology, New Zealand
Daegyeom Kim	Korea University, South Korea
Jonghong Kim	Kyungpook National University, South Korea
Mutsumi Kimura	Ryukoku University, Japan
Irwin King	Chinese University of Hong Kong, China

Gisela Klette	Auckland University of Technology, New Zealand
Mallika Kliangkhlao	Walailak University, Thailand
Alistair Knott	Victoria University of Wellington, New Zealand
Kunikazu Kobayashi	Aichi Prefectural University, Japan
Rangachary Kommanduri	Indian Institute of Information Technology, Sri City, India
Aneesh Krishna	Curtin University, Australia
Rita Krishnamurthi	Auckland University of Technology, New Zealand
Adam Krzyzak	Concordia University, Canada
Sumant Kulkarni	Zenlabs, Zensar Technologies, India
Estine Kumar	University of the South Pacific, Fiji
Neetesh Kumar	IITR, India
Praveen Kumar	Banaras Hindu University, India
Rakesh Kumar	IIT BHU, India
Swaroop Kumar	L&T Technology Services, India
Chithrangi Kumarasinghe	University of Moratuwa, Sri Lanka
Anuradha Kumari	Indian Institute of Technology Indore, India
Naga Jyothi Kunchala	Massey University, New Zealand
Souraja Kundu	Indian Institute of Technology Guwahati, India
Hiroki Kurashige	Tokai University, Japan
Tomoki Kurikawa	Future University Hakodate, Japan
Kurnianingsih	Politeknik Negeri Semarang, Indonesia
Fatih Kurugollu	University of Sharjah, UAE
Thomas Lacombe	University of Auckland, New Zealand
Hamid Laga	Murdoch University, Australia
Edmund Lai	Auckland University of Technology, New Zealand
Leon Lange	University of California, San Diego, USA
Walter Langelaar	Victoria University of Wellington, New Zealand
Sang Hun Lee	Kookmin University, South Korea
Tet Chuan Lee	Auckland University of Technology, New Zealand
Yuan Lei	University of Chinese Academy of Sciences, China
Chi Sing Leung	City University of Hong Kong, China
Bingxian Li	Heilongjiang University, China
Bo Li	Baidu Inc, China
Fengyu Li	Beijing University of Posts and Telecommunication, China
Hongfei Li	Xinjiang University, China
Jiale Li	University of Auckland, New Zealand
Jun Li	Nanjing Normal University, China
Lining Li	CASIA, China
Maodong Li	Soochow University, China

Mengmeng Li	Zhengzhou University, China
Mengshu Li	University of Toronto, Canada
Mengting Li	Beijing University of Posts and Telecommunications, China
Ming Li	Wuhan University of Technology, China
Peifeng Li	Soochow University, China
Ruifan Li	Beijing University of Posts and Telecommunications, China
Sirui Li	Murdoch University, China
Tieshan Li	Dalian Maritime University, China
Weiwei Li	UESTC, China
Xiaohong Li	Northwest Normal University, China
Yantao Li	Chongqing University, China
Yi Li	Lancaster University, UK
Yinbao Li	Baidu Tech., China
Yun Li	Nanjing University of Posts and Telecommunications, China
Ziwei Li	UCAS, China
Jiawen Liang	City University of Hong Kong, China
Xiaokun Liang	Shenzhen Institute of Advanced Technology, China
Xiaoyi Liang	Tongji University, China
Xu Liang	Harbin Institute of Technology, China
Zhuonan Liang	University of Sydney, Australia
Junfeng Liao	Shanghai University of International Business and Economics, China
Xiao-Cheng Liao	South China University of Technology, China
Jianpeng Lin	Guangdong University of Technology, China
Qiuhua Lin	Dalian University of Technology, China
Yang Lin	University of Sydney, Australia
Baodi Liu	China University of Petroleum, China
Binghao Liu	Beihang University, China
Gangli Liu	Tsinghua University, China
Hanyuan Liu	City University of Hong Kong, China
Jian-Wei Liu	China University of Petroleum, China
Juan Liu	Wuhan University, China
Lei Liu	Northeastern University, USA
Renyang Liu	National University of Singapore, Singapore
Shang Liu	UCAS, China
Weifeng Liu	China University of Petroleum, China
Wen Liu	Chinese University of Hong Kong, China
Xiaoyang Liu	Huazhong University Science & Technology, China

Yang Liu	Fudan University, China
Yanming Liu	Georgia Institute of Technology, USA
Yaozhong Liu	Australian National University, Australia
Zhaoyi Liu	KU Leuven, Belgium
Zhe Liu	Jiangsu University, China
Han Long	National University of Defense Technology, China
Jieting Long	University of Sydney, Australia
Chu Kiong Loo	University of Malaya, Malaysia
Andrew Lowe	Auckland University of Technology, New Zealand
Gewei Lu	Shanghai Jiao Tong University, China
Heng-Yang Lu	Nanjing University, China
Hongtao Lu	Shanghai Jiao Tong University, China
Weihai Lu	Peking University, China
Wenhao Lu	Nanyang Technological University, Singapore
Yu Lu	Shenzhen Technology University, China
Shijie Luan	Chinese Academy of Sciences, China
Fangzhou Luo	McMaster University, Canada
Róisín Luo	University of Galway, Ireland
Siwen Luo	University of Western Australia, Australia
Yuchuan Luo	National University of Defense Technology, China
Raymond Lutui	Auckland University of Technology, New Zealand
Jiancheng Lv	Sichuan University, China
Yuezu Lv	Beijing Institute of Technology, China
Liangfu Lyu	Federation University Australia, Australia
Qingguo Lü	Southwest University, China
Jing Ma	Auckland University of Technology, New Zealand
Jinwen Ma	Peking University, China
Ming Ma	Yeshiva University, USA
Yan Ma	Fudan University, China
Yuqi Ma	Chinese University of Hong Kong, China
Alexei Machado	Pontifical Catholic University of Minas Gerais, Brazil
Jyoti Maggu	Thapar Institute of Engineering and Technology, India
Tariq Mahmood	COMSATS Institute of Information Technology, Pakistan
Mufti Mahmud	King Fahd University of Petroleum and Minerals, Saudi Arabia
Snehashis Majhi	Inria Sophia Antipolis, France
Mishaim Malik	University of Auckland, New Zealand

Mamta	IIT Patna, India
Raquel Marasigan	University of Asia and the Pacific, Philippines
Gerard M. Freixas	Fudan University, China
Stefan Marks	Auckland University of Technology, New Zealand
Archana Mathur	Nitte Meenakshi Institute of Technology, India
Yoshitatsu Matsuda	Seikei University, Japan
Tomas Maul	University of Nottingham Malaysia Campus, Malaysia
Jacek Mańdziuk	Warsaw University of Technology, Poland
Yogendra Meena	Delhi University, India
Yi Mei	Victoria University of Wellington, New Zealand
Erik Meijering	University of New South Wales, Australia
Qing-Xin Meng	China University of Petroleum, China
Alexander Merkin	AUT, New Zealand
Cheng Miao	Guangxi Normal University, China
Ashish Mishra	University of Nevada Las Vegas, USA
Sajib Mistry	Curtin University, Australia
Tanja Mitrovic	University of Canterbury, New Zealand
Seiji Miyoshi	Kansai University, Japan
Mohammadreza Montazerijouybari	Polytechnique Montréal, Canada
Francesco C. Morabito	University of Reggio Calabria, Italy
Satoru Morita	Yamaguchi University, Japan
Mpatisi Moyo	AiTonomy, UK
Taslim Murad	Georgia State University, USA
Shingo Murakami	Chuo University, Japan
Dharmalingam Muthusamy	Bharathiar University, India
Chitrakala Muthuveerappan	Victoria University of Wellington, New Zealand
Muhan Na	Inner Mongolia University, China
Isao Nambu	Nagaoka University of Technology, Japan
Parma Nand	Auckland University of Technology, New Zealand
Ajit Narayanan	Auckland University of Technology, New Zealand
Gokulmuthu Narayanaswamy	IIT Kharagpur, India
Kiyohisa Natsume	Kyushu Institute of Technology, Japan
Azadeh Nazemi	Norwood Systems, Australia
Usman Nazir	Lahore University of Management Sciences, Pakistan
Elena Nechita	Vasile Alecsandri University of Bacau, Romania
Chandra Mohan Singh Negi	Siemens, India
Mehdi Neshat	University of Adelaide, Australia
Kourosh Neshatian	University of Canterbury, New Zealand
Frank Neumann	University of Adelaide, Australia

Bach Nguyen	Victoria University of Wellington, New Zealand
Bao Sinh Nguyen	HUST, Vietnam
Binh P. Nguyen	Victoria University of Wellington, New Zealand
Minh Nguyen	Auckland University of Technology, New Zealand
Quang Vinh Nguyen	Western Sydney University, Australia
Mukku Nisanth Kartheek	National Institute of Technology Warangal, India
Sou Nobukawa	Chiba Institute of Technology, Japan
Anupiya Nugaliyadde	Murdoch University, Australia
Jeongbin Ok	Victoria University of Wellington, New Zealand
Diego Oliva	Universidad de Guadalajara, Spain
Toshiaki Omori	Kobe University, Japan
Sihem Omri	Higher School of Communication of Tunis, Tunisia
Hideaki Orii	Fukuoka University, Japan
Seiichi Ozawa	Kobe University, Japan
Achmad Pahlevi	Auckland University of Technology, New Zealand
Susmita Palmal	Ramgarh Engineering College, India
Guangyuan Pan	Linyi University, China
Wenkai Pan	Linyi University, China
Pankaj Pandey	Indian Institute of Technology, Delhi, India
Jason Pang	Lincoln University, UK
Paresh Kumar Panigrahi	VIT-AP University, India
Dipendra Pant	Norwegian University of Science and Technology, Norway
Hyeyoung Park	Kyungpook National University, South Korea
Dipanjyoti Paul	Indian Institute of Technology Patna, India
Mangor Pedersen	Auckland University of Technology, New Zealand
Siamak Pedrammehr	Deakin University, Australia
Anjie Peng	Southwest University of Science and Technology, China
Nasca Peng	Statistics New Zealand, New Zealand
Yong Peng	Hangzhou Dianzi University, China
Joao Pereira	Imperial College London, UK
Krassie Petrova	Auckland University of Technology, New Zealand
Somnuk Phon-Amnuaisuk	Universiti Teknologi Brunei, Brunei Darussalam
Pasu Poonpakdee	Walailak University, Thailand
Vijay Prakash	Thapar University, India
Narinder Singh Punn	Mayo Clinic, Arizona, USA
Junjie Qi	Guangxi Normal University, China
Weihua Qiang	Tianjin University, China
Yu Qiao	Shanghai Jiao Tong University, China
Sitian Qin	Harbin Institute of Technology at Weihai, China

Xiaoyang Qu	Huazhong University of Science and Technology, China
Abdul Quadir	IIT Indore, India
Uday Kiran Rage	University of Aizu, Japan
Krishna Raghuwaiya	University of the South Pacific, Fiji
Ibrahim Rahman	Open Polytechnic of New Zealand, New Zealand
Jessica Rahman	University of Dhaka, Bangladesh
Shri Rai	Murdoch University, Australia
Surbhi Raj	Indian Institute of Technology Patna, India
K. Ramakrishnan	Auckland University of Technology, New Zealand
R. Kanesaraj Ramasamy	Multimedia University, Malaysia
Deepak Ranjan Nayak	Malaviya National Institute of Technology, Australia
Rabia Naseer Rao	University of Auckland, New Zealand
Munish Rathee	Auckland University of Technology, New Zealand
Ramesh Rayudu	Victoria University of Wellington, New Zealand
Khalid Raza	Jamia Millia Islamia, India
Jianfeng Ren	University of Nottingham Ningbo, China
Minsi Ren	Chinese Academy of Sciences, China
Shao Renrong	East China Normal University, China
Tobias Rettenmeier	University of Applied Sciences Heilbronn, Germany
Young Ju Rho	Tech University of Korea, South Korea
Daniel Riccio	University of Naples Federico II, Italy
Rishabh	University of Delhi, India
Horacio Gonzalez	Universidad de Guanajuato, Mexico
Gargi Roy	Brunel University London, UK
Kaushik Roy	West Bengal State University, India
Muhammad Fakhru Rozi	National Institute of Information and Communications Technology, Japan
Ji Ruan	Auckland University of Technology, New Zealand
Khairun Saddami	Universitas Syiah Kuala, Indonesia
Michał Sadowski	Jagiellonian University, Poland
Amit Kumar Sah	South Asian University, India
Pranab Sahoo	Indian Institute of Technology Patna, India
Naveen Saini	Indian Institute of Information Technology Allahabad, India
Ken Saito	Nihon University, Japan
Toshimichi Saito	Hosei University, Japan
Md Sajid	IIT Indore, India
Ko Sakai	University of Tsukuba, Japan
Nazmus Sakib	Ahsanullah University of Science & Technology, Bangladesh

Rohit Salgotra	AGH University of Krakow, Poland
Michel Salomon	IUT Belfort-Montbéliard, France
Toshikazu Samura	Yamaguchi University, Japan
Xue Sang	Northwestern Polytechnical University, China
Takashi Sano	Tokyo University, Japan
Yassine Saoudi	Faculté des Sciences de Tunis, Tunisia
Naoyuki Sato	Future University Hakodate, Japan
Eri Sato-Shimokawara	Tokyo Metropolitan University, Japan
Seiya Satoh	Tokyo Institute of Technology, Japan
Wojciech Sałabun	West Pomeranian University of Technology, Poland
Erich Schikuta	University of Vienna, Austria
Eric Scott	MITRE Corporation, USA
Mahdi Setayesh	Microsoft Inc., USA
Noushath Shaffi	Sultan Qaboos University, Oman
Nida Shahab	Auckland University of Technology, New Zealand
Reza Shahamiri	University of Auckland, New Zealand
Jiaxing Shang	Chongqing University, China
Peng Shao	Jiangxi Agricultural University, China
Zhenzhou Shao	Capital Normal University, China
Sourabh Sharma	Avantika University, India
Megha Sharma	Indian Institute of Technology Mandi, India
Swakkhar Shatabda	BRAC University, Bangladesh
Hualei Shen	Henan Normal University, China
Zhixiang Shen	UESTC, China
Yin Sheng	Huazhong University of Science and Technology, China
Yongpan Sheng	Southwest University, China
Pumeng Shi	University of British Columbia, Canada
Qiushi Shi	NTU, Singapore
Xinxin Shi	Changchun University of Science and Technology, China
Hiroki Shibata	Tokyo Metropolitan University, Japan
Hayaru Shouno	University of Electro-Communications, Japan
Jiang Shuyu	Sichuan University, India
Shijing Si	Duke University, USA
Jiten Sidhpura	Sardar Patel Institute of Technology, India
Rangika Silva	Queensland University of Technology, Australia
Simeon Simoff	Western Sydney University, Australia
David Simpson	Massey University, New Zealand
Balkaran Singh	Auckland University of Technology, New Zealand
Om Singh	National Institute of Technology Patna, India

Shashank Singh	Kanpur Institute of Technology, India
Roopak Sinha	Deakin University, Australia
Soumen Sinha	Mahindra University, India
Stephen Skalicky	Victoria University of Wellington, New Zealand
Ferdous Sohel	Murdoch University, Australia
Upika Somaratne	Murdoch University, Australia
Aiguo Song	Southeast University, China
Chao Song	Zhejiang Gongshang University, China
Haohao Song	Xiamen University, China
Liang Song	Fudan University, China
Xianfeng Song	South China University of Technology, China
Laxmi Soni	AKS University, India
Paul Sowman	Auckland University of Technology, New Zealand
Aleksei Staroverov	MIPT, Russia
Amy Stewart	University of Waikato, New Zealand
Martin Stommel	Auckland University of Technology, New Zealand
Jianbo Su	Shanghai Jiao Tong University, China
Xinyan Su	University of Chinese Academy of Sciences, China
Yila Su	Inner Mongolia University of Technology, China
Zhixun Su	Dalian University of Technology, China
Badri Narayan Subudhi	Indian Institute of Technology Jammu, India
Toshiharu Sugawara	Waseda University, Japan
John Sum	National Chung Hsing University, China
Alexander Sumich	Nottingham Trent University, UK
Hao Sun	Nankai University, China
Shiwen Sun	Inner Mongolia University, China
Shuo Sun	Inner Mongolia University, China
Tao Sun	Beihang University, China
Zhanquan Sun	University of Shanghai for Science and Technology, China
Suryavardan Suresh	New York University, USA
Kanata Suzuki	Fujitsu Limited, Japan
Satoshi Suzuki	NTT Yokosuka, Japan
Yoshimi Suzuki	University of Yamanashi, Japan
Mikołaj Ślupiński	University of Wrocław, Poland
Izak Tait	Auckland University of Technology, New Zealand
Murtaza Taj	Lahore University of Management Sciences, Pakistan
Norikazu Takahashi	Okayama University, Japan
Hiroshige Takeichi	RIKEN, Japan
Takashi Takekawa	Kogakuin University, Japan

Hiroshi Tamura	Osaka University, Japan
Christine Nya-Ling Tan	Massey University, New Zealand
Chunyu Tan	Anhui University, China
Renzo Roel Tan	Nara Institute of Science and Technology, Japan
Ying Tan	Peking University, China
Zhengguang Tan	Guangdong University of Technology, China
Takuma Tanaka	Shiga University, China
Fengxiao Tang	Tohoku University, Japan
Haoran Tang	Beijing University of Posts and Telecommunications, China
Maolin Tang	Queensland University of Technology, Australia
Yang Tang	East China University of Science and Technology, China
Yihang Tang	Chongqing University of Posts and Telecommunications, China
Zerui Tang	Xiamen University, China
M. Tanveer	Indian Institute of Technology, Indore, India
Zerui Tao	Tokyo University of Agriculture and Technology, China
Jules-Raymond Tapamo	University of KwaZulu-Natal, South Africa
Prithvi Tarale	University of Massachusetts Amherst, USA
Shuhei Tarashima	NTT Communications Corporation, Japan
Yassin Terraf	Mohammed VI Polytechnic University, Morocco
Putthiporn Thanathamathee	Walailak University, Thailand
Veerakumar Thangaraj	National Institute of Technology Goa, India
Chuan Tian	University of Canterbury, New Zealand
Hao Tian	Zhejiang University of Technology, China
Aruna Tiwari	IIT Indore, India
Sadhana Tiwari	Galgotias College of Engineering & Technology, India
Ewaryst Tkacz	Silesian University of Technology, Poland
Kar-Ann Toh	Yonsei University, South Korea
Farank Tohidi	Charles Sturt University, Australia
Cong Tran	QUOC, Vietnam
Chidentree Treesatayapun	Walailak University, Thailand
Richa Tripathi	Washington University in St. Louis, USA
Enmei Tu	Shanghai Jiao Tong University, China
Nitin Tyagi	IIT Roorkee, India
Hamid Usefi	Memorial University of Newfoundland, Canada
Alireza Valizadeh	Universitat de les Illes Balears, Spain
Manju Vallayil	Auckland University of Technology, New Zealand
Maryam Var Naseri	Victoria University of Wellington, New Zealand

Matthieu Vignes	Massey University, New Zealand
Nobuhiko Wagatsuma	Toho University, Japan
Shanchuan Wan	University of Tokyo, Japan
Tao Wan	Beihang University, China
Wang	City University of Hong Kong, China
Bin Wang	Nanjing University of Finance & Economics, China
Binqiang Wang	Chinese Academy of Sciences, China
Chao Wang	China Academy of Railway Sciences Corporation Limited, China
Chen Wang	Chinese Academy of Sciences, China
Chengliang Wang	Chongqing University, China
Chunshi Wang	Guilin University of Electronic Technology, China
Guanjin Wang	Murdoch University, Australia
Haizhou Wang	Sichuan University, China
Han Wang	Xidian University, China
Hao Wang	Shenzhen University, China
Haowen Wang	Alipay, Ant Group, China
Jiale Wang	Zhejiang Sci-Tech University, China
Jianzong Wang	Ping An Technology Co., Ltd., China
Jun-Wei Wang	University of Science and Technology Beijing, China
Kaier Wang	Volpara Health Technologies Ltd., New Zealand
Liang Wang	Beijing University of Technology, China
Liang Wang	Huazhong University of Science and Technology, China
Liantao Wang	Hohai University, China
Ming Hui Wang	China University of Petroleum, China
Nana Wang	Jiangsu Normal University, China
Peijun Wang	Anhui Normal University, China
Rui Wang	Ningbo University, China
Ruiying Wang	Southwest University of Science and Technology, China
Xianzhi Wang	University of Technology Sydney, Australia
Xiao Wang	Chongqing Normal University, China
Xiulin Wang	Dalian University of Technology, China
Yadi Wang	Henan University, China
Ye Wang	National University of Defense Technology, China
Yong Wang	Southeast University, China
Yongyu Wang	JD Logistics, China
Zhenni Wang	City University of Hong Kong, China

Zhongsheng Wang	University of Auckland, New Zealand
Zi-Peng Wang	University of Jinan, New Zealand
Ziwei Wang	Huazhong University of Science and Technology, China
Yoshikazu Washizawa	University of Electro-Communications, China
Fengchen Wei	University of Sussex, UK
Hongxi Wei	Inner Mongolia University, China
Alastair Wells	Auckland University of Technology, New Zealand
Guanghui Wen	RMIT University, Australia
Junjie Wen	Chinese University of Hong Kong, China
Jinta Weng	Chinese Academy of Sciences, China
Lei Wenjie	City University of Hong Kong, China
David White	Auckland University of Technology, New Zealand
Tom White	Victoria University of Wellington, New Zealand
Savindi Wijenayaka	University of Auckland, New Zealand
Michael Witbrock	University of Auckland, New Zealand
Hiutung Wong	City University of Hong Kong, China
Ka-Chun Wong	City University of Hong Kong, China
Kevin Wong	Murdoch University, Australia
Boqi Wu	University of Wuppertal, Germany
Chao Wu	Zhejiang University, China
Chengkun Wu	National University of Defense Technology, China
Haha Wu	Xinjiang University, China
Song Wu	Hainan Tropical Ocean University, China
Xianze Wu	Shanghai Jiao Tong University, China
Zipeng Wu	University of Birmingham, UK
Zhiqiu Xia	Rutgers University, USA
Ziwei Xiang	Chinese Academy of Sciences, China
Jinying Xiao	Changsha University of Science & Technology, China
Qiang Xiao	Huazhong University of Science and Technology, China
Xi Xiao	University of Alabama at Birmingham, USA
Shiwen Xie	Central South University, China
Zaipeng Xie	Hohai University, China
Yucheng Xing	Stony Brook University, USA
Wang Xinshen	Guangdong University of Foreign Studies, China
Wenxin Xiong	City University of Hong Kong, China
Chao Xu	Changsha University of Science and Technology, China
Fanchao Xu	University of Science and Technology of China, China

Jianhua Xu	Nanjing Normal University, China
Jinhua Xu	East China Normal University, China
Lele Xu	Southeast University, China
Qing Xu	Tianjin University, China
Xinyue Xu	Hong Kong University of Science and Technology, China
Junyu Xuan	University of Technology Sydney, Australia
Felix Yan	Victoria University of Wellington, New Zealand
Shankai Yan	Hainan University, China
Teng Yan	Shenzhen University, China
Weiqi Yan	AUT, New Zealand
Da Yang	Beijing University of Aeronautics and Astronautics, China
Haitian Yang	Chinese Academy of Sciences, China
Jie Yang	Shanghai Jiao Tong University, China
Mengyu Yang	Beijing University of Posts and Telecommunications, China
Minghao Yang	Chinese Academy of Sciences, China
Peipei Yang	Chinese Academy of Science, China
Peng Yang	Chongqing Three Gorges University, China
Qinmin Yang	Zhejiang University, China
Wei Yang	University of Chinese Academy of Sciences, China
Yanfeng Yang	South China University of Technology, China
Zhan Yang	Central South University, China
Zhikai Yang	KTH Royal Institute of Technology, Sweden
Wangshu Yao	Soochow University, China
Yifeng Yao	University of South China, China
Yun Ye	Intel, USA
Wang Yingying	Shenyang Aerospace University, China
Yongmin Yoo	Macquarie University, Australia
Shuyuan You	Tianjin University, China
Dianzhi Yu	Chinese University of Hong Kong, China
Na Yu	Zhejiang University, China
Ping Yu	Nanjing University of Science and Technology, China
Wenxin Yu	Southwest University of Science and Technology, China
Zhibin Yu	Ocean University of China, China
Zhiwen Yu	South China University of Technology, China
Fangfang Yuan	Chinese Academy of Sciences, China
Xin Yuan	Southeast University, China

Dmitry Yudin	Moscow Institute of Physics and Technology, Russia
Chao Yue	University of Chinese Academy of Sciences, China
Xiaodong Yue	Shanghai University, China
Farzana Zahid	University of Waikato, New Zealand
Ruijie Zeng	Chinese Academy of Sciences, China
Weixin Zeng	National University of Defense Technology, China
Xinhua Zeng	Fudan University, China
Zekeng Zeng	Chinese Academy of Sciences, China
Ewa Zeslawska	University of Rzeszow, Poland
Zhiyuan Zha	Renmin University of China, China
Chao Zhang	Shanxi University, China
Chao Zhang	South China Normal University, China
Chenyi Zhang	University of Canterbury, New Zealand
Chong Zhang	Xi'an Jiaotong-Liverpool University, China
Chufan Zhang	Shanghai Jiao Tong University, China
Congwei Zhang	Southeast University, China
Fan Zhang	Shandong Technology and Business University, China
Francis X. Zhang	Durham University, UK
Gaoyan Zhang	Tianjin University, China
Han Zhang	Northwest University, China
Haoyang Zhang	Chongqing University of Posts and Telecommunications, China
Hongtao Zhang	Kochi University of Technology, Japan
Jiahui Zhang	Beijing University of Technology, China
Jinchuan Zhang	University of Electronic Science and Technology of China, China
Kai Zhang	East China Normal University, China
Kang Zhang	Kyushu University, Japan
Li Zhang	Soochow University, China
Qian Zhang	Jiangsu Open University, China
Ruixiao Zhang	University of Southampton, UK
Ting Zhang	Central China Normal University, China
Tong Zhang	China Mobile Research Institute, China
Weili Zhang	Xi'an Jiaotong University, China
Weilun Zhang	Tianjin University, China
Wendy Zhang	University of Canterbury, New Zealand
Xiaowei Zhang	Qingdao University, China
Xu Zhang	Jiangsu Normal University, China

Xulong Zhang	Ping An Technology (Shenzhen) Co., Ltd., China
Xunhui Zhang	National University of Defense Technology, China
Yang Zhang	City University of Hong Kong, China
Yangsong Zhang	Southwest University of Science and Technology, China
Zijing Zhang	University of Waikato, New Zealand
Bo Zhao	Beijing Normal University, China
Hui Zhao	University of Jinan, China
Jianhui Zhao	Wuhan University, China
Jigui Zhao	Xinjiang University, China
Jing Zhao	Qilu University of Technology, China
Keer Zhao	Zhejiang University of Technology, China
Ming Zhao	Central South University, China
Rui Zhao	University of Technology Sydney, Australia
Runjie Zhao	Zhejiang University, China
Xujian Zhao	Southwest University of Science and Technology, China
Yan Zheng	Kyushu University, Japan
Yingtao Zheng	University of Auckland, New Zealand
Yuchen Zheng	Shihezi University, China
Guoqiang Zhong	Ocean University of China, China
Jinghui Zhong	South China University of Technology, China
Ping Zhong	Central South University, China
Guangchong Zhou	Chinese Academy of Science, China
Jinjia Zhou	Hosei University, China
Shihua Zhou	Dalian University, China
Xiao-Hu Zhou	Chinese Academy of Sciences, China
Xinyu Zhou	Jiangxi Normal University, China
Yu Zhou	National University of Defense Technology, China
Zhenxiong Zhou	National University of Defense Technology, China
Leyi Zhu	University of Macau, China
Qiaoming Zhu	Soochow University, China
Qiyuan Zhu	Swinburne University of Technology, Australia
Xuanying Zhu	Australian National University, Australia
Yuesheng Zhu	Peking University, China
Wang Ziling	Sichuan University, China
Yuan Zong	Southeast University, China
Xu Zou	Zhejiang University, China

# Contents – Part I

Defend from Scratch: A Diffusion-Based Proactive Defense Method for Unauthorized Speech Synthesis .....	1
<i>Ying Wang, Yuchuan Luo, Zhenyu Qiu, Lin Liu, and Shaojing Fu</i>	
Transformers as Approximations of Solomonoff Induction .....	16
<i>Nathan Young and Michael Witbrock</i>	
Interpreting Decision Transformer: Insights from Continuous Control Tasks ...	26
<i>Dhillu Thambi, Praveen Paruchuri, and Perusha Moodley</i>	
Flexible-Order Feature-Interaction for Mixed Continuous and Discrete Variables with Group-Level Interpretability .....	42
<i>Zijie Zhai, Junchen Shen, Ping Li, Jie Zhang, and Kai Zhang</i>	
Critical Feature Sifting and Dynamic Aggregation for Anomalous Audio Sequence Detection .....	58
<i>Erteng Liu, Kewei Gao, Xing Zhou, Sen Lin, Jianhai Chen, Yijun Bei, and Zunlei Feng</i>	
Parallel Interpretation Network via Semantic Visual Probe and Counterfactual Verification .....	73
<i>Hao Wan, Keyang Cheng, and Hao Zhou</i>	
Real-Time Decentralized M2M Decision-Making via Deep Learning and Incremental Learning .....	88
<i>Mohamed Dwedar, Fatih Bayram, and Alexander Jesser</i>	
Explainable Federated Stacking Models with Encrypted Gradients for Secure Kidney Medical Imaging Diagnosis .....	103
<i>Sharia Arfin Tanim, Al Rafi Aurnob, Md Rokon Islam, Md Saef Ullah Miah, M. Mostafizur Rahman, and Mufti Mahmud</i>	
DDFGNN: Dual-Dimensionality Fusion Graph Neural Network for Social Bot Detection .....	119
<i>Yuze Bai, Yingjie Sun, Chengping Zheng, and Yizhou Li</i>	
A Motif-Based Graph Convolution Network for Stock Trend Prediction .....	134
<i>Lei Zhou, Yuqi Zhang, Nancy Wang, Jian Yu, Guiling Wang, and Xin Zheng</i>	

VAGNN: Advancing the Generalization of Graph Neural Networks . . . . .	150
<i>Shuming Liang, Yu Ding, Bin Liang, Zhidong Li, Siqi Zhang,     Yang Wang, and Fang Chen</i>	
TrajAngleNet: Transformer-Based Trajectory Prediction Through Multi-task Learning with Angle Prediction . . . . .	166
<i>Vibha Bharilya and Neetesh Kumar</i>	
Correlation Disentangling and Spatio-Temporal Cooperative Optimizing Network for Temperature Prediction Revision . . . . .	181
<i>Aoao Wei, Xitie Zhang, Suping Wu, Shaohua Yang, Junfeng Zhao,     and Kehua Ma</i>	
Hierarchical Adaptive Position Encoding-Based Transformer for Point Cloud Analysis . . . . .	197
<i>Jiawei Yao, Junfeng Yao, Yong Yang, and Chunyang Huang</i>	
In-Context Learning for Temperature Field Reconstruction Under Multiple Layouts . . . . .	211
<i>Wenzhe Zhang, Zixue Xiang, Ming Sun, and Wen Yao</i>	
Loosely Coupled Oscillators as a Correlate of Behavioral Control Circuits Within the Central Complex of the Fruit Fly . . . . .	226
<i>Saul Garnell, Mehmet Turkcan, Maryam Doborjeh, Brian Smith,     and Paul Szyszka</i>	
EL-LSTM: A Multivariate Time Series Forecasting Model Combining Spiking Neurons and Long Short-Term Memory Networks . . . . .	241
<i>Lei Yang, Yuhang Jiang, Kaixin Wang, Pinjie Zhao, and Kangshun Li</i>	
A Two-Stage Network for Enhanced Intracranial Artery 3D Segmentation in TOF-MRA Volume . . . . .	256
<i>Bin Hu, Shi-Qi Liu, Xiao-Liang Xie, Xiao-Hu Zhou, Tao Wang,     Ji-Chang Luo, De-Lin Liu, Zeng-Guang Hou, and Jia-Xing Wang</i>	
Independence Constrained Disentangled Representation Learning from Epistemological Perspective . . . . .	271
<i>Ruoyu Wang and Lina Yao</i>	
Utilizing Small and Large Spectral Radii for Appropriate Reservoir Computing Design . . . . .	286
<i>Bungo Konishi, Akira Hirose, and Ryo Natsuaki</i>	

Noisy Deep Ensemble: Accelerating Deep Ensemble Learning via Noise Injection .....	301
<i>Shunsuke Sakai, Shunsuke Tsuge, and Tatsuhito Hasegawa</i>	
LCNet: Lightning Hierarchical Convolution for Occupancy Flow Prediction ...	318
<i>Yanjie Zhao and Zhongwen Xiao</i>	
FedTS: Leveraging Teacher-Student Architecture in Federated Learning Against Model Heterogeneity in Edge Computing Scenarios .....	330
<i>Zihong Lin, Yucheng Tao, and Haopeng Chen</i>	
Physics-Informed Antisymmetric Recurrent Neural Networks for Solving Nonlinear Partial Differential Equations .....	350
<i>Pavodi Maniamfu, U. A. Md. Ehsan Ali, Ko Sakai, and Keisuke Kameyama</i>	
APS: An Adaptive Policy Switching Framework to Improve the Generalization of Branching Policy .....	366
<i>Ce Zhang, Dapeng Li, Lin Lin, Xinyue Lu, and Guoliang Fan</i>	
Efficient Pruning and Compression Techniques for Convolutional Neural Networks to Preserve Knowledge and Optimize Performance .....	381
<i>Jakub Skrzyski and Adrian Horzyk</i>	
Enhancing Convnets with Pruning and Symmetry-Based Filter Augmentation .....	396
<i>Igor Ratajczyk and Adrian Horzyk</i>	
Improved Approximation Algorithms for the Cumulative Vehicle Routing Problem .....	410
<i>Jingyang Zhao and Mingyu Xiao</i>	
<b>Author Index .....</b>	<b>427</b>



# Defend from Scratch: A Diffusion-Based Proactive Defense Method for Unauthorized Speech Synthesis

Ying Wang, Yuchuan Luo<sup>(✉)</sup>, Zhenyu Qiu, Lin Liu, and Shaojing Fu

College of Computing, National University of Defense Technology,  
Changsha 410073, China

{wangying,luoyuchuan09,qiuzhenyu22,liulin16,fushaojing}@nudt.edu.cn

**Abstract.** With the advent of Deep Learning (DL), speech synthesis technologies have made remarkable progress, enabling the creation of highly realistic human voices. Although this technology offers numerous benefits, it also introduces substantial security risks, particularly through “Deepfake” speech attacks. These attacks pose severe threats to personal security and societal trust. Existing defense mechanisms, primarily focused on post-attack detection, fall short when confronted with the advanced sophisticated speech synthesis techniques. Worse still, these defenses are often insufficient because significant harm may have already been inflicted by the time deepfake audio is identified. In this paper, we present a novel proactive defense method against unauthorized speech synthesis named “Defend from Scratch” (DFS). By leveraging the idea of adversarial examples, our method can proactively hinder the creation of deepfake speeches. To do that, we propose to incorporate a pretrained decoupled denoising diffusion model (DDDM) to introduce robust and imperceptible adversarial perturbations into the source audio, which enables effectively defense against adaptive attacks without significant audio quality downgrade. Furthermore, Enhanced defenses have been implemented through the application of ensemble learning, which extend its capability to counter a diverse range of threats, thereby ensuring robust voice privacy protection without compromising the integrity or usability of the original audio. Experiments show that our approach stands out for its efficacy in maintaining a high standard of voice privacy in the face of emerging Deepfake technologies.

**Keywords:** DeepFake Defense · Adversarial Attacks · Speech Synthesis · Generative AI · Machine Learning

## 1 Introduction

The development of Deep learning (DL) has significantly advanced speech synthesis technologies, which enable the synthesis of human voices with high fidelity, and make them play a pivotal role across a wide spectrum of applications. These

include enhancing user engagement with virtual assistants and enabling more natural and interactive dialogues with chatbots. However, they also pose significant security risks of speech attacks, commonly known as “Deepfakes”.

**The Risks of DeepFake.** The spread of deepfake speech poses significant challenges to the fabric of truth and trust in the society. The deepfake speech can be misused for malicious purposes [1], such as blackmail, fraud, and cyberbullying, which present a threat to personal safety. Moreover, the capacity of deepfake speech to create convincing false narratives has amplified concerns over disinformation, particularly in sensitive areas like politics and public opinion [2,3], such as harming reputations, manipulating public sentiment, swaying elections, and eroding democratic processes. As a result, the need for vigilance and advanced defense to deepfake speech has never been more critical.

**Existing Defenses.** In response to the critical threats of deepfake speech, existing research has predominantly concentrated on post-attack detection strategies, employing methods based on either hand-crafted or data-driven feature analysis to identify deepfake speeches [4–6]. However, these post-attack detection methods may fall short when confront with the advanced sophisticate speech synthesis techniques based on CNN algorithms. Furthermore, such defenses frequently prove inadequate, as substantial damage may have already been caused before the detection of deepfake speech. Although recent works utilizing deep learning have achieved remarkable success on deepfake speech detection, the post-fake detection methods are inherently limited by their inability to prevent deepfake creation and vulnerability to being bypassed. There is a pressing need for more proactive and robust defense approaches that can prevent the creation of deepfake speech from the outset. In this context, Yu et al. [7] propose AntiFake, a proactive defense approach that use adversarial attacks to prevent unauthorized synthesis. However, this method adds unnatural and conspicuous changes to the original audios, resulting in poor acoustic quality, making the audios unusable in their original applications. Moreover, prior research has demonstrated that the reverse diffusion process can purify the added adversarial perturbations [8], which implies that adaptive attackers may compromise the integrity of AntiFake’s protective measures by leveraging this phenomenon.

**Defend from Scratch.** An ideal audio defense approach should introduce only natural or imperceptible adversarial perturbations to the original audios to guarantee their usability in their original application while maintaining robustness against adaptive attacks. In order to achieve this goal, we propose a novel diffusion-based proactive defense method called Defend from Scratch (DFS). By incorporating a pretrained decoupled denoising diffusion models (DDDM) [9], DFS disrupts the synthesis process by introducing robust and imperceptible adversarial perturbations in the speaker embedding. This effectively prevents attackers from extracting timbral features from the audio, thereby providing a robust defense against deepfake speech generation without significant downgrade in audio quality. Specifically, we first use diffusion models to encode an original audio into the Mel-spectrogram latent space, then introduce adversar-

ial perturbations by interacting with an ensemble of synthesizers during the reverse denoising process, finally generating adversarial audios that are not only of high quality but also exhibit transferability and robustness. Leveraging these diffusion-based adversarial perturbations, we can effectively prevent attackers' malicious speech synthesis.

## Contributions

- Innovative Defense Approach: We propose a novel diffusion-based proactive defense method that employs a pretrained decoupled denoising diffusion model (DDDM) to craft adversarial audios. By introducing subtle perturbations into the latent space, ensuring robustness against unauthorized speech synthesis while maintaining the fidelity and usability of the original audio.
- Enhanced Defense through Ensemble Learning: Our method leverages ensemble learning to create a defense mechanism that is highly effective against an array of unknown attack models. By integrating diverse state-of-the-art (SOTA) speech synthesizers and refining defenses through a joint optimization process, our approach significantly disrupts the distinctive acoustic features utilized by potential attackers.
- We substantiate the effectiveness of our defense mechanism through objective metrics, achieving an Authentication Evasion Reduction Rate (AEER) exceeding 98% across various speech synthesis systems. Additionally, the high average Perceptual Speaker Dissimilarity (PSD) scores, rated at  $4.91 \pm 0.27$  by human evaluators, outperform the SOTA baseline and demonstrate that the adversarial audios exhibit minimal similarity in extracted voice features to the original speaker's voice, validating the success of our proactive defense strategy in both quantitative and qualitative terms.

## 2 Related Work

### 2.1 Adversarial Examples for Defense

Adversarial examples are carefully designed inputs that have been crafted to deceive the model into producing incorrect outputs. Beyond their malicious uses, they can also serve for beneficial purposes [7, 10–12]. V-Cloak [11] and Voice-Cloak [11] injected adversarial perturbations into human voices to mask speaker identities from recognition systems, effectively achieving voice anonymization. The recent studies, AntiFake [7] and SongBsAb [13], aim at AI-generated speech and singing voices. These proactive approaches utilize adversarial perturbations to prevent unauthorized synthesis of speech or singing. Despite the combination of  $l_p$ -based perturbation measurements with human perception principles, the quality of the perturbed audio samples remains substandard, readily provoking suspicion among potential attackers. Thus adaptive attackers may attempt to leverage transformation operations, optimization, or even reverse diffusion process [14, 15] to remove perturbations and proceed with synthesis with them.

## 2.2 Diffusion Models for Adversarial Attacks

Recently, Diffusion Models emerged as state-of-the-art methods capable of generating realistic, high-resolution images with stable training, and their strong generative capabilities have been applied to craft adversarial examples. AdvDiffuser [16] employs these models to produce natural and stealthy adversarial examples, while deceiving advanced classifiers without affecting human perception. The Semantic Adversarial Attack framework [17] manipulates semantic properties for high-fidelity, versatile, and transferable attacks. Diff-PGD [18] enhances adversarial sample generation with improved stealth and controllability, demonstrating improved transferability and anti-purification capabilities compared to traditional methods. Building on these advancements, we propose a proactive defense against voice DeepFakes, leveraging the naturalness and resistance to purification of diffusion model-based attacks, while maintaining human imperceptibility and enhancing defense robustness against adaptive threats.

## 3 Background

**DDDM-VC.** The decoupled denoising diffusion model for voice conversion (DDDM-VC) [9] employs a novel approach to disentangle representations within voice conversion tasks. This method allows for precise control over individual speech attributes and facilitates high-fidelity waveform reconstruction. Distinct from traditional diffusion models, DDDM-VC employs  $N$  denoisers, each with a disentangled representation. The attribute-specific disentangled representation  $Z_n$  serves as the prior for its corresponding denoiser. The forward process generates the noisy sample  $X_{n,t}$  can be formulated as:

$$dX_{n,t} = \frac{1}{2}\beta_t(Z_n - X_{n,t})dt + \sqrt{\beta_t}d\vec{W}_t, \quad (1)$$

where  $n \in [1, N]$ ,  $n$  denotes each attribute.  $\beta_t$  signifies a non-negative noise schedule function, and  $\vec{W}_t$  denotes the forward Wiener process. The reverse process for each disentangled denoiser is given by:

$$d\hat{X}_{n,t} = \left( \frac{1}{2}(Z_n - \hat{X}_{n,t}) - \sum_{n=1}^N s_{\theta_n}(\hat{X}_{n,t}, Z_n, t) \right) \beta_t dt + \sqrt{\beta_t} d\overleftarrow{W}_t, \quad (2)$$

where  $t \in [0, 1]$ ,  $s_{\theta_n}$  denotes the scoring function with parameter  $\theta$ .  $\overleftarrow{W}_t$  is the backward Wiener process. The reverse process (2) is trained by the optimizing objective as follows:

$$\theta_n^* = \arg \min_{\theta_n} \int_0^1 \lambda_t \mathbb{E}_{X_0, X_{n,t}} \left\| \sum_{n=1}^N s_{\theta_n}(X_{n,t}, Z_n, s, t) - \nabla \log p_{t|0}(X_{n,t}|X_0) \right\|_2^2 dt, \quad (3)$$

where  $\theta = [\theta_1, \dots, \theta_N]$  and  $\lambda_t = 1 - e^{-\int_0^t \beta_s ds}$ .

The DDDM-VC framework consist of a source-filter encoder and decoder. It initiates by disentangling speech into three core attributes: content, pitch,

and speaker representations. The filter encoder processes content and speaker information, whereas the source encoder manages pitch and speaker attributes to generate source and filter Mel-spectrogram, referred to as  $Z_{src}$  and  $Z_{ftr}$ . Utilizing these representations as priors, the decoder synthesizes a target Mel-spectrogram from each attribute's prior. This advancement in DDDM-VC significantly enhances the precision and quality of voice conversion applications.

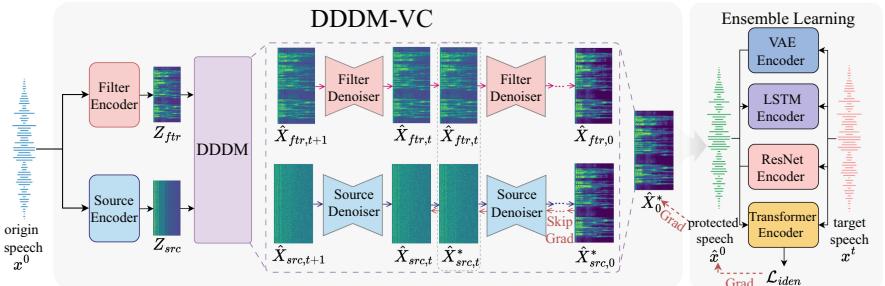
## 4 Method

### 4.1 Problem Formulation

The overall pipeline is shown in Fig. 1. Given the speech audio  $x^0$  to be protected, the target speech audio  $x^t$ , along with the speaker embeddings encoder  $g(\cdot)$ , We employs a diffusion model DDDM( $\cdot$ ) to safeguard the original audio, yielding the protected speech audio  $x$ . This process is designed for targeted protection and can be formulated as:

$$\min \mathcal{L}_{iden}(x) = \mathcal{D}(g(x), g(x_t)), \text{s.t. } x = \text{DDDM}(x^0), \quad (4)$$

where  $\mathcal{D}(\cdot)$  denotes the metric for assessing disparities between speaker embeddings.



**Fig. 1.** Pipeline of Diffusion-based Proactive Defense Method.

### 4.2 Diffusion-Based Adversarial Audio

To generate natural and powerful adversarial audios for DeepFake defense, we propose to introduce adversarial guidance noise in the Mel-spectrogram latent space. We assume that speech synthesis model mainly focuses on style-relevant representation and tries to reconstruct these features during the synthesis process. To leverage this assumption, we employ a diffusion model to map audio into a latent space, and introduce adversarial perturbations in the source-related latent space during the reverse process of pretrained DDDM-VC model. Because

the latent space is perceptually equivalent to the time-domain space, we adopt the projected gradient descent (PGD) [19] attack to introduce suitable perturbations. Different from traditional proactive defenses that rely on perturbations based on benign audio inputs, leading to a degradation of acoustic quality and potentially attracting the attention of attackers, our approach use a diffusion model to construct adversarial audios completely from scratch, thus enhances the quality and imperceptibility of the adversarial audios.

This iterative optimization process is presented in Algorithm 1. At each reverse step, it starts with sampling latent  $\hat{X}_{src,t}$  from the previously perturbed latent variable  $\hat{X}_{src,t+1}$ , and get an approximate result  $\hat{X}_0$  predicted from the Denoiser  $D_{n,t}(\hat{X}_{n,t})$ . In pursuit of minimizing the adversarial loss, the original latent representation  $\hat{X}_{src,t}$  is updated to its adversarial counterpart  $\hat{X}_{src,t}^*$  using the Projected Gradient Descent (PGD) method. This update is mathematically expressed as:

$$\hat{X}_{src,t}^* = \mathcal{P}_\epsilon \left( \hat{X}_{src,t} + \eta_t \cdot \text{sign}(g_k) \right) \quad (5)$$

where  $\mathcal{P}_\epsilon(\cdot)$  denotes the projection operator that confines the perturbed latent within an  $l_\infty$ -norm ball of radius  $\epsilon$ . The adversarial guidance is derived from the sign of the gradient of the identity loss  $\mathcal{L}_{iden}(x)$ . It is noteworthy that as the number of reverse steps increases, the step length  $\eta_t$  is dynamically reduced to align with the reverse process of the diffusion model.

### 4.3 Diffusion-Based Proactive Defense

Motivated by the high-quality, transferability and robustness of diffusion-based adversarial attacks, we propose to create adversarial audio from scratch to proactively defend the unauthorized speech synthesis. In scenarios where the attacker's model is unknown, we utilize ensemble learning to disrupt audio features and a skip-gradient technique to boost computational efficiency, thereby improving the method's applicability.

*Enhancing Defense Performance via Ensemble Learning.* In the real world, the specific models used by attackers are typically unknown and inaccessible, presenting a significant challenge in crafting adversarial examples capable of withstanding various audio DeepFake models. Despite differences in model designs, the core functionality of these synthesizers' encoders is to capture distinctive acoustic features of individual speakers. Therefore, as long as we induce sufficiently significant alterations to the features of the protected audio within a certain range, our method can be highly effective in defense, even when confronted with unknown models.

To achieve this, implement an ensemble learning strategy to manipulate the feature space. In our work, we selected the same state-of-the-art speech encoders as [7], including diverse architectures ranging from ResNet to LSTM. By integrating the identity loss from each encoder and performing a joint backward pass to refine the perturbations, the synthesized adversarial samples are designed to be effective against an array of unknown models that adversaries might utilize.

**Algorithm 1.** DFS: Diffusion-based Proactive Defense Algorithm

---

**Input:** origin speech audio  $x^0$ , target speech audio  $x^t$ , speaker embedding encoder  $\epsilon(\cdot)$ , source-filter encoder  $\varphi(\cdot)$ , Denoiser  $D_{n,t}(\cdot)$ , vocoder  $H(\cdot)$ , distance function  $d(\cdot)$ , SDE solver steps  $N$ , defense iterations  $N_a$ , ensemble encoder number  $N_e$ , momentum factor  $\mu$ , adversarial gradient  $g_k$ , step length  $\eta_t$

- 1: Calculate  $Z_{src}$  and  $Z_{ftr}$  by source-filter encoder  $\varphi(x^0)$
- 2: **for**  $k = 1, \dots, N_a$  **do**
- 3:   Initialize  $h = 1/N$ ,  $\delta_0 \leftarrow 0$ ,  $g_0 \leftarrow 0$
- 4:   // **Diffusion-based Adversarial Audio**
- 5:   **for**  $t = 1, 1-h, \dots, h$  **do**
- 6:      $d\hat{X}_{n,t} = \left( \frac{1}{2}(Z_n - \hat{X}_{n,t}) - \sum_{n=1}^N s_{\theta_n}(\hat{X}_{n,t}, Z_n, t) \right) \beta_t dt + \sqrt{\beta_t} d\tilde{W}_t$
- 7:      $\hat{X}_{n,t} \leftarrow \hat{X}_{n,t+1} - d\hat{X}_{n,t+1}$   
vs.skip 1 mm
- 8:      $\hat{X}_0 \leftarrow D_{n,t}(\hat{X}_{n,t})$   
vs.skip 1 mm
- 9:      $\hat{x}^0 = H(\hat{X}_0)$   
vs.skip 1 mm
- 10:    // **Ensemble learning**  
vs.skip 1 mm
- 11:     $\mathcal{L}_{iden}^i = d(\epsilon(\hat{x}^0), \epsilon(x^t))$   
vs.skip 1 mm
- 12:     $\mathcal{L} = \sum_{n=1}^{N_e} \mathcal{L}_{iden}^i$   
vs.skip 1 mm
- 13:    // **Skip Gradient**  
vs.skip 1 mm
- 14:     $\nabla_{\hat{X}_{src,t}} \mathcal{L} \leftarrow \rho \frac{\partial \mathcal{L}}{\partial \hat{X}_0}$
- 15:     $g_k \leftarrow \mu \cdot g_{k-1} + \frac{\nabla_{\hat{X}_{src,t}} \mathcal{L}}{\|\nabla_{\hat{X}_{src,t}} \mathcal{L}\|_1}$
- 16:     $\hat{X}_{src,t}^* \leftarrow \mathcal{P}_\epsilon \left( \hat{X}_{src,t} + \eta_t \cdot \text{sign}(g_k) \right)$
- 17:   **end for**
- 18: **end for**

**Output:** The protected speech audio  $\hat{X}_0^*$ .

---

*Improving Computation Efficiency.* For a complete denoising process that encompasses  $T$  sequential calculation graph, the direct computation of gradients  $\frac{\partial \mathcal{L}}{\partial \hat{X}_{src,t}^*}$  is high expense or even feasible due to GPU memory overflow. In this work, we use a skip-gradient method proposed in [20] to address this problem.

$$\frac{\partial \mathcal{L}}{\partial \hat{X}_{src,t}^*} = \frac{\partial \mathcal{L}}{\partial \hat{X}_{src,0}^*} \cdot \frac{\partial \hat{X}_{src,0}^*}{\partial \hat{X}_{src,h}^*} \cdot \frac{\partial \hat{X}_{src,h}^*}{\partial \hat{X}_{src,2h}^*} \cdots \frac{\partial \hat{X}_{src,t-h}^*}{\partial \hat{X}_{src,t}^*}, \quad (6)$$

The entire calculation graph of  $\frac{\partial \mathcal{L}}{\partial \hat{X}_{src,t}^*}$  consists of two parts, as expressed in Eq. (6). The initial segment, denoted by  $\frac{\partial \mathcal{L}}{\partial \hat{X}_0^*}$ , represents the gradient of the identity loss with respect to the reconstructed mel-spectrogram  $\hat{X}_{src,0}^*$ , and it leads the direction of the adversarial gradient. Subsequently, the computation

of the iterative product of  $\frac{\partial \hat{X}_{src,t}^*}{\partial \hat{X}_{src,t+1}^*}$  represents the back-propagation process through the generative model. Through a skip gradient strategy, the gradient back-propagation process approximates to  $\frac{\partial \hat{X}_{src,0}^*}{\partial \hat{X}_{src,t}^*}$ .

Let  $\alpha$  and  $\theta$  be positive constants and consider the following SDE:

$$dX(t) = -\alpha X(t)dt + \sigma dW(t), \quad (7)$$

A function (or a path)  $X(t)$  is a solution to the differential equation above if it satisfies:

$$X(t) = X(0)e^{-\alpha t} + \int_0^t \sigma e^{\alpha(s-t)} dW(s), \quad (8)$$

Rearranging Eq. (8), we get the expression  $X(0) = \frac{1}{e^{-\alpha t}} X(t) - \frac{\int_0^t \sigma e^{\alpha(s-t)} ds}{e^{-\alpha t}}$ . As the timestep  $t$  approaches to 1, we deduce that  $\lim_{t \rightarrow 1} \frac{\partial \hat{X}_{src,0}^*}{\partial \hat{X}_{src,t}^*} = \lim_{t \rightarrow 1} \frac{1}{e^{-\alpha t}} \approx \rho$ , where  $\rho$  is a constant. This approximation results in a simplified gradient for the denoising process, given by  $\frac{\partial \mathcal{L}}{\partial \hat{X}_{src,t}^*} = \rho \frac{\partial \mathcal{L}}{\partial \hat{X}_{src,0}^*}$ , which significantly reduces computational complexity and memory usage. Furthermore, considering the relationship  $\hat{X}_0^* = \hat{X}_{src,0}^* + \bar{X}_{ftr,0}$ , where  $\bar{X}_{ftr,0}$  is a independent of the source representation  $\hat{X}_{src,0}^*$  and remains unperturbed, it is deduced that  $\frac{\partial \mathcal{L}}{\partial \hat{X}_{src,0}^*} = \frac{\partial \mathcal{L}}{\partial \hat{X}_0^*}$ . Consequently, the gradient can be efficiently computed using the loss function evaluated with respect to the Mel-spectrogram of adversarial audio  $\hat{X}_0^*$ .

## 5 Experiments

### 5.1 Experimental Settings

**Speech Synthesis Models.** We focus on speech synthesizer with the state-of-the-art zero-shot capabilities, and choose five advance models, including four baseline systems SV2TTS [21], YourTTS [22], TorToiSe<sup>1</sup>, and Adaptive Voice Conversion (AdaptVC) [23], along with a commercial platform ElevenLabs. Text-to-Speech (TTS) systems like SV2TTS, YourTTS, and TorToiSe are designed to convert text into a synthetic voice, allowing users to create high-quality, natural-sounding voice output and customization. Adaptive Voice Conversion (AdaptVC), on the other hand, enables the conversion of one voice to another, adjusting the tone, pitch, and other characteristics of the voice, making it a valuable tool for voice impersonation and transformation, such as in entertainment or security-related scenarios. Meanwhile, ElevenLabs stands out as a commercial platform, offering a range of services, including text-to-speech, voice cloning, and voice conversion.

---

<sup>1</sup> TorToiSe: <https://github.com/neonbjb/tortoise-tts>.

**Speaker Verification Systems and Setup.** We employ CAM++ [24] and Resnet293 [25] as our speaker verification systems which have achieved notably low EERs of 0.745% and 0.447%, respectively, on the VoxCeleb-dev datasets. Leveraging the Python package offered by [26], we perform speaker verification (SV) tasks to verify if a given voice recording corresponds to the identified speaker. This process enable us to evaluate the efficacy of our work in preventing unauthorized speech synthesis.

**Speech Datasets.** We evaluate our method on two commonly used speech datasets: VCTK [27] and LibriSpeech [28]. Following the experimental methodology designed in Antifake, We select 25 speakers from each source corpus, and each speaker providing five audio samples from short (3–4 s) to long (6–7 s). In addition, we employ the speech content same to that used in Antifake for the purpose of DeepFake speech synthesis. Then we leverage the speech synthesizers, audio samples and DeepFake content mentioned before, resulting in 30,000 synthesized speech samples ( $50 \times 5 \times 24 \times 5$ ). We chose samples that not only evaded at least one SV system but also maintained high fidelity and perceptual similarity to the authentic speaker among these numerous outputs. We obtained a collection of 300 synthesized speech clips, with 100 samples able to bypass the authentication mechanisms of each speaker verification system. Subsequently, the DeepFake audio samples were utilized in the proactive processing and assessment stages. For each original victim audio, we selected four distinct target speakers, including two of the same gender and two of a different gender, to ensure a comprehensive and diverse evaluation of our method’s efficacy.

**Evaluation Metrics.** We assess performance using three objective metrics, Authentication Evasion Reduction Rate (AERR), Cosine Similarity and Mean Opinion Score (MOS), alongside a subjective measures, Perceptual Speaker Dissimilarity (PSD). AERR is the measure of how successful a system is at reducing the number of successful evasion attempts. It is calculated by comparing the number of successful evasions before and after the implementation of the security enhancements. Cosine similarity between the centroid identity feature of the victim voice and the identity feature of the synthetic speech output is used to measure identity disruption. The MOS, derived from the NISQA [29] deep learning model, predicts speech quality on a scale of 1 to 5. This score measures the overall audio quality of the processed samples, with higher values being preferable. Typically, a MOS of 3 or higher is considered to indicate satisfactory speech quality. The PSD score is obtained through subjective listening tests, where synthesized voices are compared with the original speech. The PSD scale ranges from 1 to 5, with 1 signifying minimal dissimilarity and 5 indicating maximum dissimilarity.

## 5.2 Experimental Results

**Overall Performance Comparison with Baseline.** The results presented in Table 1 demonstrate the superior performance of our method (DFS) over the existing baseline. Regarding AEER, our findings indicate that the protection efficacy against the chosen speech synthesis systems consistently surpasses 98%, outperforming the baseline research AntiFake and validating the potency of our approach. Additionally, the average PSD scores, as assessed by human evaluators, are notably high ( $4.91 \pm 0.27$ ). This also suggests that audio samples produced by DFS bear no resemblance to the original speaker’s voice, thereby effectively implementing proactive defense. Regarding the MOS comparisons, the audio quality of our method is slightly inferior to the baseline, which can be ascribed to the synthesis capabilities of the DDDM-VC model. As more advanced generative models emerge, it is anticipated that even more realistic adversarial examples will be feasible to do good. As Table 2 shows, compared with the original source audios  $\mathcal{L}$ , the undefended speech audio  $y$  exhibits higher identity similarity, while the defended counterparts  $\hat{y}$  show significantly lower identity similarity. This indicates the robust defense capability of our method against all tested Audio DeepFake models across two datasets.

**Table 1.** Overall performance comparison with baseline.

	MOS	AdaptVC		SV2TTS		YourTTS		Tortoise		ElevenLabs	
		AEER	PSD	AEER	PSD	AEER	PSD	AEER	PSD	AEER	PSD
AntiFake	$3.42 \pm 0.37$	99.4%	$4.81 \pm 0.21$	100%	$4.89 \pm 0.57$	98.2%	$4.90 \pm 0.33$	99.2%	$4.71 \pm 0.65$	96.4%	$4.86 \pm 0.32$
DFS	$3.34 \pm 0.63$	99.8%	$4.88 \pm 0.32$	100%	$4.90 \pm 0.13$	99.0%	$4.91 \pm 0.27$	99.7%	$4.83 \pm 0.11$	98.5%	$4.90 \pm 0.57$

**Table 2.** Comparison of Identity Similarity between Undefended Speech  $y$ , Defended Speech  $\hat{y}$ , and Original Audio  $\mathcal{L}$

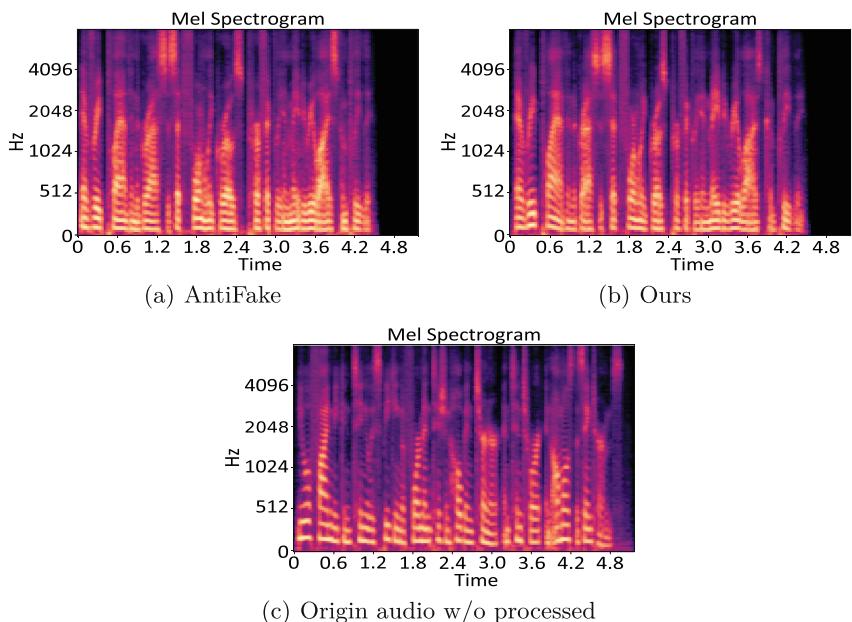
		$\mathcal{L}$	AdaptVC		SV2TTS		YourTTS		Tortoise		ElevenLabs	
			$y$	$\hat{y}$	$y$	$\hat{y}$	$y$	$\hat{y}$	$y$	$\hat{y}$	$y$	$\hat{y}$
VCTK	Antifake	0.18	0.43	0.02	0.48	0.04	0.52	0.05	0.55	0.08	0.79	0.15
	DFS			0.01		0.05		0.05		0.07		0.10
LibriSpeech	Antifake	0.21	0.47	0.03	0.49	0.06	0.54	0.09	0.57	0.12	0.80	0.23
	DFS			0.03		0.07		0.08		0.10		0.16

**Targeted Performance and Cross-Gender Analysis.** In practical applications, non-targeted defense mechanisms may cause synthesized voices to significantly deviate from natural human speech, potentially introducing unnatural

artifacts. As a result, we advocate for targeted defense strategies as more meaningful and valuable.

We implemented our targeted defense method on specific voice samples and evaluated the perceptual speaker identity. Notably, we extended our defense to cross-gender scenarios and were surprised to discover that our approach successfully achieved cross-gender voice protection. With targeted voice of different gender, our method and AntiFake get protected audios, and we use these audios to generate synthesis audio. As depicted in Fig. 2, the Mel-spectrogram of the synthesized audios varied between the unprotected audio, AntiFake, and our method. Both AntiFake and our method derived the same identity from the target’s Mel-spectrum, distinctly different from the original. This outcome may be attributed to perturbations in either the time-domain or latent space that inevitably altered intrinsic audio features, such as pitch, leading to a high success rate in cross-gender voice protection.

Furthermore, compared to AntiFake, the synthesized audio produced by our method was shorter in duration. We hypothesize that our method influences the duration prediction of the speech synthesis model, thereby offering an additional layer of defense against audio DeepFakes by altering the tempo of the synthesized speech.

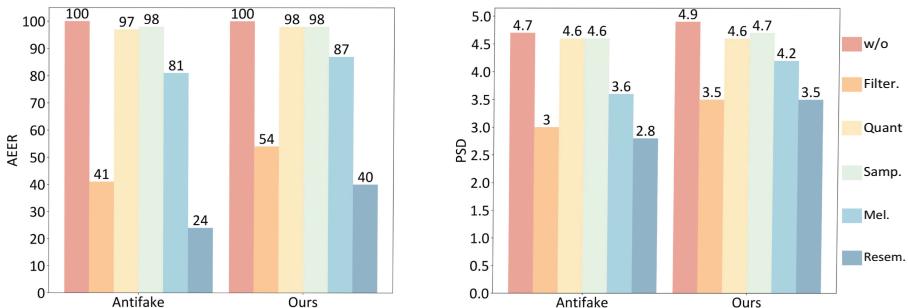


**Fig. 2.** Comparison on Mel-spectrogram of synthesis audio between the undefended audio, AntiFake and our method.

**Time Consuming.** In our experiments, we set up an optimization process with a total of 1000 iterations, and use the early stopping at the same time. The optimization was conducted on an NVIDIA RTX 4090 GPU, completing in 216.5 s. We attribute the relatively short runtime to our implementation of the skip-gradient method, which renders our method computationally efficient and enhances its usability.

### 5.3 Evaluation Against Adaptive Attackers

Our method was further evaluated against adaptive attackers by utilizing five of the most effective strategies to process the audio: Quantization-Dequantization, Down-sampling and Up-sampling, Frequency Filtering, Mel-spectrogram Extraction and Inversion, and AudioPure based on the diffusion model<sup>2</sup>. We believe our evaluation in these five scenarios will demonstrate its advantages in the context of adaptive attacks. The results for these approach are summarized in Fig. 3. Among the five transformations, we found that frequency filtering and AudioPure degraded the protection the most. This is attributed to the fact that these filters removed a significant amount of perturbations from the Mel-spectrogram, more than the other transformations. Specifically, AudioPure, through its forward and backward diffusion process, removed noise that was not representative of the original distribution. Nonetheless, our method demonstrated superior adaptive performance compared to AntiFake. This is due to our defense approach’s inherent resistance to diffusion model-based purification, as it introduces noise across the entire Mel-spectrogram space, in contrast to time-domain noise addition which tends to concentrate on high frequency components.



**Fig. 3.** Comparison of performance against adaptive attackers between AntiFake and our method.

<sup>2</sup> Resemble AI: <https://www.resemble.ai/>.

## 6 Conclusion

The rapid advancement of speech synthesis technologies has inadvertently facilitated the creation of Deepfake speech, posing significant security and societal challenges. Existing defense algorithms, which introduce constrained perturbations to audio to protect against unauthorized synthesis, often compromise acoustic quality. In contrast, we propose a novel proactive defense method grounded in the diffusion-based adversarial framework called Defend from Scratch (DFS). Specifically, we propose to use a decoupled denoising diffusion model for voice conversion (DDDM-VC) as the speech synthesizer. Accordingly, the adversarial audio synthesizing task is formulated as an optimization problem via searching in the hidden space of benign audios projected by DDDM-VC. Meanwhile, the ensemble learning strategy and skip-gradient technique employed not only enhance defense applicability and computational efficiency but also underscore the potential for more sophisticated defense mechanisms against adaptive threats. Through extensive experimentation and evaluation, we demonstrated the superiority of our method against existing baselines, showcasing its effectiveness in various scenarios, including cross-gender voice protection. Our work contributes to the field by advancing the state-of-the-art in voice privacy protection, generating high-quality, robust adversarial audios that are imperceptibly perturbed, thus preserving the usability of the original audio while hindering unauthorized synthesis. We anticipate that our diffusion-based proactive defense method will serve as a foundational framework for future research, extending to other modalities like video and image synthesis, to create a comprehensive protection framework. As the landscape of technologies continue to evolve, we are committed to enhancing our defenses in step with the evolving field, and look forward to contributing to the ongoing development of proactive security strategies.

## References

1. Vyas, K., Pareek, P., Jayaswal, R., Patil, S.: Analysing the landscape of deep fake detection: a survey **12**, 40–55 (2024)
2. Deepfakes and beyond: A survey of face manipulation and fake detection. Inf. Fusion **64**, 131–148 (2020)
3. Chintha, A., et al.: Recurrent convolutional structures for audio spoof and video deepfake detection. IEEE J. Sel. Top. Sig. Process. **14**(5), 1024–1037 (2020)
4. Rodríguez-Ortega, Y., Ballesteros, D.M., Renza, D.: A machine learning model to detect fake voice. In: International Conference on Applied Informatics, pp. 3–13. Springer (2020)
5. Singh, A.K., Singh, P.: Detection of AI-synthesized speech using cepstral & bispectral statistics. In: 2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 412–417. IEEE (2021)
6. Borrelli, C., Bestagini, P., Antonacci, F., Sarti, A., Tubaro, S.: Synthetic speech detection through short-term and long-term prediction traces. EURASIP J. Inf. Secur. **2021**(1), 1–14 (2021). <https://doi.org/10.1186/s13635-021-00116-3>

7. Yu, Z., Zhai, S., Zhang, N.: Antifake: using adversarial audio to prevent unauthorized speech synthesis. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, pp. 460–474 (2023)
8. Wu, S., Wang, J., Ping, W., Nie, W., Xiao, C.: Defending against adversarial audio via diffusion model. arXiv preprint [arXiv:2303.01507](https://arxiv.org/abs/2303.01507) (2023)
9. Choi, H.Y., Lee, S.H., Lee, S.W.: Dddm-vc: decoupled denoising diffusion models with disentangled representation and prior mixup for verified robust voice conversion. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 17862–17870 (2024)
10. Deng, J., Teng, F., Chen, Y., Chen, X., Wang, Z., Xu, W.: {V-Cloak}: Intelligibility-, naturalness- & {Timbre-Preserving}{Real-Time} voice anonymization. In: 32nd USENIX Security Symposium (USENIX Security 23), pp. 5181–5198 (2023)
11. Chen, M., et al.: Voicecloak: adversarial example enabled voice de-identification with balanced privacy and utility. Proc. ACM Interact. Mobile Wearable Ubiquitous Technol. **7**(2), 1–21 (2023)
12. Huang, C.Y., Lin, Y.Y., Lee, H.Y., Lee, L.S.: Defending your voice: adversarial attack on voice conversion. In: 2021 IEEE Spoken Language Technology Workshop (SLT), pp. 552–559. IEEE (2021)
13. Chen, G., Zhang, Y., Song, F., Wang, T., Du, X., Liu, Y.: A proactive and dual prevention mechanism against illegal song covers empowered by singing voice conversion. arXiv preprint [arXiv:2401.17133](https://arxiv.org/abs/2401.17133) (2024)
14. Carlini, N., Tramer, F., Dvijotham, K.D., Rice, L., Sun, M., Kolter, J.Z.: (certified!!) adversarial robustness for free! arXiv preprint [arXiv:2206.10550](https://arxiv.org/abs/2206.10550) (2022)
15. Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., Anandkumar, A.: Diffusion models for adversarial purification. arXiv preprint [arXiv:2205.07460](https://arxiv.org/abs/2205.07460) (2022)
16. Chen, X., Gao, X., Zhao, J., Ye, K., Xu, C.Z.: Advdifuser: natural adversarial example synthesis with diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4562–4572 (2023)
17. Joshi, A., Mukherjee, A., Sarkar, S., Hegde, C.: Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4773–4783 (2019)
18. Xue, H., Araujo, A., Hu, B., Chen, Y.: Diffusion-based adversarial sample generation for improved stealthiness and controllability. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
19. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint [arXiv:1706.06083](https://arxiv.org/abs/1706.06083) (2017)
20. Chen, Z., Li, B., Wu, S., Jiang, K., Ding, S., Zhang, W.: Content-based unrestricted adversarial attack. Adv. Neural. Inf. Process. Syst. **36**, 125–139 (2024)
21. Wan, L., Wang, Q., Papir, A., Moreno, I.L.: Generalized end-to-end loss for speaker verification. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4879–4883. IEEE (2018)
22. 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)
23. Qian, K., Zhang, Y., Chang, S., Yang, X., Hasegawa-Johnson, M.: Autovc: zero-shot voice style transfer with only autoencoder loss. In: International Conference on Machine Learning, pp. 5210–5219. PMLR (2019)

24. Wang, H., Zheng, S., Chen, Y., Cheng, L., Chen, Q.: Cam++: a fast and efficient network for speaker verification using context-aware masking. arXiv preprint [arXiv:2303.00332](https://arxiv.org/abs/2303.00332) (2023)
25. Chen, Z., Liu, B., Han, B., Zhang, L., Qian, Y.: The sjtu x-lance lab system for cnsrc 2022. arXiv preprint [arXiv:2206.11699](https://arxiv.org/abs/2206.11699) (2022)
26. Wang, H., et al.: Wespeaker: a research and production oriented speaker embedding learning toolkit. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1–5. IEEE (2023)
27. Yamagishi, J., Veaux, C., MacDonald, K.: Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit (version 0.92) (2019). <https://api.semanticscholar.org/CorpusID:213060286>
28. Panayotov, V., Chen, G., Povey, D., Khudanpur, S.: Librispeech: an asr corpus based on public domain audio books. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5206–5210. IEEE (2015)
29. Mittag, G., Naderi, B., Chehadi, A., Möller, S.: Nisqa: a deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. arXiv preprint [arXiv:2104.09494](https://arxiv.org/abs/2104.09494) (2021)



# Transformers as Approximations of Solomonoff Induction

Nathan Young<sup>(✉)</sup> and Michael Witbrock

University of Auckland, Auckland, New Zealand  
`{nathan.young,m.witbrock}@auckland.ac.nz`

**Abstract.** Solomonoff Induction is an optimal-in-the-limit unbounded algorithm for sequence prediction, representing a Bayesian mixture of every computable probability distribution and performing close to optimally in predicting any computable sequence.

Being an optimal form of computational sequence prediction, it seems plausible that it may be used as a model against which other methods of sequence prediction might be compared.

We put forth and explore the hypothesis that Transformer models - the basis of Large Language Models - approximate Solomonoff Induction better than any other extant sequence prediction method. We explore evidence for and against this hypothesis, give alternate hypotheses that take this evidence into account, and outline next steps for modelling Transformers and other kinds of AI in this way.

**Keywords:** Explainability · Interpretability · Solomonoff Induction

## 1 Introduction

It has become something of a cliché for papers in the field of Artificial Intelligence to begin by recognising the rapid progress the field has made in the past few years; progress made possible by the ubiquitous Transformer architecture. Transformers - the basis of Large Language Models (LLMs) - seem to scale without limit, achieving greater performance with greater size and greater amounts of training data.

However, despite their power and ubiquity, Transformers are still poorly understood. Many papers have been written and experiments carried out in pursuit of making these models interpretable, understandable, and explainable, with some success - in particular, identification of patterns identified by particular neurons [2, 15]. But such work is largely ad hoc, with no common approach. The field of interpretability, as well as AI safety research in general, remains pre-paradigmatic.

Having a standardised way of analysing the performance or structure of black box algorithms like Transformers requires a standard model to compare against - some generalisation of Transformers and other sequence predictors, of which all other models can be considered departures. One class of such models are

*unbounded solutions*; that is, solutions to a given problem that ignore computational constraints.

One influential example of such an algorithm is Shannon’s 1950 solution to computer chess [11]: Shannon’s model evaluates the entire game tree, evaluating which moves lead to an optimal outcome if both players play optimally. This requires an absurd amount of memory, but modern chess algorithms evaluate smaller sections of the game tree and use other heuristics to estimate the value of the resulting board layouts. This illustrates how unbounded algorithms can be invaluable theoretical ideals, even when they are useless in practice.

Such an ideal unbounded model of sequence prediction exists, and was proposed in 1964 by Ray Solomonoff. [13] Solomonoff often referred to his conception as “algorithmic probability”, but it is more commonly known today as Solomonoff Induction (which we abbreviate as SolInd). In contrast to Shannon’s chess algorithm, which was merely impractical, SolInd is incomputable without infinite memory and the capability to execute an infinite number of programs in a finite time period. SolInd is considered to be an optimal computable sequence prediction scheme, making only a finite amount of unforced prediction error on any computable sequence relative to any other prediction scheme.

In this paper, we propose that Transformers - and, indeed, all sequence prediction schemes - should not just be measured against SolInd, but modelled and understood as direct approximations thereof. Of course, actual implementations of SolInd - using infinite computing power and infinite memory - are incomputable and impossible within a finite observable universe, and so no practical algorithm can ever implement it directly. Still, we propose that Transformers and Neural Networks (NNs) in general can be thought of as bounded approximations of SolInd, with a similar basic underlying structure supplemented by optimisations not present in the unbounded model.

We will outline some specific reasons to believe that Transformers approximate SolInd, as well as some findings that appear to contradict this hypothesis. We conclude by synthesising these results and outlining next steps for exploring this way of modelling Transformers.

## 2 Background

We will begin by outlining, in short, how Solomonoff Induction works.

We first consider a Universal Turing Machine, or UTM, which we call  $M$ , with a unidirectional input tape, a bidirectional working tape, and a unidirectional output tape. For simplicity, we will assume that  $M$  uses a binary alphabet, but these results generalise to any alphabet. Any input string  $s$  given to  $M$  will result in some output string  $x$ . Since  $M$  is capable of simulating any Turing Machine (TM), there exists an  $s$  that will generate any arbitrary  $x$ . The a priori probability of  $s$  being given as input to  $M$  is considered to be the length  $l(s)$  of  $s$  in bits. Since  $M$  is likely to give  $x$  on many different inputs, we consider the set of minimal input strings  $S$  that produce  $x$  when given to  $M$  (possibly followed by other characters).

The probability  $P$  (with respect to machine  $M$ ) of a string  $x$  is given as:

$$P_M(x) = \sum_{s \in S} 2^{-l(s)}$$

The smallest such  $l(s)$  will usually make up the majority of this sum, so considering only the smallest description of a given program gives a reasonable approximation.

The probability of a given continuation  $y$  of  $x$  - in other words, the probability of the output string beginning with  $xy$ , given that  $M$  has only produced  $x$  so far - is simply:

$$P_M(xy|x) = \frac{P_M(xy)}{P_M(x)}$$

In this way, Solomonoff Induction represents a Bayesian prior probability on every computable string, with each new symbol in the input string triggering a Bayesian update.

An equivalent form of SolInd, given in [13], considers Probability Evaluation Methods (PEMs) instead of TMs, updating probability weights on particular inputs rather than eliminating those that do not produce  $x$ . This conception of SolInd gives equivalent results to the version given above.

SolInd's speed of converging on any given sequence depends on the choice of UTM - if a sequence  $x$  has a shorter description on  $M$  than on some other UTM  $N$ , then  $M$  will make less prediction error on  $x$  than  $N$  will. This has limited importance in the limit, though it may be a vital consideration in practice.

SolInd is considered optimal in the following ways:

- Bounded error: the amount of prediction error SolInd will make on any given sequence (measured in log odds to base 2, making errors comparable with bits) is guaranteed to be less than  $P(x)$ .
- Universality: SolInd's predictions will converge to optimality on every computable string.
- Pareto optimality: while other prediction schemes can predict some sequences better than SolInd, it is impossible for any method to predict a sequence better than any SolInd without predicting some other sequence worse.

Proofs of these claims and further analysis of SolInd's performance can be found in [7, 14].

### 3 Hypothesis

We now outline our key hypotheses, beginning with our reasoning and ending with their significance to AI research.

### 3.1 Reasoning

The basis of this paper was perhaps best expressed by Solomonoff himself in a paper published 32 years after his initial description of SolInd: [14]

Just about all methods of induction can be regarded as approximations to [SolInd], and from this point of view, we can criticize and optimize them. Criticism is always easier if you have an idealized (possibly unrealizable) standard for comparison. [SolInd] provides such a standard.

In other words, it makes sense to analyse any sequence prediction method by comparing it to SolInd, considering any architecture not found in SolInd to be an optimisation that helps it achieve performance closer to SolInd’s theoretical ideal.

Our main consideration is in comparing Transformer models to SolInd, since they currently represent an unopposed state-of-the-art in almost all fields of machine learning and artificial intelligence; however, the intention is to introduce a way of understanding every method by which continuations of sequences can be predicted, including all those invented to the present day and all those which might be invented in the future. Indeed, it is necessary to compare Transformers’ approximations of SolInd to other prediction methods if we are to understand how they achieve their superior performance.

### 3.2 Statement of Hypotheses

We now outline our main hypotheses that we will interrogate in this paper and in future work:

1. Transformers can be modeled as approximations of Solomonoff Induction.
2. Transformers approximate Solomonoff Induction more closely than other methods of sequence prediction (e.g. other neural networks).
3. Transformers can be modeled as approximations of Solomonoff Induction at test time specifically.
4. Stochastic gradient descent approximates Solomonoff Induction during training time, and does this better than other training schemes.

The final hypothesis recognises a key difference between SolInd and machine learning systems; Solomonoff inductors are not ‘trained’ in the same way as NNs, and therefore must be thought of in a slightly different way. For example, when comparing SolInd and an NN, we might consider the SolInd input string  $s$  to contain the entire training corpus used to train the NN. Therefore, we must consider how a NN’s training approximates SolInd, as well as its operation at runtime.

### 3.3 Significance

We expect confirmation and/or refutation of our hypotheses to be consequential for the way Transformers and other machine learning systems are understood. Some possible implications of our hypotheses include:

- **Improvements to architecture:** if the Transformer architecture is found to resemble SolInd in some important way that other NNs lack, this would explain their superior performance and possibly prescribe further improvements.
- **Explainability:** identifying specific ways that Transformers implement different TMs/PEMs and update weights on each would go some way towards explaining how Transformers work and why a particular output was given. (This point is predicated on the assumption that SolInd is naturally more easily explainable than a Transformer. We believe this is reasonable, but that identifying the purpose of a particular TM (and thus the output of any SolInd) is still non-trivial.)
- **Producing explicit models:** once key parts of a Transformer have been identified, it may be useful for both performance and explainability for these parts to be split into separate programs - no longer hiding in inscrutable matrices but operating more like traditional algorithms.
- Finally, if our hypothesis is refuted and Solomonoff Induction is shown to not be a useful model to compare prediction schemes against, the following questions must be raised: Why not? What about Solomonoff's optimal sequence predictor is so unlike its computable counterparts? And is there another model - unbounded or otherwise - against which arbitrary prediction methods *can* be compared?

## 4 Findings in Favour

In this section, we will present several findings that support and shed light on our hypotheses.

### 4.1 Universality

One key result on which our hypothesis relies is the idea that Transformers - and NNs in general - can implement arbitrary programs (equivalently, can emulate arbitrary TMs). That Recurrent Neural Networks (RNNs) can emulate arbitrary TMs was first shown in 1992 by Siegelmann and Sontag [12]. This result has since been replicated for many other machine learning models; indeed, Transformers have had multiple papers showing distinct ways in which they might emulate arbitrary TMs. [1, 10]

Therefore, one way that Bayesian mixtures of TMs could be emulated using a NN is for them to be simply averaged; given a probability distribution over a set of TMs, construct emulations of each on a NN of a certain size and give each weight in the NN its average value among the emulations, weighted by that emulation's probability mass. It is possible that the class of weights that can be achieved by weighted-averaging emulated TMs in this way is equivalent to the class of all randomised weights that can be given to a NN.

## 4.2 Decomposition

There exists a substantial amount of research into NN decomposition [9]; that is, taking a NN trained for some task (or set of tasks) and identifying smaller subnetworks that can perform the task without the rest of the network (or perform one of the set of tasks, with other parts of the network possibly performing other tasks).

One well-supported hypothesis in this space is the *lottery ticket hypothesis* [5]: that randomly-initialised networks contain subnetworks that can achieve similar performance to the whole network if trained in isolation, having been given random values that quickly reach a high level of performance. This implies that such “winning tickets” exist as part of any random initialisation of any NN, for every function that NN is capable of emulating. Further, since a RNN can emulate a UTM with as few as 1058 neurons [12] - many fewer than modern deep NNs - most functions can be reasonably considered to be emulable by most NNs.

In other words, there is good reason to believe that randomly-initialised networks contain something close to a representation of any arbitrary TM, the most relevant of which can be found during training and given increased weight over other subnetworks. This seems to closely mirror SolInd’s process of induction.

## 5 Findings Against

In this section, we will explore findings that seem to contradict or complicate our hypothesis. Of course, computable instantiations of Solomonoff Induction are impossible, so the strongest version of our hypothesis, that Transformers act as Solomonoff Inductors, cannot be absolutely true; any attempt to approximate SolInd will be subject to limitations. These findings can be thought of as exploring and naming these limitations.

### 5.1 Limits of Stochastic Gradient Descent

Computational tasks can be sorted into four categories: regular, context-free, context-sensitive, and recursively enumerable. These categories each represent an increase in the level of complexity required to complete them, forming the Chomsky hierarchy. [3]

After training on tasks at varying levels of the Chomsky hierarchy, Delétang et al. [4] found that Transformers can generalise well to regular tasks, but suffer greater performance penalties on tasks that are further up on the Chomsky hierarchy - eventually leading to recursively enumerable tasks, which correspond to those that can be solved by Turing machines. Other kinds of NNs tested exhibited similar behaviour. The authors speculate that this may be due to the limitations of stochastic gradient descent as a training technique - when weights are gradually changed to approach an ideal value, small imperfections can accumulate.

## 5.2 Computational Limits in Practice

NNs failing to generalise may also be explained by a fundamental limitation of the way they emulate TMs.

To emulate arbitrary Turing machines, an infinitely large tape is required. This can be achieved in NNs with infinite width, but is more commonly achieved by using the value of particular neurons as memory, with the decimal expansion of the neuron’s value (in some base) forming a stack. For this method to achieve arbitrary memory size requires unlimited precision, which is rarely achieved in practice. This may or may not be a relevant limitation in practice; it does not stop finitely large computers from running useful programs, but it may shrink the amount of memory available to a NN below what is needed for many programs.

Another fundamental limitation is that of state memory. If arbitrarily many NNs that emulate TMs are averaged to give a new NN that has the average weights of each (possibly weighted by their probability mass in a distribution, as outlined above), the resulting NN may be able to evaluate the composite transition function in finite time and space, but cannot track arbitrarily many state variables in finite space. This may prevent the SolInd emulation method given above of averaging weights among emulated TMs in a given probability distribution, since their states will, without any extra work, all be tracked in the same set of neurons. This can be worked around if the TMs’ states share meaningful information, but this is not a trivial requirement.

## 5.3 Transformers Make Poor Solomonoff Inductors

Finally, an attempt was made by Grau-Moya et al. [6] to train a Solomonoff Inductor using data sampled directly from a UTM. They trained multiple kinds of machine learning systems, including Transformers. (This is an example of meta-learning; that is, training systems that learn new tasks quickly instead of simply training them on the tasks themselves. This is distinct from our work, which seeks to show that Transformers already have this capability to at least some degree without training.)

Among the models tested in this paper, Transformers often learnt best of all models tested when tested on sequences of lesser or equal length to that given to them at training time. However, when given longer sequences, their performance often became the worst of any model.<sup>1</sup> This suggests that Transformers may not be suited to learning and/or emulating arbitrary TMs in practice.

---

<sup>1</sup> This may be explained by an architectural choice in this paper: Grau-Moya et al.’s Transformer model used Vaswani et al.’s original fixed positional encoding [16], rather than a relative encoding. This may have lead to the Transformer not being able to recognise patterns that were longer than 256 symbols long, as it had not previously seen positional encodings larger than 256. A relative positional encoding may have helped to address this problem.

## 6 Syntheses and Alternate Hypotheses

In this section, we will propose some alternate hypotheses, which are more complicated than those previously outlined but take the above findings into consideration.

### 6.1 Alternate Solomonoff Models

We begin by exploring models that utilise the key insights of SolInd, while taking into account the fact that Transformers may be poor at (and possibly even incapable of) emulating arbitrary TMs.

SolInd represents a Bayesian mixture over all inputs to a UTM, and therefore all Turing machines. But there is no reason why we cannot consider similar probability distributions over alternative methods of computation, lower on the Chomsky hierarchy.

The idea that Transformers can more easily approximate simpler models than TMs, as discussed before in the context of Delétang et al.'s work [4], was also explored by Liu et al. [8], who found that Transformers are well-suited not only to learning functions that can be implemented in finite-state automata (i.e. regular languages), but to finding shortcuts to them that can yield identical solutions in logarithmic time compared to executing the automaton - that is, if the automaton would take  $T$  steps to yield a solution, Transformers can learn to find the same solution in  $O(T)$  time.

It may therefore make more sense to think of Transformers as a mixture of finite-state automata rather than TMs, or possibly other models of computation like Markov chains, pushdown automata, or bounded Turing machines. Since Turing machines are capable of simulating every possible form of computation, including these, it may still make sense to consider decomposing Transformers into TMs, if not *all* TMs.

### 6.2 Alternate Hypotheses

In light of these considerations, we add the following to our list of hypotheses:

5. Transformers can be modelled as Solomonoff inductors with limited memory, preventing them from emulating all TMs and making it harder for them to solve tasks higher on the Chomsky hierarchy.
6. Transformers can be modelled as Bayesian mixtures of simpler computational schemes than the class of all TMs.
7. The further down a model of computation is on the Chomsky, the more quickly it is converged upon by stochastic gradient descent, leading Transformers to give much more weight to some classes of TM than others.

It should be noted that these hypotheses are not all mutually exclusive, and that some middle ground may well explain Transformers better than any individual hypothesis alone. For example, Transformers might represent a probability

distribution over all finite-state automata, as well as executing some TMs that evaluate recursively-enumerable functions that simpler automata cannot approximate.

Still, it is useful to consider which hypotheses have more explanatory power - are Transformers more like TM-infused departures from mixtures of finite-state automata, or UTMs that encode automata well but can still approximate any TM with enough time and space? These two composite hypotheses would prescribe different implications towards explaining and decomposing Transformers, and so must be separated and carefully considered.

## 7 Conclusion

In this paper, we have proposed that Transformers and other NNs should be analysed under the lens of Solomonoff’s optimal unbounded sequence prediction scheme, Solomonoff Induction. Since SolInd represents the upper bound of what any sequence predictor can be capable of, it likely has extensive prescriptive and descriptive power with respect to NNs, but little work has been done to utilise this.

We have outlined reasons why this way of modelling NNs may be helpful in understanding how they work and achieving similar performance using algorithms that are less inscrutable and easier for programmers to work with. We have given several hypotheses as to how Transformers may approximate or relate to Solomonoff Induction.

We have outlined some relevant findings, showing that our hypotheses seem to be reflected in NNs in some ways but are complicated by computational limitations; we have then taken these findings into consideration to develop more complicated hypotheses that may paint a fuller picture of how Transformers approximate SolInd’s optimal sequence prediction in practice.

We hope that future work in this area will help establish a common framework under which NNs can be considered and analysed.

### 7.1 Future Work

There is much subsequent work to be done in this area towards investigating our hypotheses in more detail. The following may be useful next steps:

- Investigate whether Transformers can find shortcuts to, or better representations of, models of computation in between finite-state automata and TMs on the Chomsky hierarchy
- Determine whether sets of Transformers or other NNs of a given size that have been constructed to emulate TMs can be averaged to yield arbitrary weights, confirming that all NNs directly correspond to probability distributions over TMs
- Explore ways of decomposing Transformers into subnetworks that seem to emulate multiple functions using shared memory

- Convert decomposed subnetworks into explicit, non-NN-based algorithms
- Evaluate to what degree training an NN can be considered as part of the induction process, and to what degree it resembles selection of a UTM with short codes for patterns seen in the training corpus

## References

1. Bhattacharya, S., Patel, A., Goyal, N.: On the computational power of transformers and its implications in sequence modeling. arXiv preprint, [arXiv:2006.09286](https://arxiv.org/abs/2006.09286) (2020)
2. Bills, S., et al: Language models can explain neurons in language models. <https://openai-public.blob.core.windows.net/neuron-explainer/paper/index.html>, Accessed 22 July 2024
3. Chomsky, N.: Three models for the description of language. IRE Trans. Inf. Theor. **2**(3), 113–124 (1956)
4. Delétang, G., et al.: Neural networks and the chomsky hierarchy. arXiv preprint, [arXiv:2207.02098](https://arxiv.org/abs/2207.02098) (2022)
5. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint, [arXiv:1803.03635](https://arxiv.org/abs/1803.03635). (2018)
6. Grau-Moya, J., et al.: Learning universal predictors. arXiv preprint, [arXiv:2401.14953](https://arxiv.org/abs/2401.14953) (2024)
7. Hutter, M.: New error bounds for Solomonoff prediction. J. Comput. Syst. Sci. **62**(4), 653–667 (2001)
8. Liu, B., et al.: Transformers learn shortcuts to automata. arXiv preprint, [arXiv:2210.10749](https://arxiv.org/abs/2210.10749) (2022)
9. Liu, X., Parhi, K.: Tensor decomposition for model reduction in neural networks: a review [Feature]. IEEE Circuits Syst. Mag. **23**(2), 8–28 (2023)
10. Pérez, J., Marinković, J., Barceló, P.: On the turing completeness of modern neural network architectures. arXiv preprint, [arXiv:1901.03429](https://arxiv.org/abs/1901.03429) (2019)
11. Shannon, C. E.: XXII. Programming a computer for playing chess. London, Edinburgh, and Dublin Philos. Mag. J. Sci. **41**(314), 256–275 (1950)
12. Siegelmann, H., Sontag, E.: On the computational power of neural nets. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 440–449 (1992)
13. Solomonoff, R.: A formal theory of inductive inference. Part I. Inf. control **7**(1), 1–22 (1964)
14. Solomonoff, R.: Does algorithmic probability solve the problem of induction. Oxbridge Research, POB 391887. (1996)
15. Templeton, A., et al.: Scaling monosematicity: extracting interpretable features from claude 3 sonnet. <https://transformer-circuits.pub/2024/scaling-monosematicity/index.html>, Accessed 22 Jul 2024
16. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30(2017)



# Interpreting Decision Transformer: Insights from Continuous Control Tasks

Dhillu Thambi<sup>1</sup>✉, Praveen Paruchuri<sup>1</sup>, and Perusha Moodley<sup>2</sup>

<sup>1</sup> Machine Learning Lab, IIIT Hyderabad, Hyderabad, India

dhillu.thambi@research.iiit.ac.in, praveen.p@iiit.ac.in

<sup>2</sup> Reading, UK

**Abstract.** Decision Transformers have demonstrated impressive performance in offline reinforcement learning, but their complex internal mechanisms remain challenging to interpret. This paper presents a comprehensive analysis of Decision Transformers(DT) trained on various MuJoCo environments using different interpretability techniques. Our analysis includes: (a) positional encoding (PE) analysis to assess the impact of temporal information on performance in different types of environments, (b) return-to-go (RTG) analysis to explore the variability in achieving specific returns, (c) embedding analysis revealing the models' ability to learn detailed representations of the environments, (d) attention visualizations to understand the role of attention in behaviour guidance, and (e) perturbation studies to test robustness to input noise. Our analysis reveals the extent to which positional information is necessary for achieving high performance in different MuJoCo environments and provides insights into the model's adaptability to varying task requirements. In addition, the embedding analysis shows hierarchical abstractions captured by the model across layers. Through these analyses, we provide novel insights into the decision-making process of transformers in continuous control domains. These insights are pivotal for designing more interpretable and reliable decision-making systems and deepen our comprehension of the capabilities and limitations of DT in complex tasks.

**Keywords:** Decision Transformer · Interpretability and Explainability · Offline Reinforcement Learning

## 1 Introduction

Decision Transformers (DT) [9] have emerged as a powerful offline Reinforcement Learning (RL) architecture, demonstrating impressive performance on offline RL benchmarks [15], including Atari games [3], OpenAI Gym [6], and robotic manipulation tasks [28], showcasing the potential of Transformer-based models

---

P. Moodley—Independent Researcher.

in tackling RL tasks. By leveraging a transformer-based architecture [29] and conditioning on the desired return, DTs can effectively model the distribution of trajectories and generalize to unseen scenarios. The complexity of these models often makes them challenging to interpret, limiting our understanding of their internal workings and decision-making processes. Interpretability is a crucial aspect of AI research, enabling us to build trust in the models we deploy, identify potential biases or failure modes, and gain insights into the underlying mechanisms of learning and decision-making.

In this paper, we present a comprehensive analysis of DT trained on various MuJoCo [28] environments using different interpretability techniques. Our focus is on continuous control tasks that are traditionally harder to interpret and our primary objective is to gain insights into the model’s learned representations and decision-making processes. We aim to answer the following questions

**(a)** To what extent does positional information impact the performance of DT in different types of environments? **(b)** How does the model handle variability in achieving specific returns, and what insights can we gain from Return-to-Go (RTG) analysis **(c)** What hierarchical abstractions and representations of the environment does the model learn across different layers? **(d)** How does the attention mechanism guide model behaviour? **(e)** How robust are DTs to input noise, and what implications does this have for their reliability?

To address these questions, we employ a range of interpretability techniques, including positional encoding analysis, RTG analysis, embedding analysis, attention visualizations, and perturbation studies. These techniques allow us to uncover the roles of individual components, quantify their importance, and trace the flow of information through the model. Our work builds upon recent advances in interpretability research, particularly in the area of mechanistic interpretability [8]. While previous work has primarily focused on interpreting language models and other sequence-to-sequence architectures [5, 17, 24], we analyse DT in the context of continuous control tasks, presenting unique challenges and opportunities. Compared to simpler problems like the grid worlds, continuous control tasks provide a deeper understanding of the learned behaviour of DT in complex and realistic settings, where the agent has to take multiple actions simultaneously. Through extensive experiments on MuJoCo environments, including Half-Cheetah, Hopper, and Walker, we demonstrate the effectiveness of our analysis in interpreting DT for continuous control tasks.

Our analysis provides key insights into the behaviour and decision-making processes of DT in continuous control tasks. Positional encoding importance varies by task complexity, with more complex tasks relying more on temporal context. The model’s ability to adapt behaviour to achieve specific returns depends on the reward distribution in the training dataset. It also learns hierarchical abstractions and representations, capturing meaningful features for decision-making. Attention mechanisms are crucial, with distinct patterns across environments. The model shows robustness to input noise and highlights the importance of specific joints in HalfCheetah for locomotion. This work bridges

the gap between DT performance and understanding, builds trust, and identifies areas for improvement. The insights can inform future architecture designs and training strategies for more powerful and interpretable models.

## 2 Related Works

**Decision Transformers (DT):** Decision Transformers [9] derive from GPT-style Transformers [7, 29] predominantly used for language modelling and generation. DT inherits causal, auto-regressive and generative traits that are applied to RL sequential trajectory data for modelling and generating RL behaviour. While DT has achieved reasonable success in various environments, it is limited by problems such as failure to generalize in complex tasks, over-fitting to a dataset and misalignment between target and achieved returns. This has led to numerous variations and studies to improve the capabilities of these models, the most relevant of which are highlighted next. [4] perform a study demonstrating that stitching is not possible with the original DT. Stitching refers to the model’s ability to combine parts of sub-optimal trajectories from the training data to produce an optimal trajectory or learn to perform optimally from sub-optimal data. [32] proposed a variant called Elastic Decision Transformer to facilitate stitching. [13] compares return-based, such as the return-to-go prompt in the Decision Transformer, with goal-based conditioning and finds that goal-state conditioning performs better in environments that feature goal states such as grid worlds, robotic tasks and maze-like tasks. Return-based conditioning appears to perform better in locomotion-based environments like Half Cheetah, Hopper and Walker [28] where there is no task goal as such e.g., retrieving a key or opening a door.

**Explainability and Interpretability in RL:** The interpretability of RL models [2, 16, 23] has gained significant attention due to the critical need for transparency in high-stakes decision-making domains [27, 33]. A growing body of literature has focused on developing methods [11, 20] to explain the decision-making processes of RL agents. Hilton et al. [18] explored the interpretability of RL models by analyzing their behaviour in the CoinRun environment [10]. They introduced the diversity hypothesis which suggests that a diverse training distribution is linked to the development of interpretable features within the model on the basis that models have no incentive to generalise if the environment and dataset is too simple. [22] make important contributions towards interpretability methods for RL in the context of the AlphaZero chess engine. They achieve this by analyzing the internal representations of RL agents to gain insights into their knowledge and reasoning. **Mechanistic Interpretability:** Mechanistic interpretability [25] focuses on understanding the inner workings of models by breaking down their components [8]. This area entails many processes, some of which are similar to reverse engineering, where the goal is to deduce the model’s learned algorithms by analysis of its weights and activations, flow of data and embedded representations. Bloom et al. [19] expands the recent mechanistic interpretation work on Large Language Models (LLMs) [12]

to Decision Transformers and highlights the complexity involved in interpreting the multimodal Decision Transformer, even on simple discrete GridWorld tasks. The multi-modality of Decision Transformers, which can represent visual input, specified rewards, and previous actions, provides a rich but complex ground for interpretability research. This is very much an ongoing research area.

### 3 Background

#### 3.1 Decision Transformer (DT)

The Decision Transformer (DT) [9] is a novel approach to offline reinforcement learning [21, 26] that frames the RL problem as a sequence modeling task. The DT leverages the expressive power of the Transformer architecture [29] to directly output optimal actions conditioned on a context of historical states, actions and return prompts. By conditioning on the desired Return-to-Go (RTG) the DT learns to make decisions that guide the agent towards the target return. The DT’s input consists of sequences of embedding tokens for the RTG, state, and action at each trajectory timestep, with position embeddings capturing temporal information. RTG embeddings encode the desired cumulative reward, while state and action embeddings capture environment details. Transformer layers use multi-headed self-attention and feed-forward networks to update hidden representations, producing transformed tokens for subsequent layers. During inference, the DT generates future actions autoregressively, conditioned on the desired RTG and past states and actions. This enables the DT to leverage prior experience for decision-making, maximizing expected return without explicitly estimating value functions or computing policy gradients.

#### 3.2 Environments

**Half-Cheetah:** Implemented in MuJoCo [28, 30], the Half Cheetah environment features a 2D model with a 17-dimensional observation space and a 6-dimensional action space. The goal is to maximize forward motion by applying torques to leg joints, with rewards based on the distance covered over 1000 timesteps.

**Hopper:** In the MuJoCo-implemented Hopper environment [14], this 2D one-legged robot has an 11-dimensional observation space and a 3-dimensional action space. The objective is to advance forward, gaining rewards for motion and penalties for falling, ending if the robot falls or reaches 1000 timesteps.

**Walker2d:** The Walker2d environment [14], also in MuJoCo, features a 2D bipedal robot with a 17-dimensional observation space and a 6-dimensional action space. It aims for forward progression, rewarding motion and penalizing falls, concluding when the robot falls or after 1000 timesteps.

## 4 Methodology

### 4.1 Positional Encoding (PE)

Positional encoding in DT is essential for capturing the sequence and multi-step temporal dependencies crucial in RL. We train the DT with and without PE, evaluating its performance in Half-Cheetah, Hopper, and Walker environments. This tests the model’s robustness to input structure changes and the importance of temporal information. Comparing performance in both scenarios helps assess the impact of positional information on decision accuracy and adaptability.

#### **What It Can Tell Us About the Environment and Learned Behavior:**

(a) *Temporal Context Importance*: A significant performance drop without PE indicates their critical role in accurate predictions, especially in dynamic environments like Walker and Hopper. (b) *Model’s Dependency*: The experiment shows the model’s reliance on event sequencing versus state and action information at each step. (c) *Environment Dynamics*: If PE is crucial, it suggests strong sequential dependencies in the environment, where past or future states heavily influence the current decision-making.

### 4.2 Return-to-Go (RTG) Analysis

Return-to-Go (RTG) is crucial in Decision Transformer (DT) models, conditioning the model’s predictions on achieving a specified future return. RTG guides action generation by setting a target reward from the current state to the episode’s end. We analyze the impact of different RTG values on DT models using three settings: original RTG values, their negative values representing low RTG scenarios, and zero RTG. This analysis, applied to HalfCheetah, Hopper, and Walker2D models, assesses the model’s adaptability to achieve specific returns. By testing performance across various RTG values and examining the resulting trajectories and returns, we gain insights into the model’s sensitivity to desired returns and its ability to generate appropriate actions.

### 4.3 Attention Visualizations

Attention mechanisms enable models to focus on relevant parts of the input sequence during decision-making. To understand attention pattern evolution throughout an episode, we visualize the averaged attention across all layers for each token at each timestep. Given our 1000-timestep episodes, traditional interpretability methods like attention matrix visualization, attention rollouts, and flow [1] are impractical. Instead, plotting average attention weights per token at each timestep reveals the model’s learned behaviour over extended sequences, offering a temporal perspective on focus and decision-making. This uncovers attention dynamics, key decision points, and token prioritization throughout an episode. It also highlights how the agent’s focus shifts in response to the environment and allows us to compare attention patterns across layers, providing insights into the hierarchical nature of learned representations and the roles of different layers in capturing various aspects of the RL task.

#### 4.4 Embedding Analysis

We analyse the input embeddings and hidden states to understand the learned representations of the Decision Transformer. Each of the states, actions, and RTG are encoded to the model dimension, which, in our case, is a 128-dimensional vector. These dimensions are maintained through the transformer, with each layer returning the same number of tokens of model dimension. This allows us to visualize the tokens after embedding, and after each layer (hidden states), using t-SNE to identify clusters and patterns in the learned representations. By comparing the representations across layers we gain insight into how the model processes and transforms the input sequence of state, action and RTGs.

#### 4.5 Action (Joint) Perturbation Analysis

We evaluate the DT’s robustness and interpretability by selectively deactivating joints in the agent’s action space during inference. Specific joints are turned off at intervals starting at the 200th timestep and then every 200 timesteps for durations of 10 and 50 timesteps, simulating their deactivation. This approach highlights the significance of each joint and the model’s adaptability to changes. We perturb random joint combinations to gauge overall resilience, followed by individual joint deactivations to assess their importance in decision-making.

#### 4.6 Experimental Setup

**Environments and Datasets:** We evaluate the DT on three D4RL continuous control tasks: HalfCheetah, Hopper, and Walker [15]. Each environment has three dataset settings containing states, actions, and normalised returns: *Expert* (trajectories from an expert policy), *Medium* (partially trained policy), and *Medium Replay* (transitions from a training replay buffer).

**Model Architecture:** We use the DT architecture from the original paper with 3 layers and 1 attention head, implemented using HuggingFace Transformers in PyTorch [31]. The context length is 20 timesteps, meaning the DT considers the past 20 timesteps of states, actions, and rewards for decision-making.

### 5 Results

#### 5.1 Embedding Analysis

In this section, we analyze the embeddings of DT models for HalfCheetah, Hopper, and Walker2d environments, as shown in Figure 1. We apply t-SNE, a technique for dimensionality reduction, to visualize and analyze these embeddings. We make the following observations.

**Input Embeddings and Environment Dynamics:** Input embeddings are the learned representations of the state, action, and RTG input sequence. These tokens are the representations of the trajectory from which the DT model learns the dynamics of each environment. For the HalfCheetah environment, the input embeddings exhibit a clear separation between the states, actions, and RTGs, suggesting that the model has effectively learned to encode the semantic information of these different components into distinct regions of the embedding space. The input embeddings in the Hopper and Walker2d show a similar separation between states, actions, and RTGs.

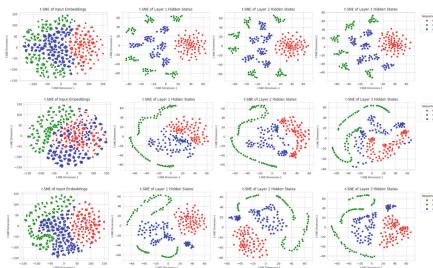
**Hidden States:** The hidden state visualizations show how models integrate and transform information from input embeddings through attention layers. In the HalfCheetah environment, the first layer’s hidden states are less entangled and nicely clustered, indicating the model’s initial learning of integrated representations capturing state, action, and RTG relationships. Progressing through the second and third layers, the hidden states become more structured, forming defined clusters of states and actions. This suggests that the model has learned to disentangle relevant information, forming meaningful representations for each component while still capturing their relationships. The separation of state and action spaces indicates that the transformer has learned to distinguish position encoding from state or action information, a capability absent in the input embedding layer. These clusters likely correspond to the state and action configurations of an optimally performing HalfCheetah.

For Hopper and Walker2d, hidden state visualizations show an initial entangled layer, with subsequent layers exhibiting more structured separations of states, actions, and RTGs. This progression suggests the DT is learning hierarchical representations. Unlike HalfCheetah, Hopper and Walker clusters are sparser and differ in nature, with states almost continuous and some separation in RTGs and actions. These differences, related to the environments, appear in later analyses. Distinct clusters in hidden layers represent learned behavioural modes and strategies like running, gait cycles, or balancing for each environment. The effective separation of states, actions, and RTGs in the input embeddings and final hidden layers in the HalfCheetah and Hopper environments suggests that the DT can encode and process these components individually while also learning their relationships through intermediate hidden layers.

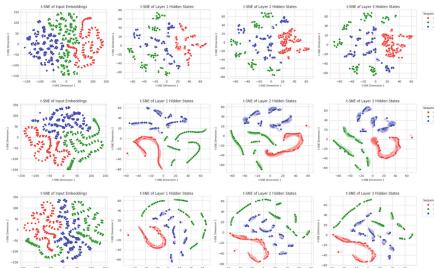
**Similarities in Hopper and Walker Tasks:** The hidden state representations for the Hopper and Walker2d environments exhibit similarities, which can be attributed to the shared underlying dynamics of these environments. The Walker2d environment is an extension of the Hopper, with additional complexity and degrees of freedom. Despite these differences, the similarity in the hidden state representations suggests that the Decision Transformer has captured common behavioural strategies that are effective across both environments.

**Table 1.** Average Returns for DT Models on Half-Cheetah, Hopper, and Walker2D, Averaged Across 100 Episodes for 5 Random Seeds, With and Without Positional Encoding (PE)

Agent	Level	With PE	Without PE
Half-Cheetah	Expert	10900.58	9531.12
	Medium	4989.15	4864.70
	M. Replay	4141.28	3581.95
Hopper	Expert	3267.53	705.42
	Medium	1851.32	951.36
	M. Replay	951.47	681.86
Walker2d	Expert	4943.18	3321.54
	Medium	3402.11	2535.16
	M. Replay	1115.07	242.89



**Fig. 1.** Visualization of input embeddings and hidden states of DT for Halfcheetah (row 1), Hopper (row 2), and Walker2d (row 3) with positional encoding. Column one displays input embeddings, followed by hidden states from layers 1 to 3. Colours represent Return-to-Go (red), States (green), and Actions (blue). (Color figure online)



**Fig. 2.** Visualization of input embeddings and hidden states of DT for Halfcheetah (row 1), Hopper (row 2), and Walker2d (row 3) without positional encoding. Column one displays input embeddings, followed by hidden states from layers 1 to 3. Colours represent Return-to-Go (red), States (green), and Actions (blue). (Color figure online)

## 5.2 Positional Encoding (PE)

We analyze the impact of positional encoding (PE) on the performance and internal representations learned by the DT for three environments. PE incorporates sequential information, aiding in capturing temporal dependencies in control tasks. The varying impact of PE across tasks is due to differences in temporal complexity. Table 1 compares model performance with and without PE. Our analysis shows that PE absence during inference does not significantly affect the HalfCheetah model’s performance, indicating that task characteristics and model architecture influence the need for sequence position information.

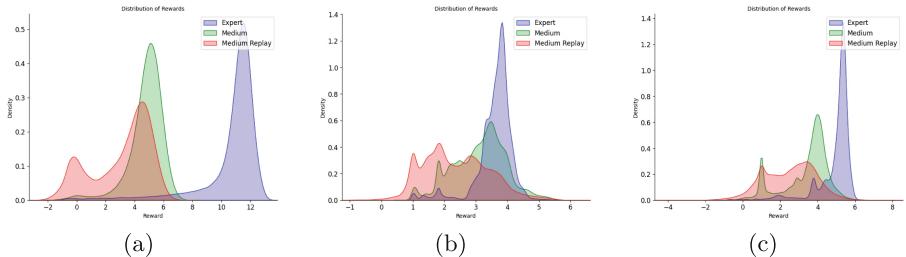
**HalfCheetah: Minimal Impact of Positional Encoding:** The HalfCheetah environment demonstrated robust performance even without PE. In this task, the primary objective is to move forward quickly, requiring a straightforward control policy without complex interactions or a defined termination condition based on the agent’s state. The episode ends after 1000 timesteps. Success depends on the agent’s current velocity and position rather than its sequence position. The HalfCheetah’s state representation includes temporal features like position, velocity, and angular velocity of body parts, encoding sequence information naturally and potentially making PE redundant. The DT model’s ability to achieve high rewards without PE shows it can learn the optimal policy by mapping current state-action pairs to rewards without needing explicit sequence position information. This highlights the DT model’s capability to leverage inherent temporal information in the state representation, making PE less crucial for tasks with simple dynamics and objectives based on immediate state-action relationships. The Half-Cheetah environment is characterized by *temporal coherence*, meaning the optimal actions do not vary wildly from one timestep to the next within a trajectory. Therefore, the absence of PE might not critically impact the model’s performance because the decision-making process does not depend heavily on the exact timestep but rather on the current state and the general phase of the run (e.g., starting, running at steady speed, ending).

**Hopper and Walker2d: Performance Deterioration without PE:** In contrast to HalfCheetah, the Hopper and Walker2d environments showed decreased performance without PE, with Hopper exhibiting a substantial drop. These environments have more complex dynamics and clear termination conditions if the agent falls or fails to maintain balance. The Hopper agent must learn to hop forward without toppling over, requiring precise timing and sequencing of actions, especially during critical phases like takeoff and landing. Without PE, the model may struggle to time these actions correctly, leading to falls and poor performance. Walker2d requires coordinating a bipedal agent to walk forward without falling. The Walker’s two legs provide more stability and recovery options than the Hopper’s single leg, making it less sensitive to the absence of PE, as there is a margin for error and less time-critical actions.

The performance discrepancy between Hopper, Walker2d, and HalfCheetah without PE arises from the varying importance of timing and sequence in these environments. The absence of PE disrupts the model’s ability to understand the temporal order of actions, which is critical for tasks where timing and order determine the success of the behaviour policy. The DT relies on the self-attention mechanism and PE to differentiate the positions of tokens in the input sequence, essential for learning and executing a temporally coherent policy in environments like Hopper and Walker2d. The reliance on PE for high-performance policies within these environments suggests potential generalizability and robustness concerns, as real-world applications with variable timing could pose challenges if a model’s performance hinges on an explicitly encoded temporal sequence.

**Input Embeddings and Hidden States:** In the HalfCheetah environment, the representations with (Fig. 1) and without (Fig. 2) PE exhibit a similar well-structured pattern, suggesting that the model’s performance is relatively robust to the absence of PE. This robustness can be attributed to the inherent dynamics of the HalfCheetah task, which involves continuous forward motion with a consistent gait pattern. The model effectively captures the underlying task structure even without explicit positional information. For Hopper and Walker2d, we observe a less structured embedding space without PE (Fig. 2). Hopper’s hidden states show the most pronounced mixing, corresponding to its significant performance drop due to the critical need for precise timing in actions to maintain balance. Walker2d exhibits mixed clusters but demonstrates a slightly more tolerant pattern to the absence of PE, aligning with its moderately better performance. With PE included (Fig. 1), the visualizations show clear, well-separated clusters, particularly in the deeper layers of the model. This indicates that the model is capturing temporal dependencies. The improved structuring in the representation space is crucial for Hopper and Walker2d, where proper sequencing of actions is vital for avoiding termination and achieving high performance.

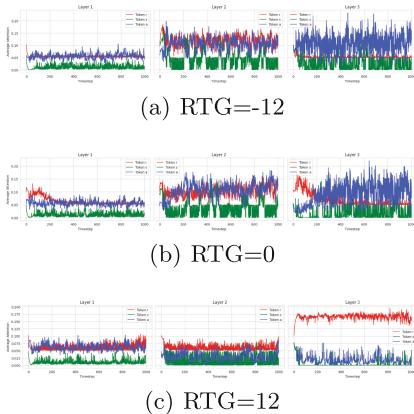
### 5.3 Return-to-Go (RTG) Analysis



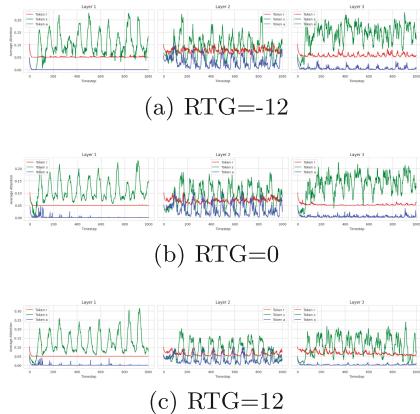
**Fig. 3.** Reward distribution for the datasets a) *HalfCheetah* b) *Hopper* c) *Walker2d*.

**Table 2.** Average Returns for Half-Cheetah DT Models for different Return-to-Go (RTG) values with and without Positional Encoding (PE)

Type	RTG	With PE	Without PE
Expert	-12	10234.47	8941.32
	0	10286.41	8962.81
	12	10900.58	9531.12
Medium	-12	4917.32	4732.50
	0	4910.84	4854.85
	12	4989.15	4864.70
Medium Replay	-12	-174.30	-42.6
	0	270.88	2786.47
	12	4141.28	3581.95



**Fig. 4.** Average attention for RTGs (red), States (green), and Actions (blue) tokens throughout the episode for different RTGs in HalfCheetah. X-axis: Timesteps (0–1000). Y-axis: Average attention (0–1).(Color figure online)



**Fig. 5.** Average attention for RTGs (red), States (green), and Actions (blue) tokens throughout the episode for different RTGs in Hopper. X-axis: Timesteps (0–1000). Y-axis: Average attention (0–1)(Color figure online)

In this section, we analyze the model’s behaviour across different RTG values for HalfCheetah. We chose RTG values of  $-12$ ,  $0$ , and  $12$  to represent a range from lower to higher expected returns. RTG represents the expected cumulative reward (we normalise it to smaller values for training stability of DT, i.e.  $12$ ) from a given state until the end of an episode. Our goal is to understand how the model’s behaviour patterns vary with RTG values and how this relates to the performance of different agent types, such as Expert, Medium, and Medium Replay. Theoretically, the DT agent should exhibit different performance levels for different input RTG values, even when trained solely on an expert dataset. The RTG acts as a "desired return" that guides the agent’s actions. By specifying different RTG values at test time, the agent should generate actions aimed at achieving returns close to those RTGs. But there are a few caveats: a) The distribution of RTG values in the expert dataset is (Figure 3(a)) skewed towards high values ( $12$ ) since expert trajectories tend to achieve high returns. This means the agent may not have seen many examples of low ( $-12$ ) or medium ( $0$ ) RTG values during training. b) When conditioned on an RTG value lower than what is commonly available in the expert data, the agent struggles to generate a precise trajectory that achieves the specified return. This is evident from Table 2, where the expert and the medium agent cannot generate lower average returns for low RTGs.

**RTG =  $-12$ :** When conditioned on an RTG of  $-12$ , which is lower than the typical returns observed in the expert dataset (Figure 3(a)), the agent’s attention mechanism displays a distinct pattern. As shown in Figure 4(a), the agent allocates a higher proportion of attention to the action tokens throughout the

episode, with consistently high attention weights for actions. This behaviour suggests that when faced with a lower target return that is not present in the dataset, the agent prioritizes immediate actions rather than heavily considering the long-term return signal or the current state. By emphasizing the action tokens, the agent appears to be optimizing for short-term rewards and progress, potentially sacrificing long-term planning in favour of reactive decision-making.

**RTG = 0:** When the RTG is set to 0, representing a neutral target return, the attention dynamics still exhibit similar behaviour to RTG -12 (Figure 4(b)). This can be explained by the dataset reward distribution (Figure 3(a)), where the expert dataset consists primarily of expert trajectories and high rewards, with only a small portion of lower rewards closer to 0. These lower rewards are predominantly from the initial timesteps when the cheetah begins running. During the initial timesteps (*timestep 0 to 200*), the cheetah pays more attention to the RTG tokens. This is presumably because these initial steps are very similar to those in the expert trajectory, where the returns are lower. Once running has commenced, RTG 0 does not provide any useful prompting information in line with expected environment dynamics, and attention reverts to actions once more, similar to the behaviour observed for RTG -12.

**RTG = 12:** In this scenario with RTG = 12, the typical return prompt for the expert dataset (Figure 3(a)), the attention weights for the RTG tokens become more prominent, and this is sustained throughout the episode (Figure 4(c)). This indicates that the agent is placing greater emphasis on the long-term return signal. This behaviour suggests that when faced with a higher target return, the agent adjusts its attention strategy to prioritize the long-term cumulative reward over immediate actions or state information. By allocating more attention to the RTG tokens, the agent generates trajectories that optimize for achieving the desired high return.

**Medium Replay DT:** For the medium replay agent, the different average rewards observed for varying RTG values can be attributed to the larger distribution of rewards present in the dataset (Figure 3(a)). With a more diverse set of reward signals, the agent can be more flexible in how it allocates attention between states, actions and RTGs. This results in different sets of behaviour based on different RTG values. We make another interesting observation: without PE, the DT achieves higher rewards for low RTGs (Table 2) and displays clustered embeddings, unlike the dispersed ones with PE. This indicates that removing PE reduces overfitting and temporal bias, promoting broader pattern learning instead of memorizing specific sequences when the dataset is diverse.

Figure 5 shows the behaviour of the Hopper model across different RTGs, highlighting a distinct attention pattern towards state tokens during hopping and forward motion phases. This pattern demonstrates the model’s adaptive attention allocation, consistent with our earlier findings on PE (Sect. 4.2), which indi-

cates that PE crucially affects the model’s behaviour due to its dependence on state sequences, resulting in lower rewards in the absence of PE for Hopper and Walker2d environments. Figures 4 and 5 demonstrate that despite consistent average returns across different RTG values, our attention visualization reveals significant variations in the DT’s internal workings. This illustrates how the model’s behaviour adapts to different RTG values, offering deeper insights into the dynamics of the DT’s decision-making.

#### 5.4 Action (Joint) Perturbations Analysis

**Table 3.** Comparative Analysis of Half Cheetah Joint Perturbation Effects on Average Rewards over 10 and 50 Timesteps With and Without Positional Encoding (PE): Starting at timestep 200

Joints	10 timesteps		50 timesteps	
	With PE	Without PE	With PE	Without PE
Random	9542.52	7671.35	3531.67	3102.05
Back Thigh	8977.05	7442.05	3277.47	2187.14
Back Shin	9369.50	8572.26	7902.85	6643.82
Back Foot	8251.29	6853.32	4337.66	2607.20
Front Thigh	10061.82	8797.21	6044.62	6575.85
Front Shin	8510.48	6728.59	3913.29	2693.94
Front Foot	10666.17	8847.09	8299.73	6101.98

We analyze the effects of joint perturbations only on the half-cheetah agent’s performance due to space constraints. Performance is measured by average rewards over 10 and 50 timestep perturbation intervals. Experiments in the HalfCheetah environment reveal DT’s resilience and limitations. Average rewards with PE are consistently lower than without, but short-term degradation (10 timesteps) is moderate as the attention mechanism retains information from previous timesteps since the context length is 20. However, long-term effects (50 timesteps) are more pronounced, indicating a reduced ability to compensate over time due to fewer recovery steps and limited context. Perturbations starting at timestep 100 result in lower average returns than those starting at timestep 200, as the cheetah requires 150–200 timesteps to reach and maintain high velocity. Table 3 shows a joint-specific impact on the agent’s locomotion. The back thigh, back foot and front shin result in the most significant decrease in average rewards, highlighting their pivotal role in the agent’s movement. This is consistent across both time intervals and PE conditions, suggesting that these actions(joints) are crucial for generating forward momentum and movement.

**Discussion:** Attention visualization reveals a drawback of DT in real-world continuous control tasks requiring multiple simultaneous actions. The current tokenization method combines all action tokens into a single embedding, reducing the interpretability and importance of individual actions. Our analysis indicates the need for different tokenization methods to create more interpretable transformer models for continuous control tasks.

## 6 Conclusion and Future Work

In this work, we analyzed Decision Transformers on continuous control tasks in the MuJoCo environment. We found that positional encoding is crucial for complex tasks, and models adapt their behaviour based on return-to-go (RTG) values, with expert and medium models showing less sensitivity to RTG due to their training datasets. Our embedding analysis revealed that the model learns hierarchical abstractions while attention visualizations provide insights into the role of attention mechanisms in guiding behaviour. Perturbation studies demonstrated the model’s robustness to input noise and the significance of specific joints for successful locomotion. These findings enhance our understanding of Decision Transformers and their potential applications and limitations in continuous control tasks. Future work should focus on interpreting training behaviours and performing detailed mechanistic interpretability to understand internal circuits and behaviours to develop more interpretable transformer-based architectures.

## References

1. Abnar, S., Zuidema, W.: Quantifying attention flow in transformers. In: Annual Meeting of the Association for Computational Linguistics (2020). <https://api.semanticscholar.org/CorpusID:218487351>
2. Alharin, A., Doan, T.N., Sartipi, M.: Reinforcement learning interpretation methods: a survey. IEEE Access **8**, 171058–171077 (2020)
3. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: an evaluation platform for general agents. J. Artif. Intell. Res. **47**, 253–279 (2013)
4. Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., Bruna, J.: When does return-conditioned supervised learning work for offline reinforcement learning? In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022). <https://openreview.net/forum?id=XByg4kotW5>
5. Bricken, T., et al.: Towards monosemanticity: decomposing language models with dictionary learning. Trans. Circ. Thread **2** (2023)
6. Brockman, G., et al.: Openai gym. arXiv preprint [arXiv:1606.01540](https://arxiv.org/abs/1606.01540) (2016)
7. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Adv. Neural. Inf. Process. Syst. **33**, 1877–1901 (2020)
8. Cammarata, N., et al.: Thread: circuits. Distill (2020). <https://doi.org/10.23915/distill.00024>, <https://distill.pub/2020/circuits>
9. Chen, L., et al.: Decision transformer: reinforcement learning via sequence modeling. Adv. Neural. Inf. Process. Syst. **34**, 15084–15097 (2021)

10. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: International Conference on Machine Learning, pp. 1282–1289. PMLR (2019)
11. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning (2017)
12. Elhage, N., et al.: A mathematical framework for transformer circuits. Trans. Circ. Thread (2021). <https://transformer-circuits.pub/2021/framework/index.html>
13. Emmons, S., Eysenbach, B., Kostrikov, I., Levine, S.: Rvs: what is essential for offline RL via supervised learning? In: International Conference on Learning Representations (2022). <https://openreview.net/forum?id=S874XAIPkR>
14. Erez, T., Tassa, Y., Todorov, E.: Infinite-horizon model predictive control for periodic tasks with contacts (2012)
15. Fu, J., Kumar, A., Nachum, O., Tucker, G., Levine, S.: D4rl: datasets for deep data-driven reinforcement learning. arXiv preprint [arXiv:2004.07219](https://arxiv.org/abs/2004.07219) (2020)
16. Glanois, C., et al.: A survey on interpretable reinforcement learning. Mach. Learn. 1–44 (2024)
17. Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., Bertsimas, D.: Finding neurons in a haystack: Case studies with sparse probing. arXiv preprint [arXiv:2305.01610](https://arxiv.org/abs/2305.01610) (2023)
18. Hilton, J., Cammarata, N., Carter, S., Goh, G., Olah, C.: Understanding rl vision. Distill 5(11), e29 (2020)
19. Joseph Bloom, P.C.: Decision transformer interpretability. <https://www.lesswrong.com/posts/bBuBDJBYHt39Q5zZy/decision-transformer-interpretability>, February 2023
20. Kenny, E.M., Tucker, M., Shah, J.: Towards interpretable deep reinforcement learning with human-friendly prototypes. In: The Eleventh International Conference on Learning Representations (2022)
21. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: tutorial, review, and perspectives on open problems. arXiv preprint [arXiv:2005.01643](https://arxiv.org/abs/2005.01643) (2020)
22. McGrath, T., et al.: Acquisition of chess knowledge in alphazero. Proc. Natl. Acad. Sci. 119(47), e2206625119 (2022)
23. Milani, S., Topin, N., Veloso, M., Fang, F.: Explainable reinforcement learning: a survey and comparative review. ACM Comput. Surv. (2023)
24. Nanda, N., Chan, L., Lieberum, T., Smith, J., Steinhardt, J.: Progress measures for grokking via mechanistic interpretability. arXiv preprint [arXiv:2301.05217](https://arxiv.org/abs/2301.05217) (2023)
25. Olah, C.: Mechanistic interpretability, variables, and the importance of interpretable bases. <https://www.transformer-circuits.pub/2022/mech-interp-essay/index.html> (2022), Accessed 25 Apr 2024
26. Prudencio, R.F., Maximo, M.R., Colombini, E.L.: A survey on offline reinforcement learning: taxonomy, review, and open problems. IEEE Trans. Neural Netw. Learn. Syst. (2023)
27. Raghu, A., Komorowski, M., Ahmed, I., Celi, L.A., Szolovits, P., Ghassemi, M.: Deep reinforcement learning for sepsis treatment. CoRR [abs/1711.09602](https://arxiv.org/abs/1711.09602) (2017), <http://arxiv.org/abs/1711.09602>
28. Todorov, E., Erez, T., Tassa, Y.: Mujoco: a physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033. IEEE (2012)
29. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)

30. Wawrzyński, P.: A cat-like robot real-time learning to run. In: Kolehmainen, M., Toivanen, P., Beliczynski, B. (eds.) ICANNGA 2009. LNCS, vol. 5495, pp. 380–390. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-04921-7\\_39](https://doi.org/10.1007/978-3-642-04921-7_39)
31. 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)
32. Wu, Y.H., Wang, X., Hamaya, M.: Elastic decision transformer. ArXiv [abs/2307.02484](https://arxiv.org/abs/2307.02484) (2023), <https://api.semanticscholar.org/CorpusID:259342857>
33. Yu, C., Liu, J., Nemati, S., Yin, G.: Reinforcement learning in healthcare: a survey. ACM Comput. Surv. **55**(1) (2021). <https://doi.org/10.1145/3477600>



# Flexible-Order Feature-Interaction for Mixed Continuous and Discrete Variables with Group-Level Interpretability

Zijie Zhai<sup>1</sup>, Junchen Shen<sup>1</sup>, Ping Li<sup>2</sup>, Jie Zhang<sup>3</sup>, and Kai Zhang<sup>1(✉)</sup>

<sup>1</sup> School of Computer Science and Technology, East China Normal University,  
Shanghai, China

{51265901041, 51215901048}@stu.ecnu.edu.cn, kzhang@cs.ecnu.edu.cn

<sup>2</sup> School of Computer Science and Software Engineering, Southwest Petroleum  
University, Chengdu, China

<sup>3</sup> Institute of Science and Technology for Brain Inspired Intelligence, Fudan  
University, Shanghai, China

**Abstract.** Deep neural networks have shown remarkable performance across diverse machine learning tasks. However, the balance between predictive accuracy and model interpretability remains a persistent challenge: high-performing models often exhibit complex structures defying human understanding, while interpretable (concise) models may sacrifice performance. In this paper, we show that feature interaction can be a crucial perspective when pursuing such balance, and propose flexible-order feature-interaction (FOFI), a new approach to exploit grouped feature interactions as the key to building accurate yet interpretable models. FOFI encourages local feature interactions that are organized into groups, which allows model capacity (parameters) to be distributed in a nuanced manner: at the lower granularity, dense interactions are restricted locally within each group to account for the complexity (performance); at the higher granularity, a flat predictive function is defined at group-level that guarantees the overall interpretability. Furthermore, FOFI is versatile in accommodating feature interactions of arbitrary order among mixed continuous and categorical variables. Extensive experiments on both simulated and real-world datasets showcase the encouraging performance and interpretability of FOFI.

**Keywords:** Feature Interaction · Interpretability · Neural Networks

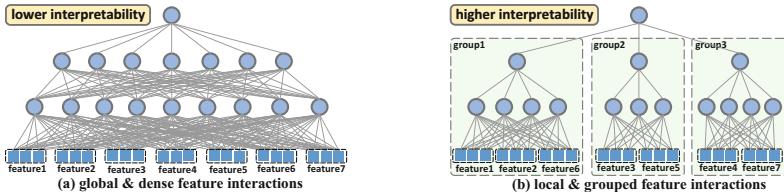
## 1 Introduction

Deep neural networks have demonstrated cutting-edge performance across diverse application domains. However, an inherent conflict still exists between performance and model interpretability. On the one hand, high-performing models are typically with intricate structures that are challenging for humans to

interpret. For example, in a feed-forward neural network, the input features are coupled globally with each other through layers of dense connections, making it non-trivial to understand how the model arrives at its decisions. On the other hand, concise models can be easy to understand but likely with compromised accuracy. For example, in a linear regression model, the coefficients provide insights into the relationships between the input features and the output. Yet its capacity is limited for difficult problems.

The goal of this paper is to develop deep networks for **tabular data** that are both accurate and interpretable. Interpretability can be defined in many ways, such as rule-based and post-hoc methods (see [5, 20, 26] for a review). Our primary focus is on model interpretability, particularly on how the model structure elucidates the relationships between variables and the mappings from input to output. Other types of interpretability will be studied in our future research. **Our main idea is to balance model performance and interpretability by exploring feature interactions**, which we explain from two angles.

*First, Feature Interactions Greatly Affect Model Interpretability.* As illustrated in Fig. 1, (a) is a multi-layer perceptron with globally coupled input features, making its decision process opaque. In contrast, (b) has a modular structure with dense interactions confined to three local groups at the bottom, and a linear combination at the top. This design is more interpretable, offering clear explanations of feature interactions at the group level, thus allowing for concise and comprehensible models.



**Fig. 1.** The landscape of feature interactions largely impacts model interpretability. (a) A network with globally dense interactions is hard to interpret. (b) A network with constrained, local group interactions is easier to understand at the group level. We aim to achieve this grouping while maintaining performances.

*Second, Feature Interactions Greatly Affect Predictive Performance.* Neural networks excel as strong learners due to their deep and intricate model structures that foster rich nonlinear feature couplings [5, 8, 25]. Analogous to real-world situations, diverse features may act as components within a complex system, naturally intertwining to influence the system’s functionality or output<sup>1</sup>. Therefore, preserving useful and sufficient amount of feature interactions is essential for accurate predictions.

<sup>1</sup> A simple example of feature interaction is that a high BMI (weight/height<sup>2</sup>) indicates an increased risk of heart disease, showcasing a pairwise interaction {height, weight}.

To summarize, balancing model interpretability and performance involves managing feature interactions: we aim to maintain enough nonlinear interactions to ensure model capacity while avoiding excessive, unorganized interactions that reduce interpretability.

To balance accuracy and interpretability, we propose **flexible-order feature-interaction (FOFI)**, an innovative approach that leverages structured feature interactions for creating precise yet interpretable models. The FOFI network organizes model capacity across feature groups: dense interactions are localized within groups to enhance performance, while a lighter, group-level predictive function enhances interpretability. The FOFI-architecture is shown in Fig. 2 (detailed in Sect. 3). Key innovations/advantages are highlighted below:

- **Adaptive binning of continuous features.** Obtaining low-dimensional embeddings for continuous features remains challenging. We proposed a learnable quantization and embedding scheme for continuous variables that captures their distribution and systematically models interactions among mixed feature types.
- **Flexible order of feature interactions.** Identifying high-order feature interactions is notoriously hard. We solve it by optimizing a probabilistic Interaction-Indicator-Matrix (IIM) to encode feature interactions of arbitrary orders, and using it as a mask to promote intra-group feature interactions while pruning inter-group ones. An annealing competition scheme is devised to improve learning of the IIM.
- **Being accurate and also interpretable.** We build an end-to-end framework that coordinates feature quantization, embedding, and interaction. Extensive experiments on toy and real-world datasets demonstrate that FOFI achieves competitive accuracy and offers group-level interpretability.

The rest of the paper is organized as follows. Section 2 reviews related work of feature interaction and interpretability. Section 3 introduces the proposed FOFI-network. Section 4 reports experimental results and last section concludes the paper.

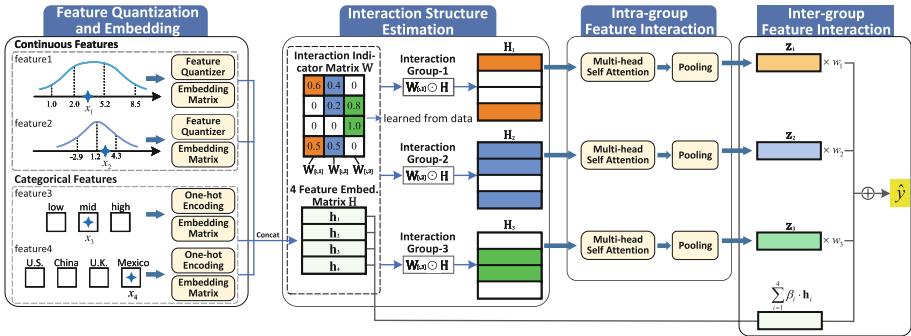
## 2 Related Work

### 2.1 Feature Interaction

Feature interaction is the complex collaborative effects of features whose modelling can be crucial for the performance of learning algorithms. Factorization machine [21] is one of the earliest models that considers interactions between every pair of features. Recent years, efforts have been put into high-order feature interactions. For example, DeepFM [10] combines Factorization Machines with DNN to jointly learn high-order and low-order feature interactions. [29] use a cross-product operation and a DNN to build high-order feature interactions. Recently, attention-based strategies were introduced. For example, [25] employs self-attention among the input features to model their interactions; [8]

stacks multiple layers of Transformer blocks in tabular data learning tasks; [32] constructs a relation graph to encode feature interactions in a Transformer.

These methods have shown that sophisticated feature interactions may improve performance. However, the feature interactions that are enforced are global and dense, which leads to lower model interpretability. In comparison, we aim to recover well-organized feature interactions that are useful while easy to interpret.



**Fig. 2.** The FOFI network on a toy data with  $k = 4$  features (two categorical, two continuous) and  $p = 3$  feature-interaction groups (encoded by the three columns of Interaction-Indicator Matrix  $\mathbf{W}$ ). Main building blocks: (1) Feature quantization and embedding; (2) Interaction structure estimation; (3) Intra-group feature interaction; (4) Inter-group feature interaction and prediction.  $\odot$  is entry-wise product with a broadcast (Eq. 7) and  $\oplus$  denotes vector addition.

## 2.2 Interpretability

Interpretability in machine learning can be defined and implemented in many different ways. Post-hoc methods analyze trained models to uncover feature interactions [6, 26, 27]; Surrogate models use simpler models to approximate complex models for easier explanations [13, 22, 24]. However, these post-hoc explanations might have a gap between the surrogate model and the fitted complex model, potentially impacting scalability. Other studies focus on the interpretability of model structures, aiming to construct models that are easier to understand by humans. Recently, rule-based models and logical neural networks [19, 30, 31] have gained interest for their interpretability. These models use logical operators instead of linear algebra computations within the networks, enhancing interpretability. Training logical neural networks is nontrivial due to non-differentiability of parameters after discretization [11, 30]. Furthermore, these models capture internal feature interactions through the enumeration of numerous predictive paths.

Similar to our motivation, TabSRA [1] introduces a “self-reinforcement” attention mechanism to explain the contribution of features to the final output. However, TabSRA’s model interpretability is primarily based on individual or second-order features. In this paper, we aim to uncover informative patterns of arbitrary-order feature interactions from a global perspective.

### 3 Method

We consider supervised classification and regression on a dataset with input-output pairs in the form of  $(\mathbf{x}, y)$ , where the input sample  $\mathbf{x} = [x_1, \dots, x_k]$  has  $k$  features, and  $y$  is the associated label. We consider two types of input features:

- if the  $i$ th feature is continuous (denoted by  $\mathcal{N}_i$ ),  $x_i$  will be a real scalar;
- if the  $i$ th feature is categorical (denoted by  $\mathcal{C}_i$ ),  $x_i$  will take values from  $c_i$  discrete integers, where  $c_i = |\mathcal{C}_i|$  is called the granularity of feature  $\mathcal{C}_i$ .

In this paper, subscript  $i$  always denotes the feature index, subscript  $j$  will denote the feature-group index, and sample index will be denoted by superscript  $0$ .

As shown in Fig. 2, FOFI network consists of four components : (1) First, all the  $k$  features of a sample are mapped to a low-dimensional continuous space through the feature quantization and embedding module; (2) Then an annealed competition scheme recovers feature interaction groupings via the interaction structure estimation module; (3) Intra-group feature interaction module enables flexible-order and nonlinear interactions within groups using self-attention; (4) Finally, inter-group interaction module aggregates group-level features representations with residual connections to input features for prediction.

#### 3.1 End-to-End Feature Quantizer and Embedding

In order to model the interaction between features of different types (continuous, categorial), it is desirable to embed the features as low-dimensional vectors in a shared Euclidian space.

Categorical features are easy to embed as vectors. Let the  $i$ th categorical feature  $\mathcal{C}_i$  have  $c_i$  discrete values ( $c_i = |\mathcal{C}_i|$ ). Then a feature value  $x_i$  can be encoded as a one-hot row-vector  $\tilde{\mathbf{x}}_i \in \mathbb{R}^{1 \times c_i}$  specifying which of the  $c_i$  values is taken in  $x_i$ . We can then embed the discrete value  $x_i$  of  $\mathcal{C}_i$  as

$$\mathbf{h}_i = \tilde{\mathbf{x}}_i \mathbf{E}_i. \quad (1)$$

where  $\mathbf{E}_i \in \mathbb{R}^{c_i \times d}$  is the embedding matrix corresponding to feature  $\mathcal{C}_i$ , with  $d$  being the embedding dimension.

Continuous features are more difficult to turn to low-dimensional embedding because infinitely many values may exist. Current methods typically quantize a continuous feature into a few bins and then treat it as a discrete variable in an offline fashion [7,31]. In this paper, we instead design an **adaptive and end-to-end optimized** binning-and-embedding strategy to produce discriminative

**Algorithm 1.** Quantizer for the  $i$ th continuous feature  $\mathcal{N}_i$ 


---

**Input:** number of bins  $c_i$ , set of all the instances in  $\mathcal{N}_i$ , denoted by  $X_i = \{x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)}\}$ , with  $n$  the training sample size; feature value for a sample of interest  $x_i^{(a)}$

**Output:** soft probability encoding  $\tilde{\mathbf{x}}_i \in \mathbb{R}^{1 \times c_i}$  for  $x_i^{(a)}$

- 1:  $[s_1, s_2, \dots, s_{c_i}] \leftarrow k\text{-means}(X_i, c_i)$  // initialize bin centers
- 2: **for**  $u = 2$  to  $c_i$  **do**
- 3:    $m_u \leftarrow (s_u + s_{u-1})/2$  // update the left and right boundaries for each bin
- 4: **end for**
- 5: **for**  $u = 2$  to  $c_i - 1$  **do**
- 6:    $\sigma_u \leftarrow (m_{u+1} - m_u)/6$  // update Gaussian kernel bandwidth for each bin
- 7: **end for**
- 8:  $\sigma_1 \leftarrow \frac{2}{6}(m_2 - s_1)$ ,  $\sigma_{c_i} \leftarrow \frac{2}{6}(s_{c_i} - m_{c_i-1})$  // bandwidth for the left- and right-most bin
- 9: **for**  $u = 1$  to  $c_i$  **do**
- 10:    $\pi_u \leftarrow \sum_{l=1}^n \frac{\exp(x_i^{(l)} - s_u)^2 / 2\sigma_u^2}{\sum_{v=1}^{c_i} \exp(-(x_i^{(l)} - s_v)^2 / 2\sigma_v^2)}$  // bin weight
- 11:    $\tilde{\mathbf{x}}_i[u] \leftarrow (\pi_u/n) \cdot \exp(-(x_i^{(a)} - s_u)^2 / 2\sigma_u^2)$  // prob. of  $x_i$  belonging to  $u$ -th bins
- 12: **end for**
- 13:  $\tilde{\mathbf{x}}_i \leftarrow \ell_1\text{-normalization}(\tilde{\mathbf{x}}_i)$
- 14: **return**  $\tilde{\mathbf{x}}_i$

---

embeddings for continuous features for the learning task while simultaneously taking into account the distribution of the feature.

(1) We first partition the  $i$ th continuous feature  $\mathcal{N}_i$  into  $c_i$  bins. In order to define an easily optimizable binning strategy, we resort to learning the placement of  $c_i$  “bin-centers”  $\{s_1, s_2, \dots, s_{c_i}\}$ . By assigning every value to the closest bin-center, the real axis naturally breaks into  $c_i$  bins, whose boundaries are located at the midpoint between every two adjacent bin-centers. As shown in Fig. 3, the left-boundary ( $m_u$ ) and right boundary ( $m_{u+1}$ ) of the  $u$ th bin can be recovered by the bin centers

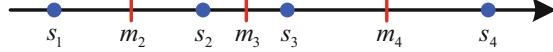
$$m_u = \frac{s_u + s_{u-1}}{2}, \quad u = 2, 3, \dots, c_i. \quad (2)$$

The use of the bin-centers makes it convenient to assign  $x_i \in \mathbb{R}$  (the value of the  $i$ th continuous feature  $\mathcal{N}_i$ ) to the  $c_i$  bins softly in the form of a probability vector  $\tilde{\mathbf{x}}_i \in \mathbb{R}^{1 \times c_i}$

$$\tilde{\mathbf{x}}_i[u] = \frac{\exp\left(-\frac{(x_i - s_u)^2}{2\sigma_u^2}\right)}{\sum_{v=1}^{c_i} \exp\left(-\frac{(x_i - s_v)^2}{2\sigma_v^2}\right)}, \quad u = 1, 2, \dots, c_i. \quad (3)$$

The bandwidth  $\sigma_u$  affects the computation of soft probabilities for the  $u$ th bin. One can either freely optimize it or make it commensurate with the width of the  $u$ th bin by setting  $6\sigma_u$  (span of a Gaussian) equal to estimated width of the  $u$ th bin, i.e.,  $m_{u+1} - m_u$ . Empirically, choosing  $\sigma_u = (m_{u+1} - m_u)/6$  is more robust.

(2) We then compute the embedding for (any value of)  $i$ th continuous feature  $x_i$  as a linear combination of  $c_i$  bin representations, forming an “embedding



**Fig. 3.** Illustration of bin-centers  $s_u$ 's and boundaries  $m_u$ 's in feature quantizer.

matrix”  $\mathbf{E}_i \in \mathbb{R}^{c_i \times d}$  where each row represents a bin’s embedding. Based on the soft probability vector  $\tilde{\mathbf{x}}_i$  (Eq. 3) specifying which bin  $\mathbf{x}_i$  most likely belongs to, we can perform a linear combination of the rows in  $\mathbf{E}_i$  using  $\tilde{\mathbf{x}}_i$  as weights, i.e.,  $\mathbf{h}_i = \tilde{\mathbf{x}}_i \mathbf{E}_i$ . This is similar to Eq. 1, but here  $\tilde{\mathbf{x}}_i$  is a soft probability instead of one-hot vector, preserving the distribution of continuous features and avoiding the information loss of discretization.

The number of bins for each continuous feature is pre-determined; empirically, when it is larger than 8–10, the performance will remain stable. The bin centers  $s_u$ 's were initialized using a univariate  $k$ -means, which places more bins around densely populated regions in the real-axis. The procedures are detailed in Algorithm 1.

After each feature in the input sample  $\mathbf{x} = [x_1, x_2, \dots, x_k]$  has been turned to low-dimensional row vectors  $\mathbf{h}_i$ 's, we will concatenate them together as the new representation of  $\mathbf{x}$ , as

$$\mathbf{H}^\top = [\mathbf{h}_1^\top, \mathbf{h}_2^\top, \dots, \mathbf{h}_k^\top], \quad (\mathbf{H} \in \mathbb{R}^{k \times d}) \quad (4)$$

which can be used for feature interaction modeling.

### 3.2 Feature Interaction Group Estimation

Identifying feature interactions of arbitrary order is challenging due to the combinatorial nature. To avoid this difficulty, we treat it as continuous optimization by learning a probabilistic Interaction Indicator Matrix (IIM) denoted by  $\mathbf{W} \in \mathbb{R}^{k \times p}$ , which encodes how the  $k$  features interact with each other to form  $p$  interaction groups. This encoding scheme provides great flexibility in identifying feature interactions with desired structural preferences (grouping).

We randomly initialize a learnable matrix  $\mathbf{W}$  and then normalize each row with softmax operator,

$$\mathbf{W}_{ij} = \frac{\exp(\mathbf{W}_{ij}/\tau)}{\sum_m \exp(\mathbf{W}_{im}/\tau)} \quad (5)$$

Here  $\tau$  is the temperature parameter, and  $\mathbf{W}_{ij}$  specifies the probability that the  $i$ th feature joins the  $j$ th group to interact with the features in that group. In order to promote the local grouping of feature interactions, the indicator matrix  $\mathbf{W}$  is preferably sparse and with little overlaps among its columns. However, learning such a structure can be difficult considering the vast combinatorial possibilities among the features.

To solve this, we propose an annealed competition scheme to learn  $\mathbf{W}$  that preserves faithful and well-constrained interactions. The key idea is to start from

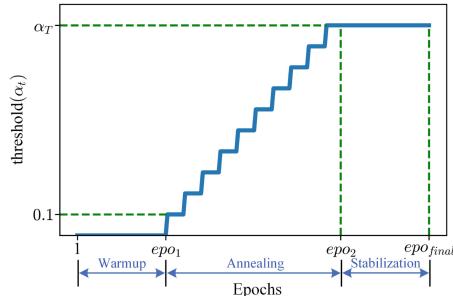
a dense  $\mathbf{W}$  and progressively truncate its entries throughout the learning process until approaching desired level of sparsity. The truncation is defined as

$$\mathbf{W}_{ij} \leftarrow \begin{cases} 0, & \text{if } \mathbf{W}_{ij} < \alpha_t \cdot \text{avg}(\mathbf{W}_{[:,j]}) \\ \mathbf{W}_{ij}, & \text{if } \mathbf{W}_{ij} \geq \alpha_t \cdot \text{avg}(\mathbf{W}_{[:,j]}) \end{cases} \quad (6)$$

where  $\alpha_t$  is the threshold for the  $t$ -th epoch, and  $\text{avg}(\mathbf{W}_{[:,j]})$  is column-wise average of  $\mathbf{W}_{[:,j]}$ . The column-wise thresholding (Eq. 6) enforces a competition for features to enter the groups: if  $\mathbf{W}_{ij}$  ranks poorly in the  $j$ th column, the  $i$ th feature will be removed from the  $j$ th feature-group. To make the truncation process more robust, we design a three-stage approach with varying thresholds  $\alpha_t$ , as shown in Fig. 4:

1. *Warmup*. In this stage, the threshold  $\alpha_t = 0$  and so the main driving force to shape  $\mathbf{W}$  is the minimization of loss function.
2. *Annealing*. In this stage,  $\alpha_t$  begins to increase for every  $l$  epochs until reaching  $\alpha_T$ . Therefore, the model is obliged to gradually reduce the number of active features in each group when minimizing the loss function.
3. *Stabilization*. After approaching desired level of truncation, some epochs are used to stabilize the training.

The annealed competition allows us to effectively refine the interaction map in the training process: irrelevant features are gradually removed, so that useful features finally dominate in each interaction-group. Using the (sparse) Interaction Indicator Matrix  $\mathbf{W}$  as mask on the learned feature representation  $\mathbf{H}$  (Eq. 4), we can conveniently pick out active features inside the same group to enforce their local interactions.



**Fig. 4.** Three-stage thresholding of Interaction Indicator Matrix  $\mathbf{W}$  (Eq. 6).

### 3.3 Intra and Inter-Group Interaction

To guarantee model interpretability, we consider feature interactions at intra-group level and inter-group level, separately.

**For intra-group (local) feature interactions**, we use self-attention [28] to promote nonlinear interactions among the features in the same group. Specifically, the feature embedding matrix  $\mathbf{H}$  is masked by the  $j$ th column of the sparse indicator matrix  $\mathbf{W}$ , as

$$\mathbf{H}_j = \mathbf{W}_{[:,j]} \odot \mathbf{H}, \quad (7)$$

where  $\odot$  indicates that each row of  $\mathbf{H}$  is re-weighted by the corresponding entry in the column-vector  $\mathbf{W}_{[:,j]}$ . Hence  $\mathbf{H}_j$  can be quite sparse since only those rows corresponding to active features of the group are not masked by zeros. We therefore shrink  $\mathbf{H}_j$  as

$$\mathbf{H}_j \leftarrow \mathcal{S}(\mathbf{H}_j), \quad (8)$$

where  $\mathcal{S}(\cdot)$  removes the masked rows in  $\mathbf{H}_j$  and only stacks the active rows to form the group-wise feature matrix  $\mathbf{H}_j \in \mathbb{R}^{k_j \times d}$ , with  $k_j$  the number of active features in  $j$ th feature group. This truncation effectively reduces the computational complexity of subsequent self-attention calculations.

Based on the embeddings of the active features in the  $j$ th group ( $\mathbf{H}_j$ ), we use multi-head self-attention with residual connections to model the intra-group feature interactions within  $j$ th feature group as

$$\mathbf{Q}_j = \mathbf{H}_j \mathbf{W}_j^q, \quad \mathbf{K}_j = \mathbf{H}_j \mathbf{W}_j^k, \quad \mathbf{V}_j = \mathbf{H}_j \mathbf{W}_j^v, \quad (9)$$

$$\text{Att}_j(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\mathbf{Q}_j \mathbf{K}_j^\top / \sqrt{d'}\right) \mathbf{V}_j, \quad (10)$$

$$\text{Att}_j^{\text{full}} = \text{concat}\left[\text{Att}_j^{\text{head-1}}, \text{Att}_j^{\text{head-2}}, \dots, \text{Att}_j^{\text{head-h}}\right], \quad (11)$$

$$\tilde{\mathbf{H}}_j = \text{relu}\left(\text{Att}_j^{\text{full}} + \mathbf{H}_j \mathbf{W}_{\text{res}}\right), \quad (12)$$

where  $\mathbf{W}_j^q, \mathbf{W}_j^k, \mathbf{W}_j^v \in \mathbb{R}^{d \times d'}$  are transform matrices for the queries, keys, and values.  $\mathbf{W}_{\text{res}} \in \mathbb{R}^{d \times hd'}$  is the residual transformation matrix.

Here, we use self-attention to encourage flexible and nonlinear coupling among the features falling in the same interaction-group. As a result, each interaction-group contains a subset of the input features with unique composition and flexible interaction-patterns, which is essential for building powerful local predictors.

We further compress  $\tilde{\mathbf{H}}_j$  by average-pooling of its rows as

$$\mathbf{z}_j = \text{Avg-Pool}(\tilde{\mathbf{H}}_j) \text{ for } j = 1, 2, \dots, p. \quad (13)$$

**For inter-group interactions**, having obtained the compact representation  $\mathbf{z}_j$ 's for each of the  $p$  feature-interaction groups (Eq. 13) and also the embedding representation  $\mathbf{h}_j$ 's for each individual feature (Eq. 4), we can then use a flat linear model to aggregate all their information to perform the final prediction

$$\mathbf{z} = \text{relu}\left(\sum_{j=1}^p w_j \cdot \mathbf{z}_j + \sum_{i=1}^k \beta_i \cdot \mathbf{h}_i\right), \quad \hat{y} = \text{FC}(\mathbf{z}). \quad (14)$$

Here, the coefficient  $w_j$ 's and  $\beta_i$ 's are all scalars and can be easily interpreted as the importance of each feature-interaction group and each individual feature. The predicted class label  $\hat{y}$  is obtained by applying fully-connected layer on top of  $\mathbf{z}$ , with Cross-Entropy or mean squared error loss function.

## 4 Experiments

We explore two main research questions with experimental studies:

**RQ1:** Is FOFI accurate on benchmark classification/regression data?

**RQ2:** Does FOFI identify meaningful feature interactions?

We introduce experimental setup in Sect. 4.1. Then we report the results of FOFI on classification/regression tasks (Sect. 4.2) and recovering feature interaction groups (Sect. 4.3). Ablation studies are in Sect. 4.4.

### 4.1 Experimental Setting

We collected 10 classification and 4 regression benchmark datasets widely used in the literature in testing the performance and interpretability of learning algorithms [2, 9, 15], including Adult Income (adult), Bank Marketing (bankm), Telco Customer Churn (telco), Churn Modelling (churnm), Cardiovascular Disease (cardio), Buddy, Connect-4, Nursery, Banknote, Wine, Cpu activity (cpu-act), California Housing (CA housing), House-16H and Diamonds, see the left part

**Table 1.** Dataset statistics (shown in left part), #con. and #cat. denote the number of continuous and categorical features, respectively. Number of feature-groups and average group size in experiments (shown in right part).

Dataset	#instance	#con.	#cat.	#classes	avg. group size	#groups ( $p$ )	$\alpha_T$
adult	32561	6	8	2	3.50	20	1.0
bankm	45211	7	9	2	3.81	16	1.0
cardio	70000	5	6	2	2.91	11	0.8
telco	7032	3	16	2	4.26	19	0.8
buddy	18834	4	5	3	2.67	15	1.0
churnm	10000	9	1	2	2.70	10	0.8
connect-4	67557	0	42	3	6.87	30	1.0
nursery	12960	0	8	5	2.00	8	0.8
banknote	1372	4	0	2	1.75	4	0.8
wine	178	13	0	3	3.23	13	0.8
cpu-act	8192	21	0	regression	4.71	21	1.0
CA housing	20640	8	0	regression	2.67	15	1.0
house-16H	22784	16	0	regression	3.31	16	1.0
diamonds	53940	6	3	regression	3.00	9	1.0

of Table 1 for details. More than half of the dataset have mixed categorical and continuous features. Additionally, we have designed 3 synthetic toy datasets to investigate the capability of FOFI to recover feature interactions, as in Table 3.

Our codes were written in PyTorch. For FOFI, the embedding dimension is  $d = 32$ ; the temperature in Eq. 5 is  $\tau = 0.1$ ; the number of attention layers is two with two attention heads. We use AdamW [14] optimizer and train FOFI for 250 epochs, with 20/80/150 epochs for the warm-up/annealing/stabilization stages in estimating the interaction matrix, and the threshold  $\alpha_t$  increases every 10 epochs during the annealing stage until reaching  $\alpha_T$ . The number of bins for continuous features is 8. The threshold  $\alpha_T$  is chosen from {0.8, 1.0}; the number of feature groups  $p$  is chosen from the integers in  $[\lceil k/2 \rceil, k + 6]$ .

## 4.2 Classification and Regression Tasks (RQ1)

**Metrics and Baselines.** Following [31], we use F1-score (macro) as the metric for classification tasks due to imbalanced datasets. Following [8], we use root mean square error (RMSE) for regression tasks and we apply standardization to regression targets. A 5-fold cross-validation was used to ensure a fair assessment of performance. Following [31], we chose a wide range of competing

**Table 2.** 5-fold cross-validated F1 score (%) (classification) and RMSE (regression) of 10 competing models on 14 real-world datasets (t-test with  $p < 0.01$ ). Some baselines either do not support or are not designed for regression tasks, so we separately report the model’s performance on classification (with avg. rank) and regression benchmarks. The best result for each dataset is underlined.

Dataset	FOFI	XGB	CatB	RRL	CRS	LR	SVM	RF	AutoInt	FT-T	T2G-F
adult	80.59	80.64	80.43	80.42	<u>80.95</u>	78.21	63.63	79.22	79.48	79.69	80.37
bankm	77.08	74.71	74.81	<u>77.18</u>	73.34	70.07	72.99	72.67	76.11	77.04	77.12
cardio	<u>73.84</u>	73.65	73.61	73.42	71.21	72.79	73.20	73.53	73.48	73.61	73.74
telco	<u>74.03</u>	72.01	71.84	72.87	69.40	70.71	70.68	71.07	72.75	73.08	73.93
buddy	<u>90.49</u>	89.55	90.07	89.98	86.39	85.72	89.01	89.70	89.14	89.41	90.13
churnm	<u>76.30</u>	75.17	75.50	75.24	73.89	63.38	71.82	72.95	73.21	75.61	75.86
connect-4	72.59	70.65	70.81	72.01	65.88	49.87	69.85	62.72	70.44	72.45	<u>73.64</u>
banknote	<u>100.0</u>	99.55	99.63	<u>100.0</u>	94.93	98.82	<u>100.0</u>	99.40	<u>100.0</u>	99.93	99.27
wine	<u>99.17</u>	97.78	97.16	98.37	97.78	95.27	96.05	98.31	94.51	94.63	94.51
nursery	<u>99.98</u>	99.94	99.41	99.76	99.69	69.63	83.87	97.94	99.18	98.96	99.73
Avg. rank	<u>1.5</u>	4.9	5.1	3.7	7.7	10.2	8.4	7.7	6.6	5.3	4.0
cpu-act	<u>0.144</u>	0.157	0.151	—	—	0.524	—	0.182	0.159	0.149	0.151
CA housing	<u>0.452</u>	0.465	0.457	—	—	0.650	—	0.479	0.471	0.462	0.454
house-16H	0.648	0.664	0.676	—	—	0.828	—	0.708	0.652	0.648	<u>0.635</u>
diamonds	<u>0.138</u>	0.146	0.156	—	—	0.496	—	0.174	0.162	0.149	0.157

methods: traditional ML models such as logistic/linear regression (LR) [12] for classification/regression and SVM [23]; decision-tree-based models such as random forest (RF) [3], XGBoost (XGB) [4] and CatBoost (CatB) [18]; rule-based models such as RRL [31] and CRS [30]; and powerful DNN-based models such as AutoInt [25], FT-T [8], and T2G-F [32]. All hyper-parameters of FOFI and baselines were chosen by using 5% of the training data as validation set in each of the 5-fold cross-validation round, aligning with [31]. For T2G-F, we follow the hyper-parameter tuning strategy in their paper [32]. For all the rest competing methods, we follow the hyper-parameter tuning strategy as in [31].

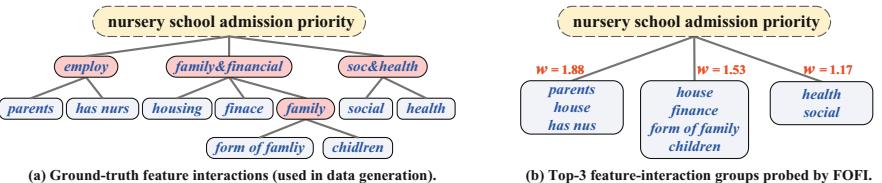
**Main Results.** The classification and regression results are reported in Table 2. The two-tailed Student’s t-test (with  $p < 0.01$ ) is used for significance testing. We observed that FOFI achieves the best results in 10 out of 14 datasets and also ranks highest on average. The average size of the feature-interaction group is shown in the right part of Table 1. For most datasets, the order-of-interaction is around 3–6, which can be easy for human to draw insights from. The impact of the hyper-parameters will be analyzed in Sect. 4.4.

### 4.3 Feature Interaction Recovery (RQ2)

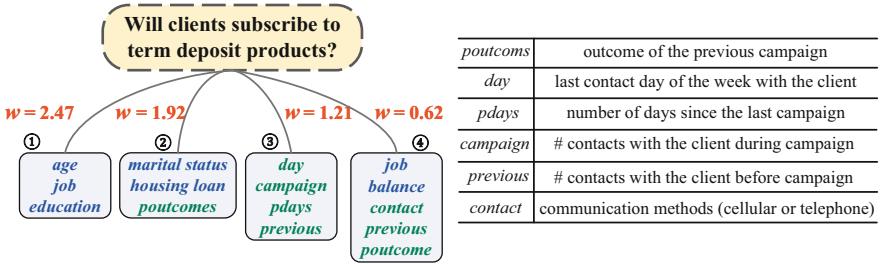
**Examples on Benchmark Datasets.** To investigate whether FOFI identifies meaningful feature interactions, we conducted case studies on the nursery and bank marketing dataset. The nursery dataset aims to prioritize nursery applications using 8 features related to family background and health conditions, while the bank marketing dataset predicts client subscription to term deposits after bank phone calls using 16 features (8 client-related and 8 campaign-related).

The nursery dataset is synthesized with known feature interactions [16]. In Fig. 5, we compare the ground-truth interactions and those identified by FOFI. Impressively, FOFI identified three feature interaction groups that closely match the three primary branches of the ground truth, achieving 88.9% overlap.

The bank-marketing dataset is a real-world dataset for which ground-truth feature interactions are unknown. Nevertheless, we still find reasonable feature-interaction groups identified by FOFI as in Fig. 6, with interesting observations.



**Fig. 5.** Case study on nursery dataset. (a) Ground truth feature interactions (blue: features that exist in the dataset; red: “latent” features generated during synthesis process). (b) Top-3 feature-interaction groups identified by FOFI. (Color figure online)



**Fig. 6.** The FOFI network identifies some key feature-interaction groups on the bank marketing dataset (blue: client features; green: campaign-related features). (Color figure online)

First, the most important features (group 1) are those related to clients' background, with a score  $w = 2.47$ ; while campaign-related features (group 3) are also predictive but with lower score  $w = 1.21$ . Second, despite the diversity, the features show a pattern across four groups; groups 1 and 3 focus on client and campaign features, respectively, whereas groups 2 and 4 mix both, uncovering potential hidden interactions. For instance, group 2 combines marital status, housing loans, and previous campaign outcomes, indicating their combined influence on predictions.

**Simulated Data.** To validate FOFI's ability to identify real feature interactions, we created three synthetic datasets listed in Table 3 with known ground truth interactions ( $\mathcal{G}$ ). Following [26], features  $x_i$  are uniformly drawn from  $[-1, 1]$ , and outputs  $y$  are converted to binary labels using the median of  $y$  as the threshold.

We evaluate the capacity of FOFI in recovering the desired feature interactions by computing a *feature interaction recovery score* as follows. For each ground-truth feature interaction set  $g_i \in \mathcal{G}$ , we examine the feature-group identified by FOFI that has the maximum overlap with  $g_i$ , and compute the Jaccard similarity between the two sets. A similarity smaller than 0.5 will be truncated to 0. Then we report the averaged similarities for all the ground-truth interaction groups as the final recovery score. We also considered two competing methods NID [26] and SIAN-FIS [6], both capable of revealing the interactions between features by post-hoc analysis on top of a trained DNN model.

Table 3 shows the results of all competing methods in recovering correct feature interactions. FOFI achieves the highest recovery scores, typically over 80%, identifying a significant portion of real feature interactions. Unlike post-hoc analysis, FOFI detects these interactions directly during the learning process.

**Table 3.** Feature interaction recovery scores  $R$  (%) on three synthetic datasets.

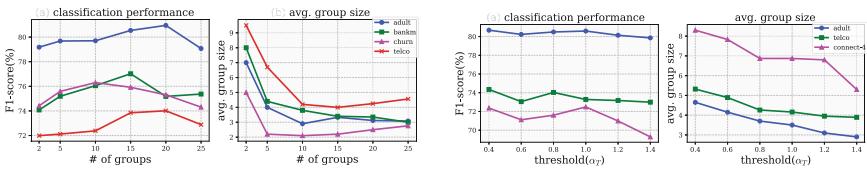
Synthetic functions	$R$ (%)		
	FOFI	NID	SIAN-FIS
$y = \exp x_1 - x_2  +  x_2x_3  - x_3^{2 x_4 } + (x_1x_4)^2 + \log(\sum_{i=4}^7 x_i^2)$	85.33	80.71	83.33
$y = 1/(1 + \sum_{i=1}^3 x_i^2) + \sqrt{\exp(x_4 + x_5) +  x_6x_7  + x_8x_9x_{10}}$	83.33	80.14	80.41
$y = \tanh(x_1x_2 + x_3x_4) + \exp(x_5x_6) + \log((x_2x_5x_7) + 1) + x_7x_8$	81.25	66.67	75.00

**Table 4.** Ablation results of adaptive feature quantization. w/o-quantizer: offline quantization (no adaptive feature quantizer).

	adult	bankm	cardio	telco	buddy	churnm
FOFI	80.59	77.08	73.84	74.03	90.49	76.30
w/o-quantizer	79.77	75.87	73.20	72.84	89.19	75.08

#### 4.4 Ablation Study

**Effect of Model Width.** The number of feature-interaction groups,  $p$ , controls the width of the FOFI-network. Here we study how it affects the model performance. As shown in Fig. 7a, the optimal number of feature groups for many datasets falls within  $p \in [10, 20]$ . Increasing group numbers leads to fewer features falling in each group, indicating FOFI’s ability to adjust intra-group interaction complexity. In case there are only a few groups, the model tends to encourage larger group size to enhance local interactions for better performance; in case there are many groups, the number of interacting features inside each group will be reduced. In practice, the tuning grids for  $p$  can be set based on the number of input features and desired interpretability level.



(a) Classification performance and average feature-group size w.r.t. number of feature-groups ( $p$ ) in 4 datasets.

(b) Classification performance and average feature-group size w.r.t. varying thresholds  $\alpha_T$  (Eq. 6) in 3 datasets.

**Fig. 7.** Ablation results of model width and feature-group size.

**Effect of Feature-Group Size.** The threshold  $\alpha_T$  in Eq. 6 controls the density of feature interactions. Larger  $\alpha_T$  reduces features that fall into each group and

interaction density, boosting interpretability but potentially lowering accuracy. To explore the trade-off between interpretability and accuracy, we examine how  $\alpha_T$  influences FOFI’s performance. Figure 7b shows that an  $\alpha_T$  around 0.8 or 1.0 achieves an optimal balance between accuracy and interpretability by maintaining 3–7 features per group, ensuring clear interactions and notable classification accuracy. In comparison, too low  $\alpha_T$  increases accuracy but complicates interactions, while too high  $\alpha_T$  reduces accuracy.

**Effect of Adaptive Feature Quantization/Embedding.** The adaptive binning of continuous features in FOFI significantly enhances model performance. As in Table 4, replacing this step with an offline quantization (quantile normalization from Scikit-learn [17]) which is independent of the learning process, reduces the classification F1-score by 0.64%–1.30% across six datasets with mixed variables. This shows that the proposed feature quantizer is useful in generating discriminative feature representations and coordinating the interaction between continuous and categorical variables.

## 5 Conclusion and Future Work

We proposed FOFI, a novel neural network to identify flexible-order feature interactions for mixed types of variables for building accurate yet interpretable models at the group level. In the future, we will apply the model to higher-dimensional data; we are also interested in combining our approach with Bayesian learning to build interpretable models that are causally more meaningful.

**Acknowledgement.** This work was supported in part by the National Key Research and Development Program of China (2022YFC3400501), and the National Natural Science Foundation of China (62276099).

## References

1. Amekoe, K.M., Dilmi, M.D., Azzag, H., Dagdia, Z.C., Lebbah, M., Jaffre, G.: Tabsra: an attention based self-explainable model for tabular learning. In: ESANN (2023)
2. Asuncion, A., Newman, D.: Uci machine learning repository (2007)
3. Breiman, L.: Random forests. *Mach. Learn.* **45**, 5–32 (2001)
4. Chen, T., Guestrin, C.: Xgboost: a scalable tree boosting system. In: SIGKDD (2016)
5. Deng, H., Ren, Q., Zhang, H., Zhang, Q.: Discovering and explaining the representation bottleneck of dnns. In: ICLR (2022)
6. Enouen, J., Liu, Y.: Sparse interaction additive networks via feature interaction detection and sparse selection. In: NeurIPS (2022)
7. Gorishniy, Y., Rubachev, I., Babenko, A.: On embeddings for numerical features in tabular deep learning. In: NeurIPS (2022)
8. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. In: NeurIPS (2021)

9. Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: NeurIPS (2022)
10. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for ctr prediction. In: IJCAI (2017)
11. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: NeurIPS (2016)
12. Kleinbaum, D.G., Klein, M., Pryor, E.R.: Logistic regression: a self-learning text, vol. 94. Springer (2002)
13. Lerman, S., Venuto, C., Kautz, H., Xu, C.: Explaining local, global, and higher-order interactions in deep learning. In: ICCV (2021)
14. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2018)
15. Moro, S., Cortez, P., Rita, P.: A data-driven approach to predict the success of bank telemarketing. *Decis. Support Syst.* **62**, 22–31 (2014)
16. Olave, M., Rajkovic, V., Bohanec, M.: An application for admission in public school systems. *Expert Syst. Public Admin.* **1**, 145–160 (1989)
17. Pedregosa, F., et al.: Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
18. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. In: NeurIPS (2018)
19. Qiao, L., Wang, W., Lin, B.: Learning accurate and interpretable decision rule sets from neural networks. In: AAAI (2021)
20. Ren, J., Li, M., Liu, Z., Zhang, Q.: Interpreting and disentangling feature components of various complexity from dnns. In: ICML (2021)
21. Rendle, S.: Factorization machines. In: ICDM (2010)
22. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you? explaining the predictions of any classifier. In: SIGKDD (2016)
23. Schölkopf, B., Smola, A.J.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press (2002)
24. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: visual explanations from deep networks via gradient-based localization. In: ICCV (2017)
25. Song, W., et al.: Autoint: automatic feature interaction learning via self-attentive neural networks. In: CIKM (2019)
26. Tsang, M., Cheng, D., Liu, Y.: Detecting statistical interactions from neural network weights. In: ICLR (2018)
27. Tsang, M., Rambhatla, S., Liu, Y.: How does this interaction affect me? interpretable attribution for feature interactions. In: NeurIPS (2020)
28. Vaswani, A., et al.: Attention is all you need. In: NeurIPS (2017)
29. Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., Chi, E.: Dcn v2: improved deep & cross network and practical lessons for web-scale learning to rank systems. In: WWW (2021)
30. Wang, Z., Zhang, W., Liu, N., Wang, J.: Transparent classification with multilayer logical perceptrons and random binarization. In: AAAI (2020)
31. Wang, Z., Zhang, W., Liu, N., Wang, J.: Learning interpretable rules for scalable data representation and classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**(2), 1121–1133 (2024)
32. Yan, J., Chen, J., Wu, Y., Chen, D.Z., Wu, J.: T2g-former: organizing tabular features into relation graphs promotes heterogeneous feature interaction. In: AAAI (2023)



# Critical Feature Sifting and Dynamic Aggregation for Anomalous Audio Sequence Detection

Erteng Liu<sup>1</sup>, Kewei Gao<sup>1</sup>, Xing Zhou<sup>1</sup>, Sen Lin<sup>2</sup>, Jianhai Chen<sup>1</sup>, Yijun Bei<sup>1(✉)</sup>, and Zunlei Feng<sup>1</sup>

<sup>1</sup> School of Software Technology, Zhejiang University, Hangzhou, China  
beiyj@zju.edu.cn

<sup>2</sup> Ningbo Donghai Group Co., Ltd., Ningbo, China

**Abstract.** Anomalous Audio Sequence Detection (AASD) is a technique employed to identify atypical sound sequences, which are especially valuable in industrial monitoring applications, such as monitoring water pipes and various types of running machinery. The current deep model utilizes a combination of deep features to detect anomalies. However, the scarcity of anomalous samples and the abundance of redundant features contribute to unsatisfactory identification outcomes. In industrial monitoring scenarios, anomalies in audio sequences are typically identified by human experts who take various factors into consideration. Factors often play distinct roles in various scenarios. Therefore, we focus on tackling two key challenges: *1) What are the essential features for anomalous audio sequence detection? 2) How to adaptively fuse those critical features to detect the anomalous audio sequence for different tasks?* In response to the above two questions, we design an end-to-end adaptive feature selection mechanism to identify important and non-redundant components for detecting anomalous audio sequences. Furthermore, based on the sifted critical features, we devise a dynamic aggregation model to adaptively extract distinguishable features for detecting anomalous audio sequences in multiple scenarios. The proposed dynamic aggregation model employs a simple network architecture with fewer parameters to extract features from pre-processed critical features with label supervision. This can effectively account for the limitations of traditional feature-based methods and deep learning-based methods. The experimental evaluations on three classic industry monitoring datasets demonstrate that the proposed method achieves SOTA performance and exhibits superior recall performance when compared with existing methods.

**Keywords:** Anomalous Detection · Audio Sequence · Industrial Monitoring · Feature Sifting · Dynamic Aggregation

## 1 Introduction

Anomalous Audio Sequence Detection (AASD) is a widely used technique that aims to detect unusual or abnormal sound patterns from massive audio record-

ings. AASD is particularly useful for industrial monitoring applications, such as water pipe and machinery monitoring [3]. For example, it can be used to monitor potential leaks, malfunctions, or other issues that may affect equipment or system performance. AASD provides a highly accurate and reliable way to detect anomalies, enabling organizations to identify and address potential problems before they become severe or expensive to correct.

AASD is a challenging task due to several factors. Firstly, anomalous audio sequences come in various types, making it difficult to find specific patterns that distinguish them from normal audio sequences. Additionally, some anomalous audio sequences have minimal differences from normal audio sequences, which further increases the difficulty of AASD. Secondly, an audio sequence is typically composed of different types of noise. For instance, underground water pipes often contain the sounds of moving vehicles, while industrial settings are full of various machine noises. These noises also make it challenging to achieve accurate anomalous audio sequence detection.

To effectively address the above challenges in AASD, researchers have proposed two broad categories of techniques: traditional feature-based methods and deep learning-based methods. Traditional feature-based methods rely on signal processing and pattern recognition techniques to preprocess and extract features from audio signals. These features are then used with traditional classification or regression algorithms, such as SVM and Random Forest, for anomaly detection [8, 11, 16, 17, 21]. Deep learning-based methods, on the other hand, leverage deep neural networks for feature extraction and model training without requiring manual feature engineering [3, 20, 25]. Traditional feature-based methods offer the advantages of computational efficiency, interpretability, applicability to small audio datasets, and the ability to provide insights into the causes of anomalies. However, they require specialized domain knowledge and expertise [9], which can be difficult for non-experts. Furthermore, selecting appropriate features for different AASD tasks can be time-consuming, and pre-selected features often include redundant information that interferes with AASD performance. Additionally, traditional machine learning classifiers are unable to extract and combine new features that could benefit AASD.

In recent years, deep learning-based methods have also demonstrated good performance for AASD. Existing works can be broadly divided into unsupervised and supervised methods. The former model the distribution of a massive number of normal samples and identify the anomalous samples by detecting distributional differences between the anomalous and normal samples [3]. Supervised deep learning-based methods utilize annotation information to extract distinguishable features from anomalous and normal samples [4, 20, 25], thereby solving the problem of unsupervised methods. Nonetheless, the categories of anomalous and normal audio sequences are usually highly unbalanced, making it difficult to collect enough anomalous samples. The imbalanced distribution and insufficient anomalous samples leave deep networks biased towards normal audio samples, resulting in poor performance for anomalous samples.

In industrial monitoring scenarios, human experts identify anomalous audio sequences by taking into account various factors, such as audio frequency, intensity, rhythm, pitch, and other indicators. The relative importance of each factor varies across different scenarios. This paper aims to address two doubts based on previous works: 1) *What are the essential features for anomalous audio sequence detection?* 2) *How to adaptively fuse those critical features to detect the anomalous audio sequence for different tasks?*

To address the two aforementioned doubts, we propose a framework called CRITER, which stands for critical feature sifting and dynamic aggregation, applicable to AASD. To find the critical features from massive classic features, we introduce an end-to-end adaptive feature selection mechanism designed for audio anomaly detection. Our approach automates the selection of optimal features from a feature pool using a series of selectors, each comprising feature transformation and selection components. This mechanism reduces computational overhead and human intervention, ensuring efficient feature selection. Then, to effectively combine the critical features, we propose a dynamic aggregation model comprising a simple feature extractor, an attention module, and a linear classifier. Our dynamic aggregation model can aggregate feature information dynamically to capture anomalous patterns in various audio types. This streamlined network architecture effectively extracts relevant features when trained with supervised annotations. Furthermore, we collected a dataset named pipenoise of 10,000 audio sequences using the water pipe monitor deployed in the city. Extensive experiments on three datasets demonstrate that the proposed CRITER achieves SOTA performance with superior recall performance compared with existing methods.

Our contribution is to design an end-to-end adaptive feature selection mechanism to find out the critical and non-redundant features for different AASD tasks, which can serve as a reference criterion for future studies on anomalous audio detection. Then, a dynamic aggregation model with a streamlined network architecture is designed to extract beneficial information for AASD from the sifted critical features using annotations, which can effectively alleviate imbalanced anomalous and normal samples. What's more, we collect the first water pipe audio sequence dataset pipenoise containing 10,000 samples, which can be used as a benchmark for future studies on AASD. Extensive comparison experiments demonstrate that the proposed CRITER achieves SOTA performance on three AASD tasks and has superior performance on the recall metric.

## 2 Related Work

**Traditional Feature Based Methods.** The traditional feature-based method for Anomalous Audio Sequence Detection (AASD) relies on signal processing to preprocess audio signals and extract features, followed by the implementation of traditional classification or regression algorithms. These methods employ machine learning classifiers or statistical modeling techniques to identify anomalous audio sequences by selecting a set of specific audio features [11, 21]. Several

studies use support vector machines to detect anomalies in audio [9], while others use one-class support vector machines to model the normal distribution of input audio and unsupervised decision-making methods for anomaly detection [16, 17]. Hidden Markov Models are widely used for sequential data in audio anomaly detection [19], where Wavelet Mel-Frequency Cepstral Coefficients (WMFCC) and a sliding window-based HMM are utilized to detect anomaly sound signals and produce excellent results [6]. In [23], a pulse sound detection module is employed to separate foreground events and background audio, with a KNN classifier using Mel Frequency Cepstral Coefficient (MFCC) features utilized in the classification stage. The MFCC audio event fuzzy detection system [8] divides audio samples into normal and abnormal samples based on the value range and histogram of MFCC, with fuzzy logic system membership functions and rules established to achieve anomaly audio detection.

**Deep Learning Based Methods.** Supervised methods utilize Long Short-Term Memory (LSTM) neural networks to train MEL frequency cepstral coefficients and MEL filter bank energies extracted from printer audio for anomaly detection tasks [4]. Some tasks utilize CNNs to detect anomalies in gamma-map images generated from audio streams [20]. In [25], single-channel short-time Fourier Transform (STFT), Mehr scale and Mehr Frequency Cepstrum Coefficient (MFCC) spectrum are used along with DenseNet-121, MobileNetV2, and ResNet-50 to evaluate the impact of superimposed feature aggregation and access selection on anomaly detection. Unsupervised anomaly detection algorithms primarily use autoencoders for the reconstruction error detection of audio features. Bayram et al. [3] adopt auto-encoders to reconstruct the error of Mayer spectrum audio data for anomaly detection in the industrial field.

**Features for Anomalous Audio Sequence Detection.** Several studies use signal processing techniques such as direct filtering, short-time Fourier Transform, or wavelet Transform to obtain fundamental features for subsequent time-series audio anomaly detection [1, 2, 28, 31]. A commonly used approach for anomalous audio sequence detection is deep feature mining through deep neural networks [12, 24]. This technique involves the use of neural networks that draw inspiration from other fields, such as EcapaTDNN and speechVGG [5, 10, 15]. Additionally, anomaly detection modeling using evolution graphs is another feature design that can be applied to sequence audio anomaly detection [7]. Some studies propose new features by fusing existing ones to improve anomaly detection performance. For example, the MGCC feature [32] is created directly from the original features, and other studies fuse features in the model [18, 28].

### 3 Method

Direct processing of raw audio features requires high computational costs and offers poor interpretability. Traditional methods utilize hand-crafted features, which are computationally inexpensive and highly interpretable, but they require expert involvement to select or design appropriate features. Deep learning

methods possess strong performance, but directly modeling raw audio features demands significant computational resources and offers poor interpretability. Using low-level features such as mel-spectrograms for modeling can lead to information loss.

We combine the advantages of traditional methods and deep learning methods by proposing an end-to-end approach that adaptively selects hand-crafted features and performs dynamic aggregation, referred to as CRITER. First, we designed a feature adaptive selection mechanism that automatically selects several features from a candidate feature pool that significantly contribute to the objective function. Subsequently, we build a deep model based on these key features. During this process, we designed a feature aggregation model that ensures the model’s high interpretability while fully leveraging the advantages of deep networks.

In the following sections, we will introduce the critical feature sifting mechanism and the feature dynamic aggregation model.

### 3.1 Critical Feature Sifting

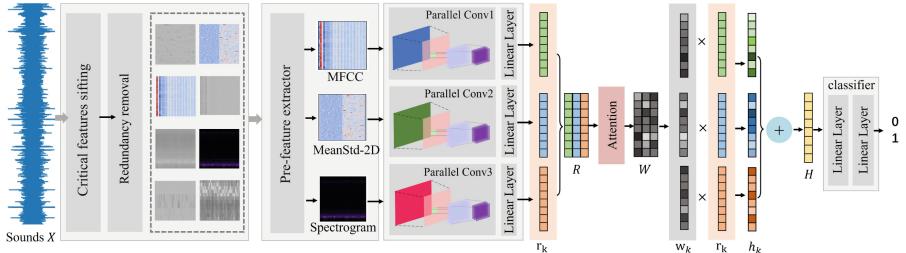
Traditional audio anomaly detection methods have extensively studied how to handle raw audio data. Researchers have summarized specialized features based on expertise and practical experience to address various application scenarios. However, selecting the appropriate features to build a model without expert involvement remains a challenge. A common approach is to experimentally verify the performance gain of candidate features to filter suitable ones. This method not only incurs significant computational costs but also requires human intervention. Consequently, the entire screening process must be repeated for each new application scenario.

We propose an end-to-end adaptive feature selection mechanism that can automatically select suitable features from a feature pool.

To ensure the model selects  $M$  suitable features from  $N$  features, we designed an adaptive feature selection mechanism composed of  $M$  selectors, each selecting one key feature. Specifically, each selector comprises a feature transformation component and a Gumbel-Softmax component. After processing by the feature transformation module of the  $m$ -th selector, the features  $F$  are converted into a vector  $f^m = \{f_1^m, f_2^m, \dots, f_N^m\}$ . We use the elements in  $f^m$  as the unnormalized probabilities of the corresponding features being selected as key features. The Gumbel-Softmax component is then used to obtain the normalized probability distribution as follows:

$$O^m = \frac{\exp((\log f^m + \epsilon)/\tau)}{\sum \exp((\log f_k^m + \epsilon_k)/\tau)},$$

where  $O^m$  denotes the output of the  $m$ -th selector, and  $\tau$  is a vector in which the elements are i.i.d samples drawn from the Gumbel distribution  $(0,1)$  to add a small amount of noise to avoid the argmax operation always selecting the element with the highest probability value.



**Fig. 1.** The framework of the proposed anomalous audio sequence detection method, which has two components: the critical feature selection module and the dynamic aggregation module. The former filters out essential and non-redundant features, whereas the latter extracts deep features from the selected features and aggregates them dynamically using an attention mechanism to obtain the final identification of anomalous audio sequences.

To ensure the selectors perform the selection, we convert  $O$  into the corresponding one-hot vectors during the forward pass, as shown below:

$$O^m = \text{one-hot}\{\arg \max_k (\log f_k^m + \epsilon_k)\}.$$

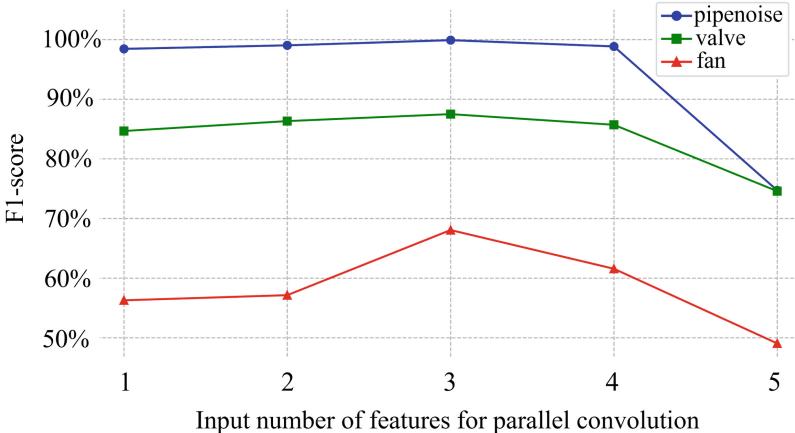
During backpropagation, the Gumbel-Softmax outputs are used to ensure the differentiability of the discrete sampling process. Finally,  $O \cdot F$  is used as the selected features. Ideally, the  $M$  selectors can select  $M$  features. However, we cannot guarantee that the features selected by each selector are unique. Therefore, we impose constraints on the selectors:  $\min \sum_m O^m$ , which is used to make the distribution of  $O$  more dispersed.

We present an ablation study examining the impact of the number of input feature types on the model's performance across three datasets: pipenoise, valve, and fan. The results, illustrated in Fig. 2, indicate that for all three datasets, the model achieves optimal performance when three types of input features are used. This trend suggests that incorporating up to three relevant features significantly enhances the model's effectiveness. However, adding more than three features does not provide further benefits and may even degrade performance due to the introduction of noise or overfitting. This underscores the importance of careful feature selection to maximize the model's accuracy and reliability.

### 3.2 Feature Dynamic Aggregation

The optimal feature combination is experimentally obtained in the previous section. Our attention now turns to the design of a simple model, addressing the second doubt: *how to fuse the critical features adaptively in order to identify anomalous audio sequences across different tasks?* The new model must have a streamlined network architecture in order to meet the design requirements. Combining the pre-processed critical features with the streamlined net-

work architecture effectively addresses the challenge of imbalanced distribution among anomalous and normal samples.



**Fig. 2.** Ablation study on the number of parallel convolution input features. The F1 score performance was evaluated on 3 datasets (pipenoise, valve, and fan). The model showed a slight advantage with 3 features on the pipenoise and valve datasets. On the fan dataset, the model exhibited a significant advantage when using 3 features.

Figure 1 depicts the use of sifted critical features in the design of a pre-feature extractor that computes the initial features of input audio data along three dimensions for subsequent parallel convolutional networks. The pre-feature extractor can either be a pure phonological or statistical feature extractor. Adaptive aggregation of deep features extracted from convolutional networks is achieved through the use of an attention mechanism, and is tailored to specific AASD tasks.

The convolution stage comprises three sets of parallel cascaded convolutions generating three sets of parallel convolution channels. The feature maps then enter the convolution channels, and the extraction of deeper features takes place. Given that input audio feature maps are of varying sizes, the next step involves conducting scale alignment operations on the deep features obtained from the convolution output. To achieve this, fully connected layers are adopted.

Although critical features may generate specific impacts in audio anomaly detection through convolution tasks, different algorithms prioritize different objectives based on specific scenarios. We introduce an attention mechanism to promote network adaptivity, dynamically aggregating critical features during the training process. This enables adjustment of the original feature weights according to varying application scenarios. The attention mechanism uses weighted aggregation to fuse the three groups of features.

Let  $R = \{r_1, r_2, r_3\}$  be deep features extracted from three parallel convolutional subnetworks. To obtain important weights for each  $r_k$ , we define an attention mapping function that determines the weight vector  $W = \{w_1, w_2, w_3\}$  as follows:

$$W = \mathcal{F}_{attention}(R),$$

where  $W$  is a concatenation of three weight vectors,  $\{w_1, w_2, w_3\}$ , and is obtained through the fully connected mapping function  $\mathcal{F}_{attention}(\cdot)$ . With the weight  $W$ , the representation  $H$  for the final classification is calculated as follows:

$$H = \sum_{k=1}^3 h_k, h_k = r_k \otimes w_k,$$

where  $\otimes$  denotes the point-wise multiplication. Finally, the final representation  $H$  is fed into the classifier, which is composed of two linear layers. During the learning and training phases, the model is trained with the Cross-Entropy loss function with fully annotated anomalous and normal samples.

**Table 1.** Comparison results of different methods on three datasets. **Bold** indicates the best performance.

Model	Pipenoise			Valve			Fan		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
AE [22]	99.71%	70.84%	82.83%	78.67%	85.51%	81.94%	62.16%	57.50%	59.74%
VAE [14]	100.00%	76.49%	86.68%	78.21%	88.41%	82.99%	61.39%	62.00%	61.69%
MobileNetV2 [27]	<b>100.00%</b>	54.72%	70.74%	67.95%	76.81%	72.11%	68.39%	59.50%	63.64%
RNN [30]	94.83%	99.79%	97.25%	69.12%	68.12%	68.61%	64.52%	30.00%	40.96%
LSTM [13]	99.79%	99.69%	99.74%	82.67%	89.86%	86.11%	61.74%	46.00%	52.72%
Time2graph [7]	99.69%	100.00%	99.85%	81.33%	88.41%	84.72%	64.52%	60.00%	62.18%
Ecapa-tdnn [10]	99.89%	91.79%	95.67%	83.56%	88.41%	85.92%	70.62%	62.50%	66.31%
PANNS [15]	99.81%	55.03%	70.95%	83.33%	86.96%	85.11%	62.69%	60.87%	61.76%
Res2Net [12]	99.21%	51.54%	67.84%	56.92%	53.62%	55.22%	55.67%	54.00%	54.82%
ResnetSe [24]	99.88%	85.73%	92.27%	61.64%	65.22%	63.38%	60.33%	55.50%	57.81%
TDNN [29]	99.85%	68.07%	80.95%	57.89%	63.77%	60.69%	71.76%	61.00%	65.95%
CRITER (ours)	99.80%	<b>100.00%</b>	<b>99.90%</b>	<b>84.00%</b>	<b>91.30%</b>	<b>87.50%</b>	<b>71.82%</b>	<b>64.68%</b>	<b>68.06%</b>

## 4 Experiment

This section details the experimental setup and datasets, including the pipenoise dataset from urban water networks and the MIMII-derived valve and fan datasets. Comparative analyses and ablation studies validate the effectiveness of our approach in detecting anomalous audio sequences.

## 4.1 Implementation Details

This section presents the datasets and the experimental setting in detail.

**Dataset.** In order to validate the effectiveness of our proposed method, we have assembled a vast pipeline noise dataset that currently holds the largest compilation of anomalous audio sequences. The pipenoise dataset is based on genuine business scenarios and is collected from urban water networks where audio data is recorded through acoustic sensors placed at different nodes to capture the sound of the underground pipelines. The dataset for pipeline noise comprises 10,000 audio sequence samples, of which 745 are anomalous. The audio samples of the pipenoise dataset have a sampling rate of 8000hz and are 5 s long. Additionally, two commonly used datasets, valve and fan, are also adopted to verify the generalization of the proposed method. The valve and fan datasets are derived from the MIMII dataset [26], a credible dataset that facilitates the study and examination of machinery faults in the industrial sector. The valve and fan datasets have an audio sampling rate of 16,000hz, and each audio sample has a length of 10 s. The MIMII dataset simulates numerous abnormal sounds made by industrial machinery. We adopt the 3 : 1 : 1 split setting for training, testing, and validating all datasets.

**Experimental Setting.** The proposed deep learning model comprises three parallel convolution blocks, each with the same structure connected through cascaded convolutions with convolution kernels of 5\*5 and 3\*3. Firstly, MeanStd-2D, Spectrogram, and MFCC audio features underwent convolution and dynamic aggregation. Subsequently, the features are transformed into embedded feature vectors through a linear layer, with a default embedding scale of 1024, which facilitates further dynamic feature aggregation operations. We employ an attention mechanism that results in two fully connected layers with equal parameters. Unaggregated vectors are concatenated to this mechanism for weight calculation, and the computed weights are used for weighted aggregation. The whole network is trained with the Cross-Entropy loss function, using the Adam optimizer. The learning rate for each dataset is set to 0.0001, with 1,000 epochs for the pipenoise dataset and 200 epochs for the other datasets. In the experiments, we adopt the commonly used classification evaluation metrics, namely Precision, Recall, and F1-score (Table 2).

**Table 2.** Information about the dataset used in the experiment.

Dataset type	Amount	Sampling rate	Anomaly rate	Duration
Pipenoise	10000	8000 hz	7.45%	5 s
Valve	1200	16000 hz	33.33%	10 s
Fan	1200	16000 hz	33.33%	10 s

## 4.2 Compare with Existing Methods

We compare the proposed CRITER against widely used supervised and unsupervised methods for detecting anomalous audio sequences. Unsuitable for this experiment is the Time2graph algorithm [7] that uses audio amplitude as network inputs. The other algorithms relied on MFCC input. The results in Table 1 reveal that CRITER exceeded the unsupervised method in the pipenoise dataset with F1-score at 99.90%, attaining a 15% to 20% rise. Furthermore, compared with algorithms such as EcapaTdn [10] and Time2Graph [7] known for their exceptional performance on the same dataset, CRITER has further boosted its original outcome. CRITER has the highest value of Recall, which is vital in anomalous audio detection tasks, for the pipenoise dataset. While some methods outperform the proposed method's Precision score on the pipenosie dataset, our technique shows a higher recall value. In detecting anomalous audio sequences, a higher recall value would make more sense since it can recognize as much anomalous data as possible. Similarly, on the valve and fan datasets, CRITER outranks both supervised and unsupervised methods in all three assigned indicators. For the valve dataset, the F1-score exceeds the LSTM [13] method by 1.6% compared to other methods, while on the fan data, CRITER's F1-score has improved by 2.9% compared to the TDNN [29]. Therefore, CRITER has an advantage on these datasets as the Recall and F1-score of the outcome reached by CRITER are better than other metrics. Ultimately, it is the dynamic aggregation of essential features that outperforms other methods.

## 4.3 Ablation Study

In this section, we first conduct a series of ablation studies on various components, which can effectively verify each component's necessity and effect. What's more, we also provide the ablation study on hyperparameter settings.

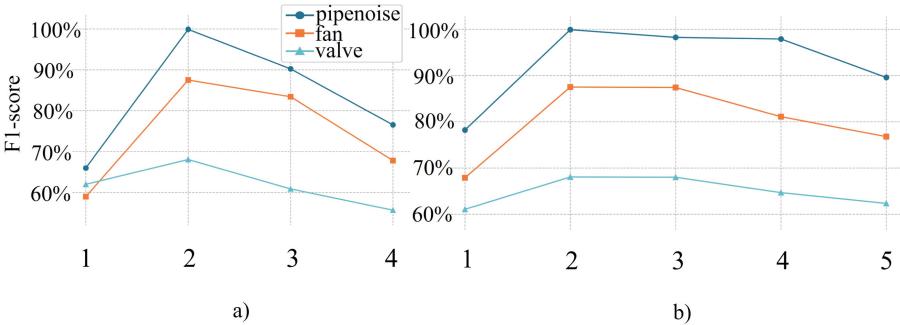
**Table 3.** Results of the ablation study conducted on the attention mechanism.

Dataset type	Without attention			With attention		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Pipenoise	98.88%	100.00%	99.44%	99.80%	100.00%	99.90%
Valve	85.29%	84.06%	84.67%	84.00%	91.30%	87.50%
Fan	56.57%	56.00%	56.28%	71.82%	64.68%	68.06%

**Attention Mechanism.** In this section, We conduct an ablation experiment to verify whether the added attention mechanism module for feature aggregation boosts performance. From Table 3, we can observe that the aggregated network with an attention mechanism results in better performance compared to the network without the module that relied solely on direct series convolutional

features. The results of this experiment concluded that the attention mechanism is more effective in feature fusion than directly aggregating the essential features that are screened by the leave-one-sift stage. The possible reason is that the attention mechanism can find out different ratios of critical features for different AASD tasks.

**Attention Feature Aggregation Mode.** In the proposed method, we directly aggregate the three feature components by adding them after multiplying them by the attention weights. Nevertheless, another way to aggregate is to first concatenate the feature components, before multiplying them by the attention weights and aggregating. We conduct an ablation experiment to compare the two methods. The results show that the F1-score of the concatenated aggregation method on the pipenoise dataset is only 99.60%, which is 0.3% lower than the F1-score of the direct addition aggregation method. Similarly, on the valve and fan datasets, the F1-score of the concatenated aggregation method is 2.64% and 3.41% lower than the F1-score of the direct addition aggregation method, respectively. These experimental results demonstrate the superiority of the direct addition aggregation method.



**Fig. 3.** Result of ablation study on the layer number of the network. The horizontal axis in both images represents the number of layers. a) Results with different layer numbers in the convolutional module; b) Results with different layer numbers in the fully connected classifier.

**The Layer Number of Network.** We extract deep features by inputting selected critical features into parallel convolutional layers. Ablation experiments verify that the number of layers in each convolutional module is optimal. After parallel convolution processes the critical features and aggregates them dynamically with the attention mechanism, the resulting features are input to fully connected layers for classification. We conduct ablation experiments on the number of fully connected layers in the classifier, comparing different architectures to classify the features and analyzing the results.

The experimental results are presented in Fig. 3, where we can observe that the optimal layer number for the convolutional module is 2. A significant decrease in model performance occurs when the layer number exceeds 2. Additionally, the optimal number of fully connected layers in the classifier should not exceed a certain depth. While the impact of layer number changes in the fully connected layers may not be as significant as that in the convolutional module, our experiments have shown that the optimal number of fully connected layers is still 2. Networks with more or fewer layer numbers display poor performance. The experiment results also verify that streamlined network architecture with suitable parameters can effectively combine the advantages of traditional feature-based methods and deep learning-based methods.

**Deep Learning Model vs SVM.** In this section, we investigate whether it is necessary to use deep learning to mine and dynamically aggregate critical feature combinations. In particular, we design ablation experiments to determine the best combination of critical features generated by direct concatenation without using deep convolution and dynamic aggregation.

Table 4 demonstrates that the experimental results on the datasets are worse when selecting the three most critical features and concatenating them as input for SVM training compared to the results obtained by using parallel convolutional dynamic aggregation with the attention mechanism. Therefore, deep networks need to be used for dynamic feature aggregation. The reason for this phenomenon can be attributed to two factors: dynamic aggregation with the attention mechanism can identify re-composed features and determine the optimal ratio of critical features for different AASD tasks.

**Table 4.** Comparison results between direct classification using SVM and deep network classification using feature aggregation.

Dataset type	Concatenated feature with SVM			CRITER		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Pipenoise	88.05%	73.75%	80.90%	99.80%	100.00%	99.90%
Valve	71.43%	66.67%	68.97%	84.00%	91.30%	87.50%
Fan	57.14%	53.33%	55.17%	71.82%	64.68%	68.06%

## 5 Conclusion and Future Work

This paper proposes a novel dynamic feature aggregation network based on critical feature selection, termed CRITER, for anomalous audio sequence detection. Inspired by the human experts' identification mechanism, we focus on handling two doubts: 1) *What are the essential features for anomalous audio sequence detection?* 2) *How to adaptively fuse those critical features to detect the anomalous audio sequence for different tasks?* To address the above two doubts, we

first devise two feature-sifting strategies to find the critical and non-redundant features from massive classic features. Then, a dynamic aggregation model combined with a linear classifier is devised to capture anomalous patterns in various audio types. The design model with streamlined network architecture can effectively extracts relevant features for anomalous audio sequence detection, which can effectively alleviate the drawbacks of imbalanced class distribution. To verify the effectiveness of the proposed CRITER, we collect the currently the largest anomalous audio sequence dataset. Extensive experiments demonstrate that the proposed CRITER can achieve the SOTA performance and effectively account for the limitations of traditional feature-based methods and deep learning-based methods. In the future, we will explore the proposed dynamic feature aggregation network trained with self-supervised learning, which can fundamentally solve the problem caused by the imbalance class distribution. What's more, we will also conduct research to improve models' adaptability for different scenarios with limited abnormal data and a large amount of normal data.

**Acknowledgement.** This work is supported by the Ningbo Key Research and Development Program (Grant No. 2023Z057) and Ningbo Natural Science Foundation (2022J182).

## References

1. Dufaux, A., Besacier, L., Ansorge, M., Pellandini, F.: Automatic sound detection and recognition for noisy environment. In: 2000 10th European Signal Processing Conference, pp. 1–4. IEEE (2000)
2. Barkana, B.D., Uzkent, B., Saricicek, I.: Normal and abnormal non-speech audio event detection using MFCC and PR-based feature sets. *Adv. Mater. Res.* **601**, 200–208 (2012)
3. Bayram, B., Duman, T.B., Ince, G.: Real time detection of acoustic anomalies in industrial processes using sequential autoencoders. *Expert. Syst.* **38**(1), e12564 (2021)
4. Becker, P., Roth, C., Roennau, A., Dillmann, R.: Acoustic anomaly detection in additive manufacturing with long short-term memory neural networks. In: 2020 IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA), pp. 921–926. IEEE (2020)
5. Beckmann, P., Kegler, M., Saltini, H., Cernak, M.: Speech-VGG: a deep feature extractor for speech processing. arXiv preprint [arXiv:1910.09909](https://arxiv.org/abs/1910.09909) (2019)
6. Chan, C.F., Yu, E.W.M.: An abnormal sound detection and classification system for surveillance applications. In: 2010 18th European Signal Processing Conference, pp. 1851–1855 (2010)
7. Cheng, Z., Yang, Y., Wang, W., Hu, W., Zhuang, Y., Song, G.: Time2graph: revisiting time series modeling with dynamic shapelets. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 3617–3624 (2020)
8. Dadula, C.P., Dadios, E.P.: Fuzzy logic system for abnormal audio event detection using MEL frequency cepstral coefficients. *J. Adv. Comput. Intell. Intell. Inform.* **21**(2), 205–210 (2017)

9. Davy, M., Desobry, F., Gretton, A., Doncarli, C.: An online support vector machine for abnormal events detection. *Signal Process.* **86**(8), 2009–2025 (2006)
10. Desplanques, B., Thienpondt, J., Demuynck, K.: ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification. arXiv preprint [arXiv:2005.07143](https://arxiv.org/abs/2005.07143) (2020)
11. Foggia, P., Petkov, N., Saggese, A., Strisciuglio, N., Vento, M.: Audio surveillance of roads: a system for detecting anomalous sounds. *IEEE Trans. Intell. Transp. Syst.* **17**(1), 279–288 (2016). <https://doi.org/10.1109/TITS.2015.2470216>
12. Gao, S.H., Cheng, M.M., Zhao, K., Zhang, X.Y., Yang, M.H., Torr, P.: Res2net: a new multi-scale backbone architecture. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**(2), 652–662 (2019)
13. Graves, A., Graves, A.: Long short-term memory. In: *Supervised Sequence Labelling with Recurrent Neural Networks*, pp. 37–45 (2012)
14. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint [arXiv:1312.6114](https://arxiv.org/abs/1312.6114) (2013)
15. Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., Plumbley, M.D.: PANNs: large-scale pretrained audio neural networks for audio pattern recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **28**, 2880–2894 (2020)
16. Lang, R., Lu, R., Zhao, C., Qin, H., Liu, G.: Graph-based semi-supervised one class support vector machine for detecting abnormal lung sounds. *Appl. Math. Comput.* **364**, 124487 (2020)
17. Lecomte, S., Lengelle, R., Richard, C., Capman, F.: One class support vector machines for audio abnormal events detection. In: *Statistical Signal Processing Workshop* (2011)
18. Li, X.L., Du, Z.L., Wang, T., Yu, D.M.: Audio feature selection based on rough set. *Int. J. Inf. Technol.* **11**(6), 117–123 (2005)
19. Liu, L., Li, B., Zhao, R., Yao, W., Shen, M., Yang, J.: A novel method for broiler abnormal sound detection using WMFCC and HMM. *J. Sensors* **2020** (2020)
20. Mahmud, M., Kaiser, M.S., McGinnity, T.M., Hussain, A.: Deep learning in mining biological data. *Cogn. Comput.* **13**(1), 1–33 (2021)
21. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. *IEEE Trans. Neural Netw. Learn. Syst.* **29**, 2063–2079 (2018)
22. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) *ICANN 2011. LNCS*, vol. 6791, pp. 52–59. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-21735-7\\_7](https://doi.org/10.1007/978-3-642-21735-7_7)
23. Megha, C.K., Reddy, V.K.: Robust classification of abnormal audio using background-foreground separation. In: *IEEE India Council International Conference* (2017)
24. Mekruksavanich, S., Jitpattanakul, A., Sitthithakerngkiet, K., Youplao, P., Yuponpin, P.: Resnet-se: Channel attention-based deep residual network for complex activity recognition using wrist-worn wearable sensors. *IEEE Access* **10**, 51142–51154 (2022)
25. Papadimitriou, I., Vafeiadis, A., Lalas, A., Votis, K., Tzovaras, D.: Audio-based event detection at different snr settings using two-dimensional spectrogram magnitude representations. *Electronics* **9**(10), 1593 (2020)
26. Purohit, H., et al.: MIMII dataset: sound dataset for malfunctioning industrial machine investigation and inspection. arXiv preprint [arXiv:1909.09347](https://arxiv.org/abs/1909.09347) (2019)

27. 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)
28. Sufisher2017anomaly, Y., Zhang, K., Wang, J., Madani, K.: Environment sound classification using a two-stream CNN based on decision-level fusion. *Sensors* **19**(7), 1733 (2019)
29. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoust. Speech Signal Process.* **37**(3), 328–339 (1989)
30. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint [arXiv:1409.2329](https://arxiv.org/abs/1409.2329) (2014)
31. Zeng, W., Lin, Z., Yuan, C., Wang, Q., Wang, Y.: A new learning and classification framework for the detection of abnormal heart sound signals using hybrid signal processing and neural networks. In: 2020 39th Chinese Control Conference (CCC) (2020)
32. Zhang, K., Su, Y., Wang, J., Wang, S., Zhang, Y.: Environment sound classification system based on hybrid feature and convolutional neural network. *Xibei Gongye Daxue Xuebao/J. Northwest. Polytech. Univ.* **38**(1), 162–169 (2020)



# Parallel Interpretation Network via Semantic Visual Probe and Counterfactual Verification

Hao Wan, Keyang Cheng<sup>(✉)</sup>, and Hao Zhou

Jiangsu University, No. 301, Xuefu Road, Zhenjiang, China  
 [{haowan,zhouhao}@stmail.ujs.edu.cn](mailto:{haowan,zhouhao}@stmail.ujs.edu.cn), [kycheng@ujs.edu.cn](mailto:kycheng@ujs.edu.cn)

**Abstract.** Interpretable Deep Neural Networks (DNNs) are becoming increasingly crucial, particularly in high-risk domains such as medical diagnosis and autonomous driving. Understanding the reasoning behind AI decisions is paramount. To address this need, a novel parallel interpretation network based on Semantic Visual Probe (SVPProbe) and Counterfactual Verification is proposed, aiming to enhance the interpretability of deep neural networks and facilitate user comprehension of network decisions. The novelty of the network lies in that SVPProbe incorporates prior knowledge into gradient computation and explains the textual attributes and their corresponding visual mappings that guide the model decisions. Concurrently, Counterfactual Verification generates counterfactual samples by altering the color, texture and shape information in the images, feeding them into the original model to evaluate the consistency of different decision interpretations, thus providing reliability verification for SVPProbe's interpretations. This parallel interpretation network structure enables users to understand network decisions more intuitively and efficiently. Extensive experiments on pre-trained models and public datasets demonstrate that our method effectively reveals the evidence behind model decisions and outperforms existing methods in terms of interpretability and credibility. Specifically, on VGG16, ResNet50 and ResNet101, the Bounding box (Bbox) scores are 64.85%, 77.02% and 77.26%, respectively. Compared with other methods, the method improves the attribute accuracy by an average of 1.8% and the top 1 attribute accuracy by 1.9%.

**Keywords:** Interpretability · Probe · Counterfactual · Visualization

## 1 Introduction

Interpretable methods aim to explain or present deep learning in a way that is understandable to humans. There is increasing interest among researchers in creating techniques to visualize and interpret deep neural networks. However, most existing methods either fail to reflect the true internal logic of the DNN or sacrifice performance for interpretability, which is problematic in high-stakes

environments. This paper introduces a perspective that allows for the interpretation of information encoded in the convolutional layers, thereby elucidating the decision-making process of Convolutional Neural Networks (CNNs) [19, 20].

In this paper, the following research question is addressed: How can we interpret the information encoded in the convolutional layers of a CNN and thus understand its decision-making process? A novel method is proposed that combines semantic and visual interpretation with reliability verification. The method consists of two components: a parallel interpretation network based on Semantic Visual Probe (SVProbe) and a Counterfactual Verification mechanism. The key contributions of this paper are as follows:

**Parallel Interpretable Network.** Unlike traditional interpretable networks, our parallel approach interprets at the middle layer of the network rather than at the output layer. This reduces the complexity and computation required for interpretation while increasing the transparency and trustworthiness of the network.

**SVProbe Method.** The SVProbe method innovatively incorporates prior knowledge into gradient computations. It elucidates the top attributes in the input by attaching the probability of each attribute to the filter via gradient attribution. By weighting feature maps with these attribute probabilities, it provides the visual mapping corresponding to the top attributes.

**Counterfactual Verification.** Counterfactual Verification is proposed to assess the reliability of interpretable models. It involves modifying the original image's color, texture, and partial shape before inputting it into the model to compare the outputs under different conditions, thereby contrasting the reliability of the interpretation. Based on this, an innovative reliability assessment metric is also introduced.

## 2 Related Work

Based on their purposes, interpretability methods can be classified into two main types: intrinsic interpretability [10, 16, 24, 36] and post-hoc interpretability [3, 21, 37–39]. Intrinsic interpretability refers to models that are inherently interpretable, with interpretability built into the model itself. This can be accomplished by applying constraints like sparsity, monotonicity, causality, or physical restrictions informed by domain knowledge. Common approaches include prototype-based methods [22] and concept-based methods [33]. Post-hoc interpretability methods seek to understand and clarify how trained deep learning models reach their decisions, offering detailed insights into the model's reasoning process after it has made predictions. Prominent techniques include attribution methods (such as GradCAM, gradients, or layer activations) [2, 23], perturbation methods (Counterfactual) [5], among others. Our approach combines the transparency of intrinsic interpretability with the flexibility of post-hoc interpretability, thereby achieving a more comprehensive interpretability framework.

**Visual Interpretation.** Numerous efforts [6, 28, 31] have been made to investigate saliency map methods that highlight the pixels with the greatest impact on model predictions. The saliency map approach in CNNs is the most straightforward method for uncovering patterns hidden within neurons. The up-convolutional network [9] transforms the feature map of the convolutional layer into an image. Gradient-based visualisation [6, 28] evaluates input images that maximizes the activation score of neural units by gradients in back propagation. All of the above methods are only global displays of network attention regions. The visualization does not offer a clear semantic interpretation, so users are expected to infer or speculate about its meaning. Our approach, however, is fine-grained, able to identify specific parts of an object and associate them with textual attributes.

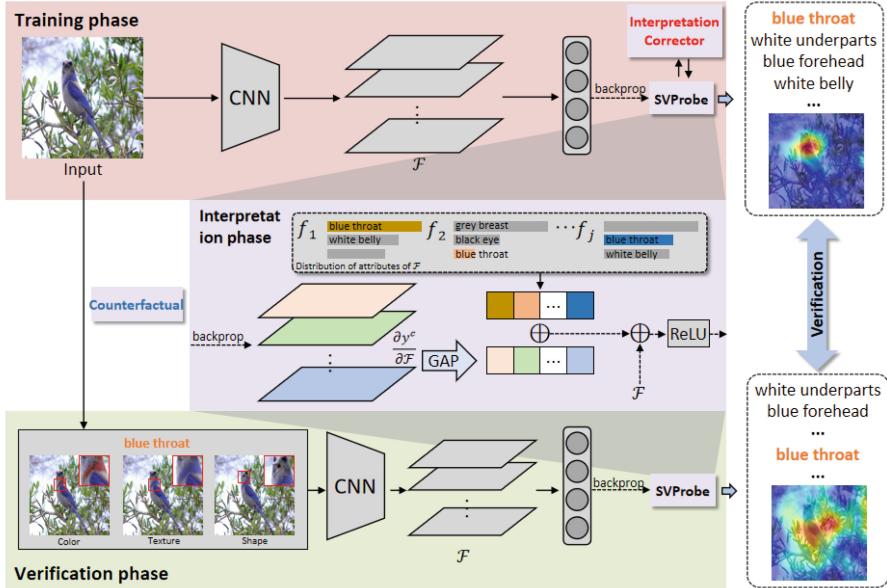
**Textual Interpretation.** Visual interpretation emphasizes key areas of an image that impact the decision, but it doesn't clarify how these regions relate to the model's reasoning process. Consequently, some research is dedicated to generating textual explanations for the decisions made by visual classifiers [14]. Additionally, other studies focus on creating multimodal explanations that connect textual and visual explanations [26]. The latest extension of this work [15] first produces multiple textual interpretations and then filters out those that cannot find a basis in the image. These methods are essentially image captioning tasks that generate captions with more category-distinctive information. Semantic Network Interpretation [11] is only capable of generating textual attributes for interpretation, but it is limited to networks that have a globally averaged pooling layer, and it does not give visual interpretation.

**Concept Discovery.** Interpretability methods based on concept discovery focus on explaining a model's decision-making process using human-understandable concepts, identifying the key concepts that the model relies on when making decisions. A number of existing methods are based on this, such as, CRAFT [32] improves model interpretability by generating concept-based explanations, CLIP-Dissect [34] automatically describes the function of hidden layer neurons. Our approach, while inspired by these methods, incorporates a priori knowledge into gradient computation and is able to explain image attributes and their visual mappings in relation to model decisions, and use probability distributions to represent the relationship between image attributes and network decisions. In this way, the researcher is able to determine which types of visual features the network is sensitive to.

### 3 Method

In this section, the parallel interpretation network based on SVProbe and Counterfactual Verification is designed to interpret the model and verify the reliability of its interpretations. An overview is provided in Fig. 1. In the training phase, probes are implanted into the trained model to obtain the text attributes associated with each filter. In the interpretation phase, the input samples are

backpropagated to obtain the gradient features of the corresponding layers. By performing global average pooling (GAP) on these gradients and weighting the attribute probabilities of each filter along with the feature maps, we generate visual mappings of the corresponding attributes. Counterfactual verification uses image modification to modify the original image's color, texture, and shape before inputting it into the model to be interpreted with a probe, and the reliability of the interpretation is judged based on the differences in interpretation before and after interpretations.



**Fig. 1.** Overview of interpretation network. During the training phase, each image is combined with textual attributes to train the SVProbe. SVProbe can obtain the probability of attributes contained in each filter. In the interpretation phase, SVProbe acquires the gradient representation of the filters through class-level backpropagation, which is then weighted by the attribute probabilities after global average pooling (GAP) to interpret the class-level attributes of the input image. Additionally, visual mappings are obtained based on the class-level attributes and feature maps. In the validation phase, counterfactual modifications are applied to the original image and input into the model to verify the consistency (reliability) of the interpretation.

### 3.1 Semantic Visual Probe

To understand the mechanism by which a black-box model operates and the evidence it uses to predict outcomes, we embed a probe in one layer of the model

to detect the semantic knowledge (attributes) carried by the filters. Attributes are attached to the input image and the probe is trained alongside the trained model. We denote the convolutional layer of the model as  $\mathcal{F} = \{f_j | j = 1, \dots, m\}$ , and the input sample as  $\mathcal{X} = \{x_i | i = 1, \dots, n\}$ . Define  $t$  as a text attribute and  $x^t$  represents the input sample  $x$  that contains attribute  $t$ . The output of a filter  $f$  with input  $x$  is denoted as  $f(x)$ , referred to as the feature map, and its width and height dimensions are indexed by  $u$  and  $v$ , respectively. We compute the gradient  $\frac{\partial y^c}{\partial f(x)}$  of the score  $y^c$  of class  $c$  with respect to the filter feature map  $f(x)$ . The probability for the attribute  $t$  contained in the filter  $f$  can be represented as:

$$p(t|f) = \frac{1}{n_t} \sum_i \phi\left(\frac{\partial y^c}{\partial f(x_i^t)}\right) = \frac{1}{n_t} \sum_i \frac{1}{Z} \sum_u \sum_v \frac{\partial y^c}{\partial (f(x_i^t))_{uv}} \quad (1)$$

where  $n_t$  represents the number of samples in the set that contain attribute  $t$ ,  $Z = (\sum_u \sum_v 1)$  is the number of pixels in the feature map and  $\phi(\cdot)$  denotes the global average pooling. Here,  $1/n_t$  is defined as the prior probability of attribute  $t$ . It indicates that the more occurrences in a set, the smaller the amount of information (prior probability) it carries. The trained probe is similar to the following formula.

$$\mathcal{T} = \begin{matrix} & attr_1 attr_2 \cdots attr \\ filter_1 & \begin{pmatrix} 0.04 & 0.05 & \cdots & 0.12 \\ 0.11 & 0.04 & \cdots & 0.06 \\ \vdots & \vdots & \ddots & \vdots \\ 0.09 & 0.15 & \cdots & 0.08 \end{pmatrix} \\ filter_2 \\ \vdots \\ filter_m \end{matrix} \quad (2)$$

It means the probability of each attribute encoded in the filter  $f_j$ . The training process for an SVProbe is outlined in Algorithm 1.

---

**Algorithm 1.** Training Semantic Visual Probe

---

**Input:** Image  $\mathcal{X}$ , Conv-layer  $\mathcal{F}$ , Class  $c$ .

**Output:** Attribute set  $\mathcal{T}$  of filters.

- 1: **Initialization:** Let one convolutional layer of the model be  $\mathcal{F}$  and its output is  $\mathcal{F}(x)$ . Let the set of attributes corresponding to the filter  $\mathcal{T} \in \mathbb{R}^{(\text{number of } f \in \mathcal{F}) \times (\text{species of } t)}$  and  $x$  contains the attribute  $t$  denoted as  $x^t$ .
  - 2: Get the gradient  $\frac{\partial y^c}{\partial \mathcal{F}(x)}$  of the score  $y^c$  relative to the Conv-layer output  $\mathcal{F}(x)$ .
  - 3: **for** every  $t$  **do**
  - 4:   **for**  $x^t$  in  $\mathcal{X}$  **do**
  - 5:      $\mathcal{T}[:, t] \leftarrow \mathcal{T}[:, t] + \phi\left(\frac{\partial y^c}{\partial \mathcal{F}(x^t)}\right)$
  - 6:   **end for**
  - 7:    $n_t = n_t + 1$
  - 8:    $\mathcal{T}[:, t] \leftarrow \mathcal{T}[:, t] / n_t$
  - 9: **end for**
  - 10: **return**  $\mathcal{T}$
- 

According to the attribute distribution of the filter derived from Eq. 1, we can interpret the prediction results for any network input  $x$ . For a model input  $x$ ,

**Algorithm 2.** Semantic Visual Probe Interpret**Input:** Image  $x$ , Attribute set  $\mathcal{T}$  of filters.**Output:** Attribute set  $\mathcal{T}^x$  of  $x$ , Visual mapping set of attribute  $\mathcal{L}^x$ .1: **Initialization:**2: Get the gradient  $\frac{\partial y^c}{\partial \mathcal{F}(x)}$  of the score  $y^c$  relative to the Conv-layer output  $\mathcal{F}(x)$ .3:  $\mathcal{T}^{x'} \leftarrow \mathcal{T} \cdot \phi(\frac{\partial y^c}{\partial \mathcal{F}(x)})$ 4: **for** every  $t$  **do**5:    $\mathcal{T}^x[t] \leftarrow \text{sum}(\mathcal{T}^{x'}[:, t])$ 6:    $\mathcal{L}^x[t] \leftarrow \text{sum}(\mathcal{T}^{x'}[:, t] \cdot \mathcal{F}(x))$ 7: **end for**8: **return**  $\mathcal{T}^x, \mathcal{L}^x$ 

we back-propagate the gradient based on the predicted score and perform global average pooling, i.e.,  $\phi(\frac{\partial y^c}{\partial \mathcal{F}(x)})$ . This represents the importance of each filter in the convolutional layer. Thus, the probability that the network input  $x$  contains the attribute  $t$  is shown as follows.

$$p(t|x) = p(t|\mathcal{F})\phi\left(\frac{\partial y^c}{\partial \mathcal{F}(x)}\right) = \sum_j p(t|f_j) \frac{1}{Z} \sum_u \sum_v \frac{\partial y^c}{\partial (f_j(x))_{uv}} \quad (3)$$

where  $p(t|f_j)$  is the attribute distribution of the  $j$ -th filter,  $\phi(\frac{\partial y^c}{\partial \mathcal{F}(x)})$  represents the importance of sample  $x$  in the network decision-making process for filter  $f_i$ . The  $p(t|x)$  represents the probability that the input sample  $x$  possesses the attribute  $t$ .

Moreover, in addition to textual attributes, we aim to intuitively understand the visual information encoded in the model prediction process. Previous CAM-like methods only provide global interpretations of salient objects in an image. Our approach, however, offers fine-grained interpretations that identify specific parts of an object and correspond them with textual attributes. The visual information corresponding to input  $x$  is obtained from the product of the probability of attribute  $t$  in each filter and the feature map, which can be represented as:

$$L = p(t|x)\mathcal{F}(x) = p(t|\mathcal{F})\phi\left(\frac{\partial y^c}{\partial \mathcal{F}(x)}\right)\mathcal{F}(x) = \sum_j p(t|f_j) \frac{1}{Z} \sum_u \sum_v \frac{\partial y^c}{\partial (f_j(x))_{uv}} f_j(x) \quad (4)$$

In Fig. 1, we show the pipeline for interpreting the attributes contained in an input image with its corresponding visual mapping, and the interpretation process is detailed in Algorithm 2. Note that our algorithm is not restricted to a specific convolutional layer and can be applied to any layer.

### 3.2 Counterfactual Verification

In this section, we utilize counterfactual methods to verify the accuracy of the interpretation. Using the key features like texture and color acquired during the training and interpretation phases, counterfactual verification is conducted.

Initially, we apply counterfactual modifications to the key texture or color features of the input image, re-input the modified image into the model, and then evaluate the results to verify the model’s interpretations.

Each raw image is composed of three color channels: Red (R), Green (G) and Blue (B). We modify the color, texture and shape information of different parts of the extracted bird regions using the following method.

**Color Modification:** For each pixel  $(p_x, p_y)$  and color channel  $\epsilon$ , we adjust the color value using the following formula.

$$x'_{p_x, p_y, \epsilon} = x_{p_x, p_y, \epsilon} + \Delta_\epsilon \quad (5)$$

where  $\Delta_\epsilon$  denotes the value of the color channel modification, which can be either positive or negative. With this approach, we can systematically change the value of the color channel on a per-pixel basis and combine it with visual mapping to produce local color transformations in the image, such as beaks, wings, etc. This targeted approach ensures that counterfactual modifications are meaningful.

**Texture Modification:** We employ a Gaussian filter for texture modification. For each pixel  $(p_x, p_y)$  and color channel  $\epsilon$ , we blend the original pixel value with the average value of neighboring pixels using a Gaussian kernel.

$$x'_{p_x, p_y, \epsilon} = \frac{1}{N} \sum_{p_i, p_j} x_{p_x + p_i, p_y + p_j, \epsilon} \times GK_{p_i, p_j} \quad (6)$$

where  $N$  is the size of the Gaussian filter, and  $GK_{p_i, p_j}$  represents the weight of the Gaussian filter at position  $(p_i, p_j)$ .

**Shape Modification:** We employ the LFR-GAN [8] to shape-modify the regional features of the initial image so that they closely resemble the features of the region of attention.

$$x' = LFR\text{-}GAN(x) \quad (7)$$

LFR-GAN modifies features by optimizing the latent space in StyleGAN [18].

To ensure that the generated counterfactual data stays within the original data distribution and to address the problem of out-of-distribution (OOD) counterfactual data, we use a discriminator network in the counterfactual verification framework. Discriminator networks can be used as filters to verify the authenticity of the generated counterfactuals. The discriminator network is trained to differentiate between real samples from the original data distribution and generated counterfactual samples. It outputs a probability score indicating the likelihood that a given sample is a true sample. By integrating the discriminator network into our counterfactual verification framework, we enhance the truthfulness and validity of the generated counterfactuals, ensure that they stay within the original data distribution, and effectively solve the OOD counterfactual problem. This approach improves the robustness and reliability of our interpretability methods.

Finally, we input the modified image into the base model and interpret it with SVProbe. The reliability of the interpretation is judged based on comparing the consistency between the pre-modified interpretation and the modified interpretation. For an attribute, a consistent interpretation before and after modifying the image indicates that the interpretation is unreliable and vice versa.

## 4 Experiments

### 4.1 Experimental Settings

The proposed method was assessed using the Birds-200-2011 (CUB) [35] and Stanford Cars [17]. The bird dataset includes 11,788 images across 200 bird subclasses, with 5994 images in the training dataset and 5794 images in the test set. Each image contains both the category information and the attribute details of the bird. The car dataset includes 8144 training samples and 8041 test samples for 196 car models. The category information is from GPT-4 [25], including headlights, doors, and car models.

Three pre-trained models were used for evaluation, VGG-16 [30], ResNet-50 [13], and ResNet-101 [13]. They all take images of size  $256 \times 256 \times 3$  as input, so all images were resized before being fed into the model. Model parameters were fine-tuned from the official ones provided. SVProbe training was done on two RTX 3090 GPUs, highlighting that it doesn't require an exceptional training environment.

### 4.2 Visualizations

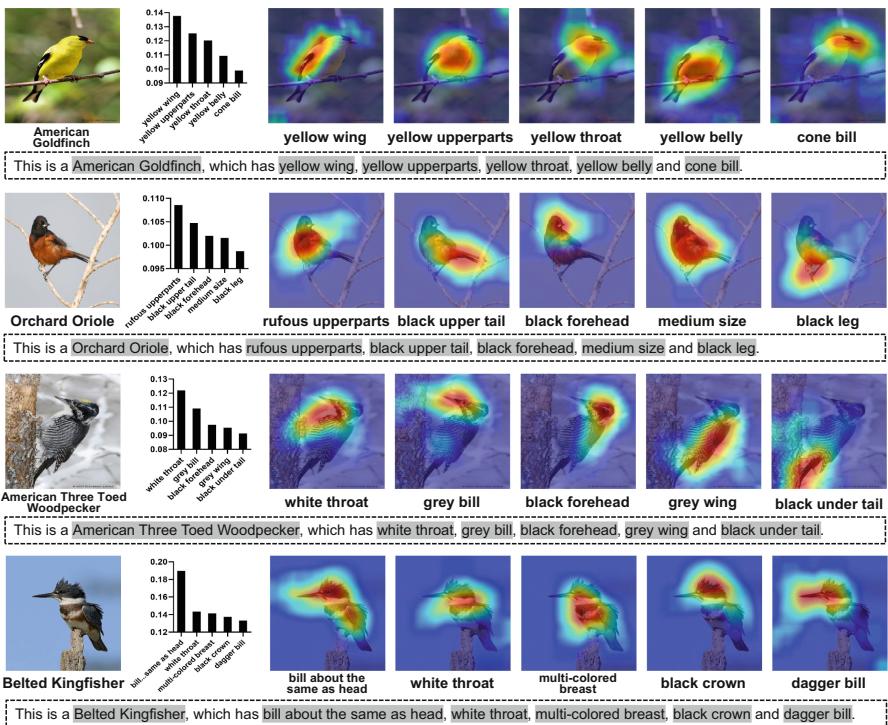
In this experiment, we present our method visually to show its effects intuitively. First, SVP interprets the semantic attributes contained in the input image and maps them to the corresponding visual features, which are then visualized. The text attributes of the filter are trained on ResNet-50, and then the class-level text attributes are derived from the gradient weights of the corresponding layers of the input image. Subsequently, Fig. 2 shows the text attributes generated for image classification with their corresponding visual mappings. The figure showcases four input samples, each illustrating the attributes for the top 5 probabilities. The histograms are plotted with the probabilities of the attributes from largest to smallest, and the visual mappings are arranged accordingly. For example, in the first row of Fig. 2, the first one is the input image, the second one shows the top 5 textual attributes and the corresponding probabilities derived by the interpretation network, and the next five images correspond sequentially to the visual mapping of the top 5 textual attributes.

The experimental results demonstrate that the SVProbe can explain the reason for decisions about the network, which is consistent with human knowledge. This is achieved by weighting the probability of a single attribute with the feature map in a set of filters, each with probability distributions for several attributes. This approach allows us to focus precisely on the target localization.

### 4.3 Quality of Interpretations

To evaluate the accuracy of our text attributes, we compared the top 5 textual attributes of each image with those of other models. As shown in Table 1, our gradient-based approach exhibits better performance than other approaches. It is worth mentioning that our proposed method is inspired by Semantic Network, which uses the feature map of the model with the weights in the global average pooling layer as the probabilities of the text attributes. However, Semantic Network requires that the model must have a global average pooling layer, which limits its applicability. Our method, SVPProbe, is based on gradient features after backpropagation and is not limited by the structure of the model. Therefore, SVPProbe can achieve better results.

As our method provides fine-grained interpretation of image parts, comparison with previous saliency maps like CAM [40], GradCAM [29], RISE [12] is not feasible. Therefore, the bounding box (Bbox) [27] metric is used to quantify



**Fig. 2.** Examples of top textual attributes and corresponding visual mappings in network decision making. Each image corresponds to having five attributes and visual mappings along with their probabilities. It's important to emphasize that SVP is designed not to precisely predict image attributes and mappings but to accurately reflect the rationale behind the neural network's decisions.

**Table 1.** Results of the accuracy comparison with the top 5 textual attributes of different models.

	mean	No.1	No.2	No.3	No.4	No.5
Semantic Network Interpretation [11]	75.8	89.2	88.1	83.3	66.4	51.9
Interpretable Basis Decomposition (IBD) [4]	71.4	85.3	82.4	79.4	60.1	49.9
Network Dissection [7]	74.4	88.4	<b>88.9</b>	81.5	63.3	50.1
Deep Generator Network [1]	75.4	89.1	86.3	80.2	65.7	<b>55.9</b>
<b>SVProbe (ours)</b>	<b>77.6</b>	<b>91.1</b>	88.5	<b>85.3</b>	<b>67.1</b>	55.8

**Table 2.** Evaluation results of Bbox metrics on VGG16, ResNet50 and ResNet101.

	mean	No.1	No.2	No.3	No.4	No.5
VGG-16	<b>64.85</b>	68.25	65.56	64.98	64.27	61.19
ResNet-50	<b>77.02</b>	80.11	78.53	77.20	75.18	74.08
ResNet-101	<b>77.26</b>	81.76	77.43	76.33	75.94	74.82

the effect of attribute visual mapping on the localization of objects of interest. Assume that the bounding box is  $B$  and the top  $k$  pixels of an attribute's visual mapping are  $P$ . We can obtain the Bbox score by  $Bbox = ((B \cap P)/P) \times 100$ . The bounding box scores for each attribute under the three different models were evaluated as shown in Table 2. The visual mapping of attributes significantly overlaps with the bounding box, indicating accurate interpretation. The more layers of the model, the better the attributes are learned and the more accurate the visual mapping, which corresponds to the accuracy of the model.

**Table 3.** Results of a user study on the top 5 attributes

	mean	No.1	No.2	No.3	No.4	No.5
Accurate	<b>77.5</b>	92.5	85.0	80.0	67.5	62.5
Reasonable	<b>74.0</b>	90.0	80.0	72.5	67.5	60.0
Accurate & Reasonable	<b>70.0</b>	85.0	75.0	67.5	65.0	57.5

#### 4.4 User Study to Assess Interpretations

The ultimate goal of the interpretation is for the user to understand the modeling decisions. Therefore, a user study is necessary. We aim to evaluate how useful these interpretations are for users in understanding the decision model. To this end, we design a process for user participation in the evaluation. Participants are shown a series of sextuplets containing the original image and five

visual mappings with corresponding textual attributes, similar to Fig. 2. In addition, we design several questions for each visual mapping and the corresponding attributes.

**Participants:** We recruit 40 participants for this study. Participants are recruited via online forums and offline requests. The demographic distribution includes 24 males and 16 females, with ages ranging from 20 to 35 years. The majority of the participants are graduate students (70%), with the remaining being undergraduate students (15%) and professionals (15%).

**Expertise:** Participants have varying levels of expertise in computer vision: 70% have prior experience or coursework in computer vision, 20% have some familiarity, and 10% are novices.

**Evaluation Process:** Participants are shown a series of sextuplets containing the original image and five visual mappings with corresponding textual attributes. For each visual mapping and its corresponding attributes, participants answer the following questions:

*Q1. Is the text attribute describing part of the target? (Accurate/Inaccurate)*

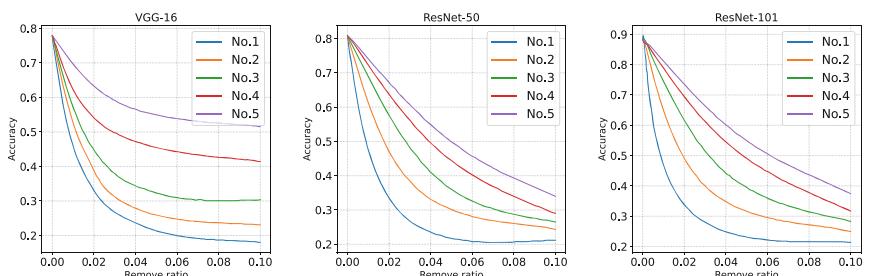
*Q2. Is the visual mapping part of the target? (Reasonable/Unreasonable)*

If the above two questions are affirmed, then the next one is triggered.

*Q3. Is the textual attribute and the visual mapping corresponding? (Reasonable & Accurate)*

If this question is also affirmed, this suggests that the image is interpreted with reasonable attention as well as accurate textual attributes.

The mean results for the participants are reported in Table 3, where participants rate the accuracy of text attributes at 77.5%, the reasonableness of visual mapping at 74.0%, and accurate and reasonable at 70.0%. It is shown that SVPProbe successfully explains the model's decisions semantically and visually. These interpretations are useful for users to understand the network.



**Fig. 3.** Perturbation results for the top 10% filters with top attributes on different models.

## 4.5 Faithfulness Evaluations

Validating the fidelity of interpretability methods ensures they accurately reflect the model’s internal workings and reveal the decision-making process. Solely relying on quantitative model evaluation is misleading. Therefore, filter perturbations are introduced to ensure the fidelity of the interpretation. We select the top 5 attributes and use counterfactual methods to perturb the filters contributing the top 10% of each attribute to assess the impact on accuracy. As shown in Fig. 3, we report the results of the perturbation on three models. It can be seen that filters with top attributes have a greater impact on the accuracy of the model. The main reason is that responding to filter importance through the gradient method allows the proposed method to more accurately reflect internal operations of the network. Interpretations are consistent with the model’s prediction results and decision-making process. The interpretation can accurately reveal the importance and impact of different features, as well as how the model makes predictions and decisions based on these features.

## 5 Conclusion

In this paper, we propose a parallel interpretation network based on SVProbe and Counterfactual Verification, aiming to improve the interpretability of deep neural networks and facilitate users’ understanding of network decisions. By incorporating a prior knowledge into gradient computation, SVProbe interprets textual attributes and their corresponding visual mappings to guide the model’s decision-making process. Meanwhile, Counterfactual Verification generates counterfactual samples by altering the image’s color, texture, and shape information. These samples are then input into the original model to assess the consistency of different decision interpretations and provide reliability verification for SVProbe’s interpretations. This parallel interpretation network structure enables users to understand network decisions more intuitively and efficiently. Experiments demonstrate that the method reveals the evidence behind model decisions and outperforms existing methods regarding interpretability and trustworthiness. Future research can further explore and optimize the method to further improve the interpretability and reliability of the model.

**Acknowledgments.** This work was supported by National Natural Science Foundation of China [62372215] and Special fund project of Jiangsu Science and Technology Plan [BE2022781].

## References

1. Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Conference and Workshop on Neural Information Processing Systems (NIPS) (2016)

2. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. In: Proceedings of the 34th International Conference on Machine Learning (2017)
3. Bau, D., Zhu, J.-Y., Strobelt, H., Lapedriza, A., Zhou, B., Torralba, A.: Understanding the role of individual units in a deep neural network. Proc. Natl. Acad. Sci. **117**(48), 30071–30078 (2020)
4. Zhou, B., Sun, Y., Bau, D., Torralba, A.: Interpretable basis decomposition for visual explanation. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)
5. Moreira, C., Chou, Y.L., Hsieh, C.: Benchmarking instance-centric counterfactual algorithms for XAI: from white box to black box. ACM Comput. Surv. **57**(6), 1–37 (2024)
6. Chattpadhyay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-cam++: generalized gradient-based visual explanations for deep convolutional networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 839–847. IEEE (2018)
7. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: quantifying interpretability of deep visual representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
8. Deng, Z., He, X., Peng, Y.: LFR-GAN: local feature refinement based generative adversarial network for text-to-image generation. ACM Trans. Multimedia Comput. Commun. Appl. **19**(6), 1–18 (2023)
9. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4829–4837 (2016)
10. Fan, F., Li, M., Teng, Y., Wang, G.: Soft autoencoder and its wavelet adaptation interpretation. IEEE Trans. Comput. Imaging **6**, 1245–1257 (2020)
11. Guo, P., Farrell, R.: Semantic network interpretation. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW), pp. 400–409 (2022)
12. Gupta, A., Vedaldi, A., Zisserman, A.: Inductive visual localisation: factorised training for superior generalisation. In: British Machine Vision Conference 2018, BMVC 2018, Newcastle, UK, 3–6 September 2018, p. 7. BMVA Press (2018)
13. 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 (CVPR) (2016)
14. Hendricks, L.A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., Darrell, T.: Generating visual explanations. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 3–19. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46493-0\\_1](https://doi.org/10.1007/978-3-319-46493-0_1)
15. Hendricks, L.A., Hu, R., Darrell, T., Akata, Z.: Grounding visual explanations. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 264–279 (2018)
16. Jiang, C., Zhao, Y., Chu, S., Shen, L., Tu, K.: Cold-start and interpretability: turning regular expressions into trainable recurrent neural networks. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 3193–3207 (2020)
17. Krause, J., Stark, M., Deng, J., Feifei, L.: 3D object representations for fine-grained categorization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops, pp. 554–561 (2013)

18. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4401–4410 (2019)
19. Mahmud, M., Kaiser, M.S., McGinnity, T.M., Hussain, A.: Deep learning in mining biological data. *Cogn. Comput.* **13**(1), 1–33 (2021)
20. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2063–2079 (2018)
21. Mei, S., Montanari, A.: The generalization error of random features regression: precise asymptotics and the double descent curve. *Commun. Pure Appl. Math.* **75**(4), 667–766 (2019)
22. Nauta, M., Schlötterer, J., Van Keulen, M.: Pip-net: patch-based intuitive prototypes for interpretable image classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
23. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning (2017)
24. Nauta, M., Van Bree, R., Seifert, C.: Neural prototype trees for interpretable fine-grained image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14933–14943 (2021)
25. OpenAI. GPT-4 technical report. [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) (2023)
26. Park, D.H., et al.: Multimodal explanations: justifying decisions and pointing to the evidence. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8779–8788 (2018)
27. Schulz, K., Sixt, L., Tombari, F., Landgraf, T.: Restricting the flow: information bottlenecks for attribution. In: International Conference on Learning Representations (2020)
28. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)
29. Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
31. Scholz, S., Weidmann, N.B., Steinert-Threlkeld, Z.C.: Improving computer vision interpretability: transparent two-level classification for complex scenes. arXiv preprint [arXiv:2407.03786](https://arxiv.org/abs/2407.03786) (2024)
32. Fel, T., Picard, A., Bethune, L., Boissin, T., Vigouroux, D., Colin, J.: Craft: concept recursive activation factorization for explainability. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2711–2721 (2023)
33. Oikarinen, T., Das, S., Nguyen, L.M., Weng, T.-W.: Label-free concept bottleneck models. In: International Conference on Learning Representations (2023)
34. Oikarinen, T., Weng, T.-W.: Clip-dissect: automatic description of neuron representations in deep vision networks. In: International Conference on Learning Representations (2023)
35. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD birds-200-2011 dataset (2011)

36. Wang, T.: Gaining free or low-cost interpretability with interpretable partial substitute. In: International Conference on Machine Learning, pp. 6505–6514. PMLR (2019)
37. You, J., Leskovec, J., He, K., Xie, S.: Graph structure of neural networks. In: International Conference on Machine Learning, pp. 10881–10891. PMLR (2020)
38. Yu, S., Principe, J.C.: Understanding autoencoders with information theoretic concepts. *Neural Netw.* **117**, 104–123 (2019)
39. Zhang, Q., Cao, R., Shi, F., Wu, Y.N., Zhu, S.-C.: Interpreting CNN knowledge via an explanatory graph. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
40. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2921–2929 (2016)



# Real-Time Decentralized M2M Decision-Making via Deep Learning and Incremental Learning

Mohamed Dwedar<sup>1</sup>(✉), Fatih Bayram<sup>2</sup>, and Alexander Jesser<sup>1,2</sup>

<sup>1</sup> University of Heilbronn, Heilbronn, Germany

[icps@hs-heilbronn.de](mailto:icps@hs-heilbronn.de)

<sup>2</sup> Institute for Intelligent Cyber Physical Systems (ICPS), Künzelsau, Germany

<https://www.hs-heilbronn.de/de/icps>

**Abstract.** This study introduces a methodological framework for facilitating Machine-to-Machine (M2M) communication among autonomous systems originating from diverse domains, such as Nao Robots, Darwin (humanoid robots), and drones. This is achieved through the establishment of machine-to-machine connectivity utilizing the MQTT Mosquitto broker. Acting as a mediator, this broker facilitates the seamless transmission and conversion of various data types between devices and the central host. Consequently, this eliminates the requirement for the participating machines to share a common domain for communication.

By centralizing the connection of all autonomous machines to the broker, they can efficiently transmit and relay essential data to the host for processing. Within the host environment, a sophisticated deep learning model is employed to analyze sensor data received from all autonomous machines. Subsequently, based on this analysis, decisions are autonomously made regarding task allocation to the most suitable machine, ensuring the fastest and most efficient execution without human intervention. This research thus serves as a fundamental cornerstone in the realm of M2M communication, facilitating essential decision-making processes through deep learning mechanisms, while minimizing human involvement. For real-time classification, incremental learning has been utilized to continually merge new data with historical data, enabling ongoing evaluation.

**Keywords:** Deep Learning · M2M · Robotics · Autonomous Systems · Decentralized Decision-Making · IoT · Embedded AI

## 1 Introduction

Machine-to-Machine (M2M) communication is vital in the Internet of Things (IoT) ecosystem, facilitating data exchange and task execution between interconnected devices without human intervention [1]. This method uses a network

of devices, from simple sensors to advanced machinery, that independently collect, transmit, and act on data, using protocols like MQTT, CoAP, and AMQP for their efficiency [2].

MQTT's lightweight messaging system excels in environments requiring minimal bandwidth, ideal for remote sensors and battery-powered devices. A broker often mediates these exchanges, crucially hosted on platforms like Raspberry Pi or Arduino, enriching data for AI model development [3].

M2M communication thrives over diverse networks such as cellular, wired, and emerging technologies like LPWAN, including LoRaWAN and NB-IoT, which support long-range, low-energy communication.

Security is critical, necessitating strong encryption and authentication to protect data integrity and privacy in M2M communications [4]. This technology is transformative for automated systems, enhancing intelligence and efficiency across sectors.

M2M configurations increasingly rely on brokers and specific device configurations (IPs or DNS) for connectivity. This setup begins direct interactions between machines and brokers, essential for subsequent data processing and Deep Learning model creation, dependent on the type of data extracted and task execution flexibility [5, 6].

Our study involves connecting two humanoid robots (Nao and Darwin-OP) and a drone using M2M MQTT Mosquitto to perform a specified task, extracting and preprocessing data to develop a Deep Learning model. This model will identify the most suitable system for the task, leveraging neural networks for accurate decision-making given their precision and decisiveness in data analysis [7, 8].

We also utilize incremental learning to verify the model's real-time efficacy, incorporating a feedback loop to blend new with historical data, which significantly improves accuracy [9, 10].

## 2 State of Art

This research aims to enhance the security and operational efficiency of interconnected devices within IoT ecosystems through a comprehensive approach. It advocates for integrating multi-domain autonomous systems using advanced machine learning techniques, particularly emphasizing online-real time solutions to improve decision-making in networking and control systems.

A central focus of this study is exploring Machine-to-Machine (M2M) connectivity, enabling direct information exchange among autonomous systems like humanoid robots, smart vehicles, and robotic arms. Despite its benefits, this interconnectivity poses challenges in data extraction due to diverse operating systems and data types. Monitoring M2M signals is crucial to ensure efficient communication and intelligent control across systems.

Deep Learning emerges as a strategic solution to address challenges posed by diverse machine types. By developing a shared AI model through Neural Networks, the framework aims to create a universal operational environment

adaptable to each autonomous system's intricacies. This approach facilitates dynamic adaptation of deep learning models to various data types, optimizing task allocation efficiency.

### 3 Related Work

R. S. Batth and A. Nayyar in [11] Described The integration of IoRT into various domains has the potential to revolutionize how robots are utilized, offering enhanced intelligence and autonomy., our study offers a detailed, practical framework for enhancing robotic operations through centralized data processing and incremental learning besides, it focuses on a specific implementation of M2M communication and deep learning for autonomous systems, emphasizing real-time decision-making and task allocation.

Kabir and R.; Watanobe, [12] discussed the use of cloud computing to support robotic systems enhances their ability to detect and respond to unknown objects in real-time. This approach reduces the computational load on individual robots, making the system more scalable and responsive to environmental changes, whereas this study focuses on SVM for object detection and cloud-based processing wherese, our work utilizes deep learning models for task allocation and incorporates incremental learning for continuous improvement.

Andronie, M. and Lăzăroiu, [13] described The combination of big data management, deep learning, and geospatial simulation significantly improves the functionality and accuracy of robotic systems. But, our research specifically applies these technologies to M2M communication and real-time task allocation.

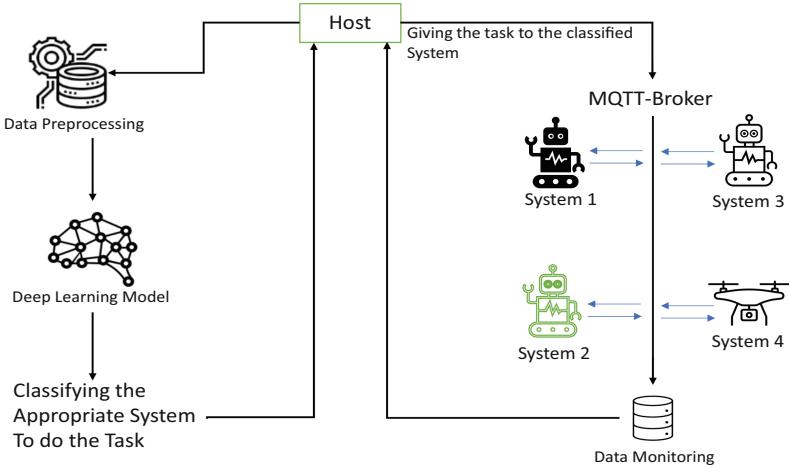
L. Chen and X. Huang, [14] described the design and implementation of AI and IoT based autonomous mobile robot system for cleaning garbage, This system showcases the potential for AIoT to automate routine tasks, improving efficiency and reducing human labor in maintenance and cleaning activities. In our study, M2M has been implemented more than one different system domains like humanoid robots and drones on the other hand DL is utilized on the host.

Toma, C. and Popa, [15] investigated the use of edge machine learning for automated decision-making and visual computing in robots, IoT devices, and drones. By processing data at the edge, the system reduces latency and improves real-time responsiveness, enhancing the overall performance of autonomous systems.

V. Richard and R. Mathias, [16] described a cooperative human-robot picking system that uses a multi-robot system with BDI (Belief-Desire-Intention) agents. The system facilitates human-robot collaboration for efficient task execution, leveraging learning algorithms to improve performance over time, While our work focuses on autonomous decision-making and task allocation.

M. Mukhandi and D. Portugal, [17] proposed a secure communication framework for ROS-enabled robotic systems using the MQTT protocol. The solution incorporates authentication and data encryption to protect against cyber-attacks, ensuring secure and reliable communication between robots and remote clients but out research focuses on M2M communication for task allocation and

decision-making, while this study addresses security concerns in robot communications.



**Fig. 1.** M2M Connection and Deep Learning Methodology

## 4 Methodology

In Fig. 1, The host is typically the central control unit or the main computational device that oversees the entire M2M communication process. Acting as the administrative center, the host manages and coordinates all interactions between the connected devices. It handles crucial tasks such as initiating communication, processing incoming data, executing control commands, and ensuring the overall health and security of the network. The host often has significant computing power and storage capacity to manage these responsibilities efficiently.

The MQTT-Broker manages the transmission of messages between different systems connected in the network. As the nerve center for MQTT-based communications, the broker facilitates message passing between clients [17]. It ensures that messages are correctly published and subscribed to relevant topics, maintaining session persistence and message queuing as needed. This is critical for systems where consistent and reliable message delivery is crucial for operational integrity.

System 1 to System 4 These represent individual machines or systems that are part of the network. Each system might have specific roles or functions within the overall process. In a typical setup, each system or device is configured with specific capabilities with a specific unique IP/TCP or DNS. For example, System 1 is a Nao Robot, System 2 is a DARwin-OP Robot, System 3 is another Nao Robot, and System 4 is DJI Mavic 2 Drone. Each system interacts with

the MQTT broker to send and receive relevant information, allowing them to function autonomously yet cohesively within the larger network.

Data Monitoring step involves monitoring the data that flows between the systems. It includes gathering, observing, and analyzing data to ensure everything operates correctly. Continuous data monitoring is essential for proactive maintenance and operational efficiency. It helps in detecting anomalies, tracking system performance, and ensuring compliance with operational standards. This data is typically visualized on dashboards and used in reports that inform decision-making processes.

Before the data can be used effectively, it often needs to be cleaned, formatted, and transformed. This step prepares the data for further analysis or processing. Effective data preprocessing may include normalization to scale data, handling missing values, and feature extraction that simplifies complex data into a form easier for machines to interpret. This is crucial for maximizing the accuracy of the models that will process this data [18].

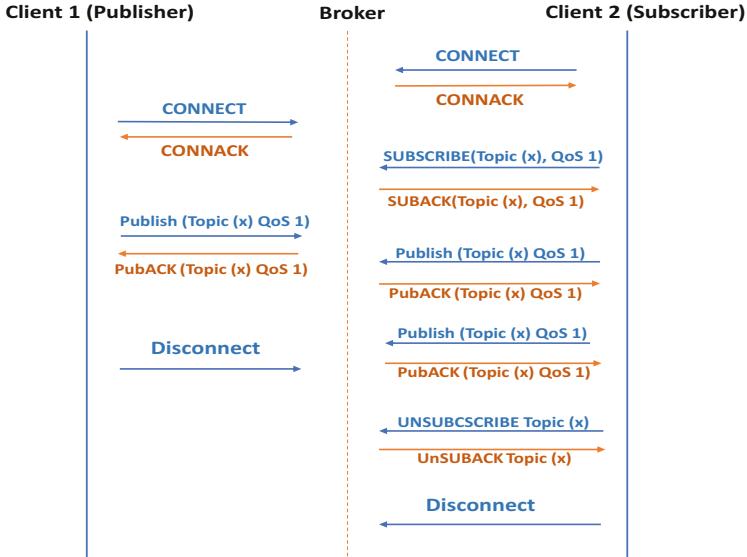
After processing the data, the deep learning model classifies and determines which system within the network is best suited to carry out a particular task based on the analysis. This classification is critical in optimizing resource allocation and operational efficiency. By assigning tasks to the most suitable system, the network can reduce bottlenecks, minimize response times, and enhance overall performance, ensuring that each part of the network is used to its full potential.

## 5 Designing M2M Communication Using MQTT and Mosquitto

MQTT (Message Queuing Telemetry Transport) is an efficient messaging protocol tailored for networks with limited bandwidth or reliability, ideal for M2M (Machine-to-Machine) communications. Mosquitto, an open-source MQTT broker, facilitates message exchanges among MQTT clients [18].

To connect machines within an M2M system to an MQTT broker, MQTT client libraries must be installed on each device. Proper network configuration, including IP addresses and port numbers, is essential. Each MQTT client is configured to handle connections, subscriptions, and messages securely using TLS/SSL. The steps to establish an M2M communication system using MQTT are outlined in Fig. 2 as described in [19]:

- **Broker Role:** Manages client connections, message delivery based on Quality of Service (QoS), and message flow between publishers and subscribers.
- **Connection:** Systems connect to the MQTT broker, which confirms connections with a CONNACK (connection acknowledgment).
- **Subscription:** Clients subscribe to topics with specific QoS levels, confirmed by the broker with SUBACK (subscription acknowledgment) messages.
- **Publishing:** Messages are published to topics at defined QoS levels, acknowledged by the broker according to the QoS specified.
- **Transmission:** The broker forwards messages to subscribers of the topics.



**Fig. 2.** MQTT Connectivity between Publisher and Subscriber

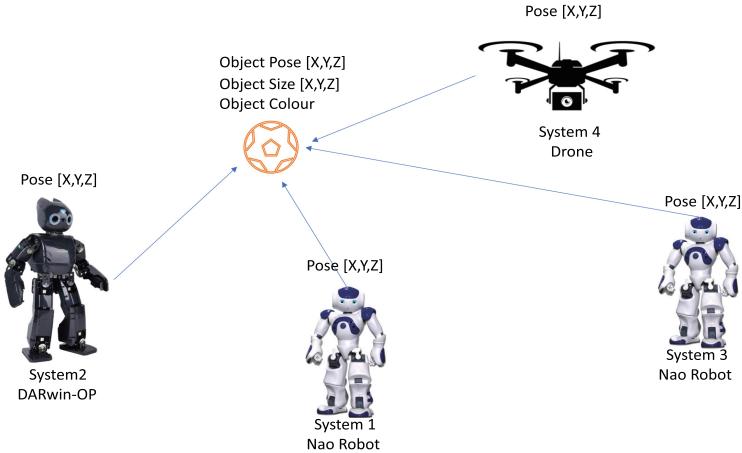
## 6 Task Identification and Data Structure

### 6.1 Task Identification

As shownen in Fig. 3, the aim of this project is to precisely identify the position, color, and size of a designated target specifically, a ball and to determine which of the four participating systems is best suited to track, pursue, and approach the target. Each system must transmit comprehensive data to a central broker, including their precise three-dimensional coordinates ( $X, Y, Z$ ), their current velocities  $V$ , and their most effective trajectories  $(X, Y, Z, \Theta)$  to reach the target [20].

Moreover, each system is required to provide updates on its operational status. This entails reporting the remaining battery life, any potential malfunctions, and the exact trajectory of the ball relative to its position, which is crucial for measuring the distance to the target. Throughout their mission, these systems may face obstacles in their intended paths. However, each is outfitted with a sophisticated obstacle avoidance algorithm that allows them to detect and circumvent any barriers they encounter effectively.

The primary objective is to assess and select the most appropriate system to accomplish the mission, based on an extensive real-time data collection from each system. This data includes spatial coordinates, system health, and obstacle navigation capabilities. In real time data monitoring, the data is sent to the broker and preprocessed at a central host where it is analyzed by an advanced deep learning model. This model evaluates the data to identify which system is best



**Fig. 3.** Task Design and Identification

equipped in terms of navigation efficiency, obstacle handling, energy efficiency, and operational dependability to successfully execute the task [21].

## 6.2 Data Structure

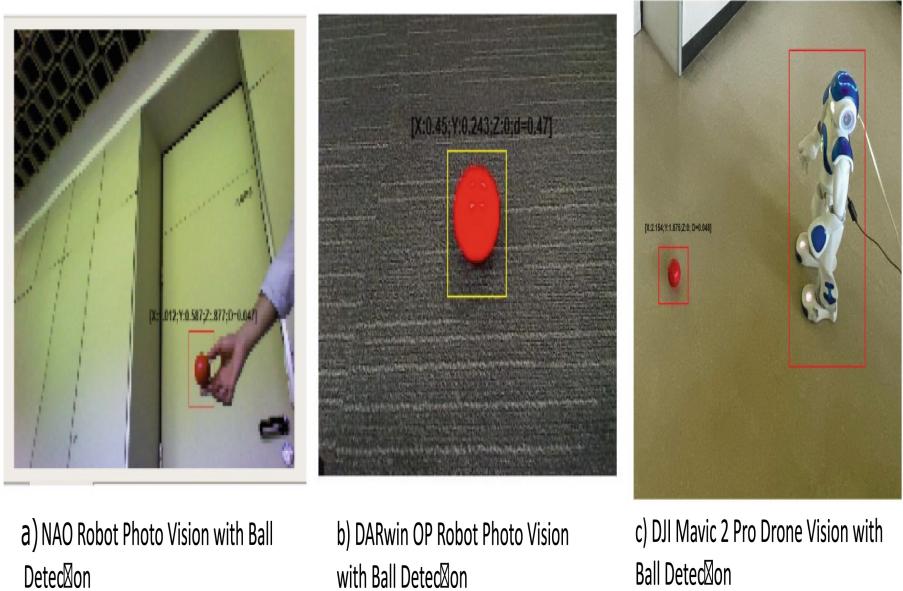
All data gathered from sensors and cameras within each system is transmitted to the main broker as real-time instant messages [22]. This data is then compiled into a specific time-based frame format as shown in Table 1.

**Table 1.** Collected Data Structure from the Broker

Features Type	Data Shape
Timestamps	0.01 s
Systems 1–4 Navigation Poses	Coordinates [X, Y, Z] (m)
Object Position	Coordinates [X, Y, Z] (m)
Systems 1–4 specific Velocities	$V \left( \frac{m}{s} \right)$
Systems 1–4 best trajectory	[X, Y, Z] (m)
Systems 1–4 failures	True – False
Detected object diameter	(m)
Detected object Color	Color
System 1–4 estimated distance	(m)
System 1–4 Battery Life	Percentage %

Additionally, OpenCV is utilized to gather and store the data in a CSV file, facilitating subsequent processing on the host. This setup also simplifies the

transfer of data to the deep learning model for the following decision-making task [19].



**Fig. 4.** Systems' Visions with Red Ball Detection (Color figure online)

Figure 4a) presents a visual capture from System 1, which consist of NAO robot equipped with cameras that supply visual data. The *ALTracker* module employs image processing techniques to identify the ball using its color, shape, and size.

Video from the camera is relayed to the processing unit, where image process is conduct. Additionally, a red boundary box had been integrated into the algorithm to enhance data visibility and to clearly indicate the ball's detected position within the boundary box [23].

Figure 4b) presents a visual capture from System 1 System 2 is a DARwin Op robot which has a built-in camera captures visual input. The *ColorFinder* class analyzes these inputs in the HSV color space to detect specific colors and compute the location of these colors in the frame, identifying the position of objects like colored balls. Web-based Interface: Real-time adjustments and viewing are facilitated through a web server implemented by *mjpgstreamer*, enabling changes to camera settings and the tracking of different colors via a user-friendly interface. Additionally, a yellow boundary box had been integrated into the algorithm to enhance data visibility and to clearly indicate the ball's detected position within the boundary box [24].

Figure 4c) shows System 4 which is DJI Mavic 2 Pro which equipped with a Hasselblad L1D-20c camera that features a 1-inch CMOS sensor. This setup

enables it to capture high-quality 20MP still images and 4K video, acclaimed for its exceptional capture of light and color. The drone employs its vision systems to identify obstacles by processing sensory data in real-time. This data helps to accurately ascertain the position and distance of nearby objects in relation to the drone. For subject tracking, the Mavic 2 Pro utilizes its ActiveTrack 2.0 system, which integrates deep learning algorithms with visual data captured by the camera. This sophisticated tracking mechanism allows the drone to keep a moving subject in focus during flight, even in environments with complex dynamics. Additionally, a red boundary box had been integrated into the algorithm to enhance data visibility and to clearly indicate the ball's detected position within the boundary box [25].

## 7 Data Preprocessing and Labeling

Data labeling, following the monitoring and preprocessing stages, is crucial in the data processing pipeline. Once data is refined and metrics are extracted from the broker, labeling commences, classifying each system's suitability for specific tasks.

System evaluation is based on well-defined, technical criteria:

- **Proximity to the Target:** Quantified using the Euclidean distance to evaluate closeness to the target, calculated by:

$$D = \sqrt{(X_t - X_s)^2 + (Y_t - Y_s)^2 + (Z_t - Z_s)^2}$$

where  $(X_s, Y_s, Z_s)$  and  $(X_t, Y_t, Z_t)$  are the coordinates of the system and target, respectively.

- **Remaining Battery Percentage:** Preference is given to systems with higher battery life to ensure task completion without interruptions.
- **Energy Efficiency and Operational Reliability:** Systems with lower energy consumption and reliable navigation are favored to enhance sustainability and ensure reliable operation under dynamic conditions.

Labels are assigned as follows:

*Label '1' (Most Appropriate):* For systems surpassing others in meeting the specified criteria, indicating optimal performance for task execution.

*Label '0' (Less Appropriate):* For systems that fall short of the necessary criteria, indicating unsuitability for the task.

## 8 Deep Learning Model Design and Evaluation

An Artificial Neural Network (ANN) consists of interconnected neurons linked by weighted connections that demonstrate the strength of influence between them [28]. Long Short-Term Memory (LSTM) networks are especially effective for handling sequential, real-time data due to their ability to maintain relevant

information over prolonged periods, crucial for predicting system appropriateness as new data flows continuously [29].

Neurons operate through aggregation, summing all weighted inputs, and activation, where the sum is transformed to produce the neuron's output. Popular activation functions include as mentioned in [30]:

- Sigmoid Function,

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

ideal for binary outputs, mapping inputs to a 0, 1 interval.

- Hyperbolic Tangent Function,

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

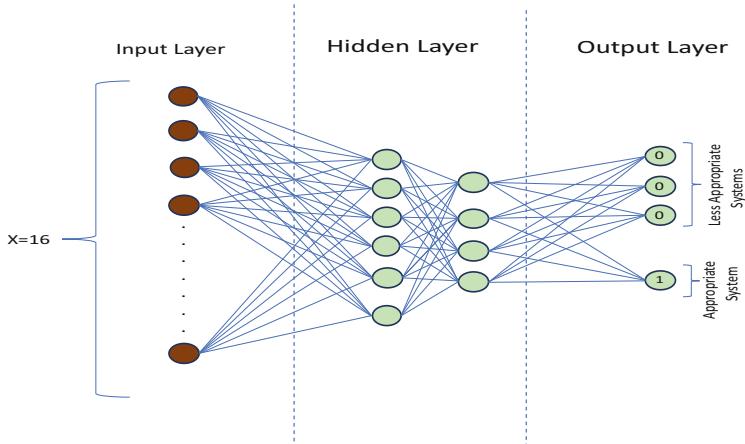
which normalizes outputs to -1 and 1.

The model employs the sigmoid function for a binary output—0 for “Less Appropriate” and 1 for “Most Appropriate.” The Rectified Linear Unit (ReLU),

$$R(x) = \max(0, x)$$

commonly used for its efficiency, is adapted to ensure outputs between 0 and 1.

Critical model components include the ADAM optimizer, which optimizes learning rates for each parameter by estimating gradients' first and second moments, enhancing learning efficiency. The Sparse Categorical Crossentropy loss function, aligned with the model's binary classification approach, is selected for its suitability in scenarios with class index targets [28].

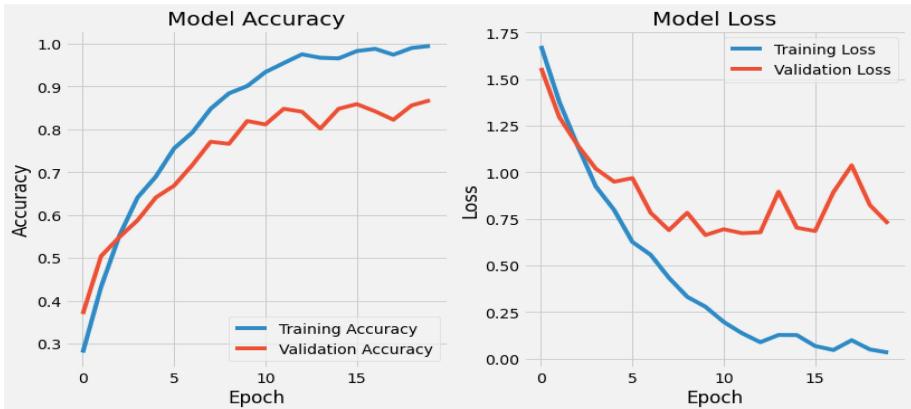


**Fig. 5.** Neural Networks Architecture

In Fig. 5, each node represents a neuron and the links depict the intricate flow of data within the network. The input layer contains 16 nodes ( $x_1$  to  $x_{16}$ ), corresponding to the 16 features in our dataset. The two hidden layers contain 6 and 4 neurons respectively, with activation functions denoted as  $h_i^{(1)}$  and  $h_i^{(2)}$ . The output layer comprises four nodes ( $\hat{y}_1$  to  $\hat{y}_4$ ), each delivering the network's binary output for each robot, indicating either 0 for "Less Appropriate" or 1 for "Most Appropriate."

## 8.1 Model Training and Evaluation

The neural network model was built using the TensorFlow Keras framework, trained over 20 epochs with batches of 64 to enhance computational efficiency and learning accuracy. The data was divided into thirds for training, testing, and validation. Extensive preprocessing, including outlier removal and noise reduction, was performed to ensure data quality.



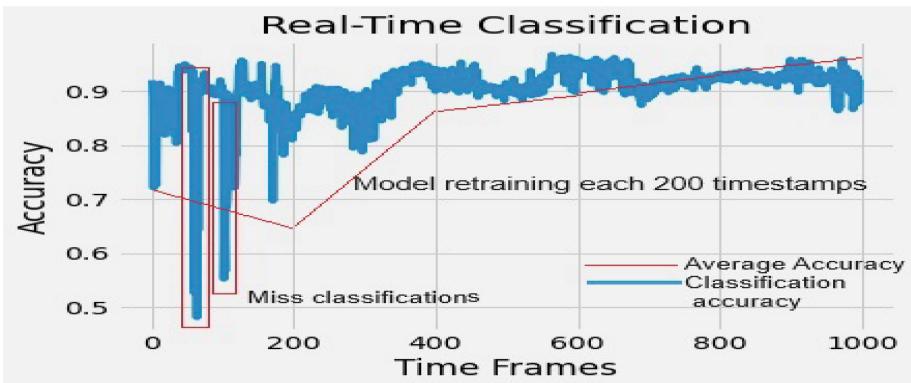
**Fig. 6.** Model Training and Evaluation

As shown in Fig. 6, the model, tasked with detecting dynamic movements, achieved a final accuracy of 94% by the 19th epoch and a validation accuracy of 93% by the 20th epoch. The model also scored an F1 of 93%, with an Area Under the Precision-Recall Curve (AUC-PR) and an Area Under the Receiver Operating Characteristic Curve (AUC-ROC) of 93% and nearly 99%, respectively.

These results highlight the model's efficiency in real-time applications, supported by advanced metrics like AUC-PR and AUC-ROC, which offer insights into its performance in handling class imbalances and predicting the minority class in binary classification tasks.

## 8.2 Model Testing and Validation Using Incremental Learning in Real Time

Incremental learning is a machine learning paradigm tailored for environments where data continuously evolves, requiring algorithms that can adapt without full retraining. This approach is crucial when immediate updates are necessary to reflect new data patterns and insights. Unlike traditional batch learning that requires the entire dataset for training, incremental learning updates a model incrementally. This is particularly beneficial in systems like surveillance and autonomous vehicles, where data storage constraints prevent keeping old information, and the system must quickly adapt to new developments [31].



**Fig. 7.** Real Time Classification Accuracy and Incremental Learning Efficiency

In our implementation, four systems connected to a broker stream data dynamically at 0.01 s intervals. This setup, involving a continuous flow of large data volumes, necessitates that our deep learning model on the host adapts not only to historical data but also to new, incoming data over time.

During practical tests, these four systems were evaluated using a dynamically moving red ball to simulate complex, real-world situations. To facilitate incremental learning, we introduced a feedback loop that integrates new data with historical data. This system is configured to automatically retrain every 200 timestamps, merging newly gathered data from the broker with the existing dataset, thereby continually enhancing its predictive accuracy. Initially, in the phase from 0 to 200 timestamps without historical data, frequent misclassifications were observed with a decreasing rapidly in accuracy to be less than 50%, as illustrated in Fig. 7. These were expected as the model was initially trained on a limited dataset.

However, after each retraining phase, which incorporates new data into the historical dataset, the accuracy of the model's classifications improved progressively. Misclassifications gradually decreased until the model stabilized and

achieved convergence. This iterative retraining process enables the model to adapt more accurately to the dynamic environment, progressively reducing errors and enhancing performance. After 1000 timestamps, corresponding to five rounds of retraining with newly incoming data from the broker, the model's accuracy reached 92%. This improvement underscores the effectiveness of continuously integrating fresh data into the training process.

This method underscores the effectiveness of incremental learning in dynamic settings, where the ability to adapt quickly and efficiently is paramount. It ensures that our models are not only theoretically sound but also practically viable, capable of handling real-world complexities with increasing precision.

## 9 Conclusion

This study showcased an innovative decentralized decision-making framework employing Machine-to-Machine (M2M) communication, facilitated by the MQTT Mosquitto broker, integrating various autonomous systems such as humanoid robots and drones. The implementation centralized data processing on a host equipped with a deep learning model that analyzed incoming sensor data. This setup enabled the autonomous allocation of tasks to the most appropriate machines based on a continuous monitoring interval of 0.01 s, thereby enhancing efficiency and diminishing the need for human oversight.

At the core of the system was a sophisticated deep learning model, designed to evaluate data and make real-time decisions to identify the optimal system for task execution.

The model's training encompassed 20 epochs, processing data in batches of 64 to optimize both efficiency and accuracy. The dataset was evenly divided into three segments: 33% each for training, testing, and validation. By the 19th epoch, the model demonstrated an impressive accuracy of 94%, indicating rapid convergence. Validation accuracy peaked at 93% by the 20th epoch, confirming its robustness and adaptability. Additionally, the model exhibited exceptional predictive capabilities, achieving a 93% F1 score, 93% AUC-PR, and nearly 99% AUC-ROC.

Incremental learning was utilized to test the model's real-life efficacy, assessing the systems' precision in tracking and predicting under dynamically changing conditions. A feedback loop was established to integrate new data with historical data continuously, enhancing the model's accuracy. This system automatically retrained every 200 timestamps, merging newly acquired data with existing data. Early tests revealed challenges, notably frequent misclassifications and a marked decrease in accuracy when limited historical data was available. However, following successive retraining cycles that incorporated both new and old data, a substantial improvement in accuracy was noted. After five retraining cycles, corresponding to 1000 timestamps, the model's accuracy notably increased to 92%. Our model and vision has accomplished almost the accuracy and has the ability to be improved and developed over the time by adding more tasks, features the ail is to get a simple task to classify between the systems.

## References

1. Kim, G., Kang, S., Park, J., Chung, K.: An MQTT-based context-aware autonomous system in oneM2M architecture. *IEEE Internet Things J.* **6**(5), 8519–8528 (2019). <https://doi.org/10.1109/JIOT.2019.2919971>
2. Park, S., Crespi, N., Park, H., Kim, S.-H.: IoT routing architecture with autonomous systems of things. In: 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea (South), pp. 442–445 (2014). <https://doi.org/10.1109/WF-IoT.2014.6803207>
3. Xu, X., He, H., Yang, S.X.: Machine learning with applications to autonomous systems. *Math. Probl. Eng.* **2015**, 385028 (2015)
4. Reddy, K., Praveen Rajalakshmi, M., Adarsh, K., Thakur, R.N., Shakila, B.: Autonomous vehicles and intelligent automation: applications, challenges, and opportunities. *Mob. Inf. Syst.* **2022**, no. 7632892 (2022)
5. Diène, B., Rodrigues, J.J.P.C., Diallo, O., Ndoye, E.H.M., Korotaev, V.V.: Data management techniques for internet of things. *Mech. Syst. Signal Process.* **138**, 106564 (2020). ISSN 0888-3270. <https://doi.org/10.1016/j.ymssp.2019.106564>
6. Chen, J., Abbod, M., Shieh, J.-S.: Integrations between autonomous systems and modern computing techniques: a mini review. *Sensors* **19**, 3897 (2019). <https://doi.org/10.3390/s19183897>
7. Lakshmann, K., et al.: A review on deep learning techniques for IoT data. *Electronics* **11**(1604) (2022). <https://doi.org/10.3390/electronics1101604>
8. Tabassum, A., Erbad, A., Mohamed, A., Guizani, M.: Privacy-preserving distributed IDS using incremental learning for IoT health systems. *IEEE Access* **9**, 14271–14283 (2021). <https://doi.org/10.1109/ACCESS.2021.3051530>
9. Malek, Y.N., et al.: On the use of IoT and big data technologies for real-time monitoring and data processing. *Procedia Comput. Sci.* **113**, 429–434 (2017). <https://doi.org/10.1016/j.procs.2017.08.281>
10. Liu, Y., Wang, J., Li, J., Niu, S., Song, H.: Class-incremental learning for wireless device identification in IoT. *IEEE Internet Things J.* **8**(23), 17227–17235 (2021). <https://doi.org/10.1109/JIOT.2021.3078407>
11. Batth, R.S., Nayyar, A., Nagpal, A.: Internet of robotic things: driving intelligent robotics of future - concept, architecture, applications and technologies. In: 2018 4th International Conference on Computing Sciences (ICCS), Jalandhar, India, pp. 151–160 (2018). <https://doi.org/10.1109/ICCS.2018.00033>
12. Kabir, R., Watanobe, Y., Islam, M.R., Naruse, K., Rahman, M.M.: Unknown object detection using a one-class support vector machine for a cloud–robot system. *Sensors* **22**, 1352 (2022). <https://doi.org/10.3390/s22041352>
13. Andronie, M., et al.: Big data management algorithms, deep learning-based object detection technologies, and geospatial simulation and sensor fusion tools in the internet of robotic things. *ISPRS Int. J. Geo-Inf.* **12**, 35 (2023). <https://doi.org/10.3390/ijgi12020035>
14. Chen, L.-B., Huang, X.-R., Chen, W.-H., Pai, W.-Y., Huang, G.-Z., Wang, W.-C.: Design and implementation of an artificial intelligence of things-based autonomous mobile robot system for cleaning garbage. *IEEE Sens. J.* **23**(8), 8909–8922 (2023). <https://doi.org/10.1109/JSEN.2023.3254902>
15. Toma, C., Popa, M., Iancu, B., Doinea, M., Pascu, A., Ioan-Dutescu, F.: Edge machine learning for the automated decision and visual computing of the robots, IoT embedded devices or UAV-drones. *Electronics* **11**, 3507 (2022). <https://doi.org/10.3390/electronics11213507>

16. Verbeet, R., Rieder, M., Kies, M., Kies, M.: Realization of a Cooperative Human-Robot-Picking by a Learning Multi-Robot-System Using BDI-Agents (2019). SSRN. <https://ssrn.com/abstract=3502934>. <https://doi.org/10.2139/ssrn.3502934>
17. Mukhandi, M., Portugal, D., Pereira, S., Couceiro, M.S.: A novel solution for securing robot communications based on the MQTT protocol and ROS. In: 2019 IEEE/SICE International Symposium on System Integration (SII), Paris, France, pp. 608–613 (2019). <https://doi.org/10.1109/SII.2019.8700390>
18. Garcia, C.A., Montalvo-Lopez, W., Garcia, M.V.: Human-robot collaboration based on cyber-physical production system and MQTT. Procedia Manuf. **42**, 315–321 (2020). ISSN 2351-9789. <https://doi.org/10.1016/j.promfg.2020.02.088>
19. Mazhar, M.S., et al.: Forensic analysis on internet of things (IoT) device using machine-to-machine (M2M) framework. Electronics **11**, 1126 (2022). <https://doi.org/10.3390/electronics11071126>
20. Kulk, J., Welsh, J.: A low power walk for the NAO robot. In: Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2008), Pasadena, California (2008)
21. Umar Suleman, M., Awais, M.M.: Learning from demonstration in robots: experimental comparison of neural architectures. Robot. Comput.-Integr. Manufact. **27**(4), 794–801 (2011)
22. Roopaei, M., Rad, P., Jamshidi, M.: Deep learning control for complex and large scale cloud systems. Intell. Autom. Soft Comput. **23**(3), 389–391 (2017). <https://doi.org/10.1080/10798587.2017.1329245>
23. NAO Hardware — NAO Software 1.12 documentation. <http://cmu.edu/>
24. DARwin-Op — e-Manual wiki documentation. <http://robotis.com/>
25. Mavic 2 Pro/Zoom User Manual, v1.4, 2018, DJI. <https://www.dji.com/mavic-2/info#downloads>
26. Sharma, P., Jain, S., Gupta, S., Chamola, V.: Role of machine learning and deep learning in securing 5G-driven industrial IoT applications. Ad Hoc Netw. **123**, 102685 (2021). ISSN 1570-8705. <https://doi.org/10.1016/j.adhoc.2021.102685>
27. Abdellah, A.R., Koucheryavy, A.: Deep learning with long short-term memory for IoT traffic prediction. In: NEW2AN/ruSMART -2020. LNCS, vol. 12525, pp. 267–280. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-65726-0\\_24](https://doi.org/10.1007/978-3-030-65726-0_24)
28. Gamboa, J.: Deep Learning for Time-Series Analysis, University of Kaiserslautern, Kaiserslautern, Germany, [arXiv:1701.01887v1](https://arxiv.org/abs/1701.01887v1) (2017)
29. Kaiser, M., Klingspor, V., Millin, J.R., Accame, M., Wallner, F., Dillmann, R.: Using Machine Learning Techniques in Real-World on Mobile Robots, University of Karlsruhe, Germany, University of Dortmund, Germany, Institute for Systems Engineering & Informatics, Ispra, Italy, University of Genoa, Italy
30. Antunes, M., Gomes, D., Aguiar, R.L.: Towards IoT data classification through semantic features. VL86 DO (2017). <https://doi.org/10.1016/j.future.2017.11.045>
31. Nemade, B., Shah, D.: An efficient IoT based prediction system for classification of water using novel adaptive incremental learning framework. J. King Saud Univ. Comput. Inf. Sci. Part A **34**(8), 5121–5131 (2022). ISSN 1319-1578. <https://doi.org/10.1016/j.jksuci.2022.01.009>



# Explainable Federated Stacking Models with Encrypted Gradients for Secure Kidney Medical Imaging Diagnosis

Sharia Arfin Tanim<sup>1</sup> , Al Rafi Aurnob<sup>1</sup> , Md Rokon Islam<sup>1</sup> ,  
Md Saef Ullah Miah<sup>1</sup> , M. Mostafizur Rahman<sup>2</sup> ,  
and Mufti Mahmud<sup>3,4,5</sup>

<sup>1</sup> Department of Computer Science, American International University-Bangladesh  
(AIUB), Dhaka, Bangladesh  
[saef@aiub.edu](mailto:saef@aiub.edu)

<sup>2</sup> Department of Mathematics, American International University-Bangladesh  
(AIUB), Dhaka, Bangladesh  
[mostafiz.math@aiub.edu](mailto:mostafiz.math@aiub.edu)

<sup>3</sup> Information and Computer Science Department, King Fahd University of  
Petroleum and Minerals,  
Dhahran 31261, Saudi Arabia

<sup>4</sup> SDAIA-KFUPM Joint Research Center for AI, King Fahd University of Petroleum  
and Minerals,  
Dhahran 31261, Saudi Arabia

<sup>5</sup> Interdisciplinary Research Center for Bio Systems and Machines, King Fahd  
University of Petroleum and Minerals,  
Dhahran 31261, Saudi Arabia

[muftimahmud@gmail.com](mailto:muftimahmud@gmail.com), [mufti.mahmud@kfupm.edu.sa](mailto:mufti.mahmud@kfupm.edu.sa)

**Abstract.** In the field of medical data analysis, both privacy retention and model interpretability are of utmost importance. This paper focuses on the vulnerability of kidney medical image analysis through Federated Learning (FL) with privacy-preserving measures and explainable artificial intelligence (XAI). We introduce a new set of proposals termed Federated Stacking Fusion with Encryption (FedStackEncFL), which integrates the predictive capabilities of ResNet-101 and InceptionV3 architectures through a stacking fusion technique. This approach also guarantees the quality and reliability of the features extracted while preserving the federated data's privacy by encrypting it through Cheon-Kim-Kim-Song (CKKS) during Federated Averaging (FedAvg). Moreover, to optimize privacy, during the computation, the method of adding Gaussian gradient noise is used. The efficiency of the developed technique is proven by the complex experimental data proving the efficiency of the humanitarian approach to enhance indicators of model precision, recall, F1-score, and accuracy in Kidney disease diagnosis. Furthermore, the integration of XAI techniques like GradCAM, GradCAM++, and ScoreCAM provides insightful visual explanations, enhancing the interpretability of our model's predictions in medical diagnostics.

**Keywords:** Federated learning · Stacking fusion · Homomorphic encryption · Gradient noise · Explainable AI · Kidney disease

## 1 Introduction

Federated Learning (FL) is being hailed for changing the medical field by enabling collaborative model training across multiple institutions without compromising data privacy [15]. FL allows the knowledge base generated in a specific area to be updated from a much larger base of data, thus making the input to predictive models more accurate and generalizable. It can be particularly useful in healthcare contexts including disease diagnosis, prognosis, and individual treatment. The integration of FL in medical research facilitates large-scale studies and accelerates advancements in healthcare by utilizing diverse datasets that were previously siloed [9].

The importance of Federated Learning in the medical sector cannot be overstated. The adoption of centralized machine learning processes means that there is data gathering in one single location, and this is a major disadvantage due to loss of privacy and high complications in data handling [3]. FL also has the advantage of letting the data stay on servers in the local network which minimizes the chances of exposure and lies compliance with strict healthcare standards such as HIPAA and GDPRv [3, 21]. In addition, it also makes the FL gain a lot more diverse datasets from different populations and institutions to make the medical model more sound. This collaborative approach not only fosters innovation but also ensures that AI-driven healthcare solutions are equitable and broadly applicable [16].

Despite its advantages, FL faces challenges [15], prominently among them being gradient leakage [29]. During FL training, client devices compute gradients on local data and transmit them to a central server. While these gradients get accumulated for making updates towards a model, they can otherwise leak specific details about what client data some of them contain due to their direct relation with the local samples. To reduce this risk, Gaussian noise is often employed [18, 31]. By adding carefully calibrated Gaussian noise to gradients before transmission. In FL systems, differential privacy is provided by stemming gradients with added noise from Gaussian distribution before transmitting them through communication channels. This technique makes sure that the inputs of each individual are masked so that the clients' information will not be revealed specifically but without affecting the model essential for providing solutions to the clients. Gaussian noise thus serves as a crucial tool in maintaining privacy while enabling collaborative model training in FL setups. Homomorphic encryption also enhances FL security by enabling computations on encrypted data without decryption further safeguarding patient privacy and data integrity [13]. Additionally, the complexity of federated models necessitates the incorporation of Explainable AI (XAI) to make their decision-making processes transparent and understandable to clinicians [22]. XAI techniques help demystify the predictions made by these models, fostering trust and facilitating their integration

into clinical workflows. By addressing these challenges, FL can become a cornerstone of AI-driven medical advancements, combining security, accuracy, and transparency [16, 18, 19].

In this study, we put forward a security framework for decentralized medical systems that utilizes Federated Learning (FL) principles for privacy and compliance. Our framework is designed to facilitate the collaborative training of machine learning models across multiple institutions without requiring the exchange of sensitive patient data. In this way, we ensure data localization with the help of efficient encryption methods by providing high levels of security to minimize the chances of leakage and improve of the collaborative learning process. The contributions of this study are as follows.

1. We present a custom model that adopts a stacking fusion methodology of melding different machine learning models to improve the predictive capability. This model can use several classifiers with different features and provide more sufficient details than the single classifiers to predict the results.
2. Our framework uses homomorphic encryption to protect data privacy keeping computation on encrypted data without divulging the information. Additionally, integrating Gaussian noise adds an extra layer of privacy protection by obscuring individual gradients during transmission to the central server.
3. To enhance the interpretability of our model, we employ Explainable AI (XAI) methodologies. These techniques give insight into the processes involved in selecting criteria in complicated federated structures, thus, enabling clinicians or researchers to be trusted with the model's decisions.

The remaining part of this paper is structured as follows: Sect. 2 provides insights into the previous literature based on this subject area. Section 3 will explain the incorporating methodology to be used in this study accompanied by appropriate diagrams and tables. Performance analysis and comparison are presented in Sect. 4. Finally, Sect. 5 summarizes the paper and mentions the future research development direction.

## 2 Related Work

Federated Learning (FL) has emerged as a transformative approach in the medical sector, enabling collaborative model training across multiple institutions without the need to share patient data. This method addresses critical privacy concerns and leverages diverse datasets to enhance model robustness and accuracy. In breast cancer detection, various studies have demonstrated the efficacy of FL. For instance, Nguyen Tan et al. [27] utilized a transfer learning approach within an FL framework to classify breast cancer, achieving high accuracy by combining local data from multiple medical institutions. Similarly, Bercea et al. [6] proposed FedDis, a disentangled federated learning model for unsupervised brain pathology segmentation, highlighting the versatility of FL in handling complex medical imaging tasks. These approaches underscore the potential of FL to

revolutionize medical diagnostics by facilitating large-scale, privacy-preserving collaborative research.

To address the privacy challenges inherent in FL, homomorphic encryption (HE) has been employed as a robust solution [20]. HE allows computations to be performed on encrypted data, ensuring that sensitive patient information remains confidential throughout the training process. This method is particularly valuable in the medical domain, where data privacy is paramount. Recent advancements have integrated HE with FL to enhance security without compromising model performance. Almufareh et al. [4] discussed a federated learning approach that incorporates homomorphic encryption to ensure privacy while predicting breast cancer, demonstrating that HE can effectively mitigate data leakage risks. The combination of FL and HE presents a compelling paradigm for secure, collaborative machine learning in healthcare, enabling institutions to build powerful predictive models while adhering to stringent privacy regulations.

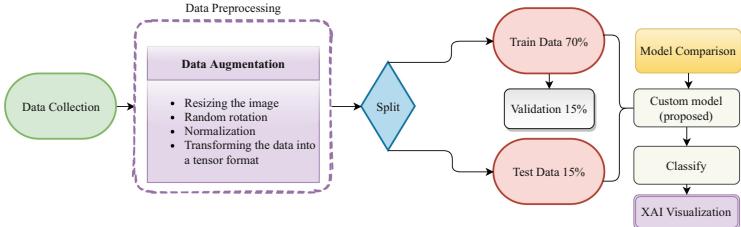
Explainable AI (XAI) in Federated Learning is gaining traction as a means to enhance transparency and trust in AI-driven medical diagnostics [5]. XAI techniques aim to make the decision-making processes of complex models understandable to clinicians, thereby facilitating better integration of AI tools in clinical workflows. In the context of FL, XAI is particularly challenging yet crucial, as it involves interpreting models trained on distributed, heterogeneous datasets. Recent work by Agbley et al. [1] emphasized the importance of XAI in federated learning frameworks, particularly for invasive carcinoma detection, where understanding the model's rationale can significantly impact clinical decisions. By incorporating XAI, FL models not only provide accurate predictions but also offer insights into the underlying patterns and features driving these predictions, thus fostering greater acceptance and reliability of AI systems in the medical community. Table 1 summarizes various Federated Learning approaches for medical image detection, highlighting the models used and their key findings.

**Table 1.** Summary of Federated Learning Approaches for Medical Data Detection

Year	Used Model	Key findings
2022 [1]	(Late fusion +FL) Gabor Net and ResNet for feature selection	Combining Gabor and residual neural network features yielded the best performance
2022 [17]	Weakly supervised learning + FL	Enabling accurate weakly supervised deep learning without sharing data directly
2023 [30]	AFEI (adaptive optimized vertical federated learning based DNN method)	Achieved 6.5% higher prediction accuracy on average
2023 [27]	Federated Learning (MILP, CNN), Transfer Learning (MobileNet, DenseNet-121, Xception, ResNet50)	FL with CNNs can classify mammogram abnormalities with 100% recall and 99.804% AUC using pre-trained MobileNet features
2023 [4]	DNN + FL	Achieved higher accuracy with proposed model
2023 [6]	Federated Learning combined with Machine Learning	The FL-driven system enhances diagnostic model accuracy and generalizability across multiple institutions

### 3 Methods and Materials

Our methodology comprises four principal components: data collection and pre-processing, stacking fused model building, homomorphic encryption, and XAI for model interpretation. Figure 1 provides an illustrative overview of our proposed methodology.



**Fig. 1.** Illustration depicting the workflow of our proposed methodology for Federated Learning (FL) with encryption.

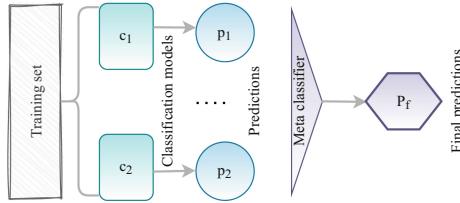
#### 3.1 Data Collection and Preprocessing

The dataset was gathered via picture archiving and communication systems (PACS) at various hospitals in Dhaka, Bangladesh [11], where patients had already been diagnosed with a kidney tumor, cyst, normal or stone results. We removed each patient's PII and meta data from the Dicom photos and converted them to a compact jpg file format. Following conversion, each image finding was double-checked by a radiologist and a medical technician to ensure that the data was correct. After that, we only included the kidney tumor and the normal stage out of all of these stages. Our created dataset has 10,000 images, of which 5000 are normal kidney images and 5000 are tumor images.

After collecting the data, we moved on to our data preprocessing part. In this step, we have done some sort of feature engineering, such as resizing the image, random rotation, random horizontal flip, normalizing the data ( $\text{mean} = [0.485, 0.456, 0.406]$ ,  $\text{std} = [0.229, 0.224, 0.225]$ ), and finally transforming the data into a tensor format. After that, we split the dataset into train, test, and validation data with a ratio of 0.7, 0.15, and 0.15, respectively. Using a data loader function, we loaded our dataset with a batch size of 8.

#### 3.2 Stacking Fused Model Building

In this section, we outline the construction of our custom model using a stacking fusion technique shown in Fig. 2, which combines ResNet-101 [25] and InceptionV3 [8] architectures. Stacking fusion involves leveraging the predictive capabilities of multiple machine learning models to enhance the overall performance of the system.



**Fig. 2.** Visualization of the Stacking Fusion Overview Diagram.

The selection of both model were chosen for the stacking fusion due to their different strengths in feature extraction. ResNet-101, with its 44.5 million parameters, is particularly effective at learning deep hierarchical features. The residual function can be represented as:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \quad (1)$$

InceptionV3, on the hand, is good at identifying multiscale features because it uses inception modules and most of these modules perform convolutions of different sizes. This approach is summarized by:

$$\text{Output} = \text{Concat}(\text{Conv}_{1 \times 1}, \text{Conv}_{3 \times 3}, \text{Conv}_{5 \times 5}, \text{Pooling}) \quad (2)$$

Stacking fusion involves combining the two models in a way that the strengths of both are used to develop a wider feature set in order to improve the predictive accuracy of the models. The fusion process can be mathematically expressed as:

$$\mathbf{z} = \text{Concat}(\mathbf{f}_{\text{ResNet-101}}(\mathbf{x}), \mathbf{f}_{\text{InceptionV3}}(\mathbf{x})) \quad (3)$$

where  $\mathbf{f}_{\text{ResNet-101}}(\mathbf{x})$  and  $\mathbf{f}_{\text{InceptionV3}}(\mathbf{x})$  are the feature vectors extracted from each model, and  $\mathbf{z}$  is the concatenated vector used for further processing.

First, we use a ResNet-101 that refers to a deep convolutional neural network (CNN) with high efficiency in image classification tasks. It can give comprehensive information about medical images and its differential diagnostics additional to the exact prognosis because of its high disposable features of architecture. We also incorporate InceptionV3 which is another CNN architecture widely known for the quality performance it offers in tasks like image recognition. We thus employ multiple parallel convolutional pathways of InceptionV3 architecture in parallel to ResNet-101 outcome to capture diverse features from the input images.

The outputs of ResNet-101 and InceptionV3 are then combined using a stacking fusion approach. This fusion technique aggregates the predictions of both models, effectively leveraging the complementary strengths of each architecture to improve the overall predictive accuracy. Through the utilization of stacking fusion, our custom model not only benefits from the robust feature extraction capabilities of ResNet-101 and InceptionV3 but also achieves enhanced performance by integrating their predictive outputs. The synergy of these approaches

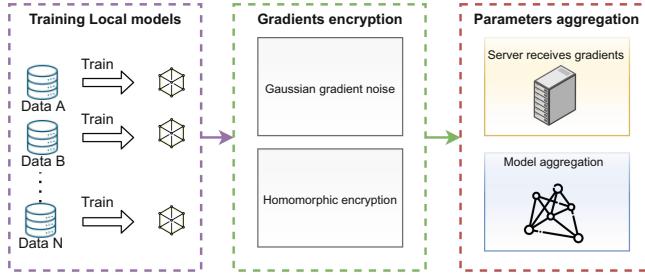
results in the creation of a more effective model that can handle medical image analysis and ultimately arrive at a diagnosis.

### 3.3 Federated Privacy-Preserving Learning

In our implementation, we utilized the CKKS (Cheon-Kim-Kim-Song) encryption [14] scheme to safeguard data privacy within the context of Federated Averaging (FedAvg) [23] during Federated Learning (FL). The CKKS scheme is a type of homomorphic encryption that is known for its capacity to perform approximate calculations on encrypted information, making it particularly suitable for scenarios where complex computations are necessary while at the same time ensuring data confidentiality. Employing the CKKS encryption scheme, we encrypted the model updates (gradients) sourced from participating clients before aggregating them to compute the global model update. The CKKS scheme empowers addition, multiplication, and scaling operations on encrypted values, allowing us to execute the essential computations securely without revealing the underlying data. The procedure for computing the average of encrypted updates  $Enc(g)$  using the CKKS encryption scheme closely mirrors the standard Federated Average algorithm. However, operations are conducted on encrypted values. Equation (4) explains how to perform the aggregation of the model updates securely while using the CKKS encryption scheme under the FedAvg environment for Federated Learning.

$$Enc(g) = \frac{1}{n} \sum_{i=1}^n (Enc(w_i) \cdot Enc(\theta_i)) + Enc(b) \quad (4)$$

Here,  $Enc(g)$  denotes the encrypted global model update,  $n$  signifies the count of participating clients,  $Enc(w_i)$  and  $Enc(\theta_i)$  represent the encrypted weight and gradient updates from each client  $i$ , respectively, and  $Enc(b)$  stands for the encrypted bias term. The symbols  $\cdot$  and  $+$  indicate the homomorphic multiplication and addition operations, respectively, while the summation symbol  $\sum$  denotes the aggregation of encrypted updates from all participating clients. Utilizing CKKS encryption in FedAvg ensures the secure aggregation of model updates from multiple clients while safeguarding sensitive information. This makes it easy to apply federated learning in areas that require privacy such as healthcare, finance, and decentralized applications. To further enhance privacy protection, Gaussian gradient noise [31] is often added to the gradients before encryption and transmission is calculated using the Eq. 5.



**Fig. 3.** Gradient Sparsification Defense Framework.

$$\tilde{\theta}_i = \theta_i + \mathcal{N}(0, \sigma^2) \quad (5)$$

The noisy gradient is denoted as  $\tilde{\theta}_i$ , where  $\theta_i$  represents the original gradient. The term  $\mathcal{N}(0, \sigma^2)$  signifies Gaussian noise with a mean of 0 and variance  $\sigma^2$ . This additional step also reduces the risk of information leakage by converting the gradient values into a controlled random distribution which prevents inferences from individual client data. Figure 3 illustrates the gradient-based defense method which complements the privacy-preserving techniques for data leakage in FL. This framework visually represents the integration of gradient sparsification and encryption methods in federated learning environments, highlighting their role in enhancing data privacy and security. Concisely, using CKKS encryption to encrypt the shared model update in FedAvg guarantees the privacy and confidentiality of the transferred information in the learning process. Alongside Gaussian gradient noise, this approach helps in federated learning across domains like healthcare, finance, and decentralized apps while maintaining high privacy and security standards allowing for the collaborative training of models without sharing data.

### 3.4 XAI for Model Interpretation

The XAI techniques used in our model interpretation include Gradient-weighted Class Activation Mapping (GradCAM) [26], Score-weighted Class Activation Mapping (ScoreCAM) [28], and Gradient-weighted Class Activation Mapping++ (GradCAM++) [7]. These techniques generate class-specific localization maps highlighting important regions in the input image for a given prediction class.

$$L_{GradCAM}^c = ReLU \left( \sum_k \alpha_k^c A_k \right) \quad (6)$$

In Eq. 6,  $L_{GradCAM}^c$  represents the GradCAM localization map for class  $c$ ,  $\alpha_k^c$  denotes the importance of the  $k$ -th feature map for class  $c$ , and  $A_k$  represents the  $k$ -th activation map from the last convolutional layer.

$$L_{ScoreCAM}^c = \text{ReLU} \left( \sum_k \frac{\exp(\alpha_k^c)}{\sum_i \exp(\alpha_i^c)} A_k \right) \quad (7)$$

In Eq. 7,  $L_{ScoreCAM}^c$  denotes the ScoreCAM localization map for class  $c$ . Here, the importance scores adjusted using an exponential weighting.

$$L_{GradCAM++}^c = \text{ReLU} \left( \sum_k \alpha_k^c \cdot \text{ReLU}(A_k) \right) \quad (8)$$

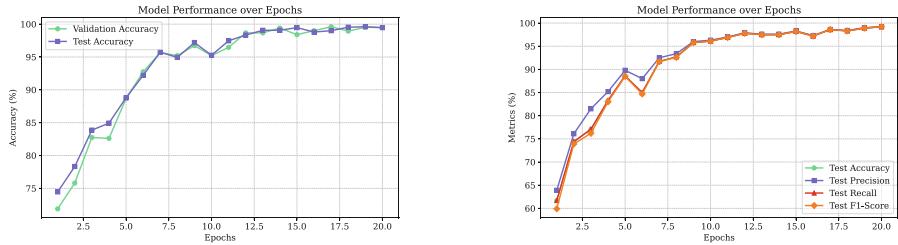
In Eq. 8,  $L_{GradCAM++}^c$  represents the GradCAM++ localization map for class  $c$ . The notation and parameters  $\alpha_k^c$  and  $A_k$  remain consistent across all methods, but GradCAM++ introduces an additional ReLU operation on the activation maps to improve localization accuracy. How each explainable AI (XAI) technique, GradCAM, ScoreCAM, and GradCAM++ computes class-specific localization maps. These maps highlight important regions in the input image for a given prediction class, aiding in the interpretation of our model's predictions. The techniques share common elements, such as the use of feature map importance weights ( $\alpha_k^c$ ) and activation maps ( $A_k$ ), but differ in their approaches to aggregating and transforming these elements to generate the final localization maps.

## 4 Experimental Results

The performance of the model's metrics is shown in Fig. 4. In this figure, we have presented a plot of the test accuracy, precision, recall, and f1-score. Initially, the accuracy is around 60% and gradually increases to 99.20%. This implies the model's ability to categorize the classes correctly. Likewise, accuracy rises and achieves a maximum of 99.21%. But in the 7<sup>th</sup> epoch, there is some small variation, but overall, the result increases gradually. In this graph, we also find that the recall and F1-score show a notable improvement that exceeds 99%. Similarly, we can see that in the 7<sup>th</sup> epoch, these metrics are falling, but thereafter, the values bounce back. In 11–20 epochs, the metrics level first remains flattened for a while, and after that, it slowly improves the results, demonstrating that the model is stable and getting better to be trained for extended periods of time.

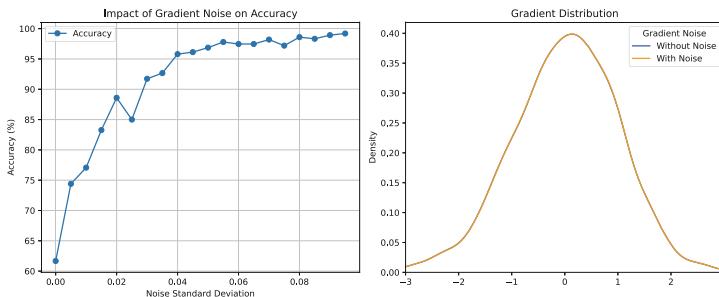
In general, this figure shows a significant enhancement in concerned parameters, such as F1-score, accuracy, precision, and recall, as the epochs of the training sessions increase in the automated model. The above findings suggest that the model has been well-trained and provides accurate recommendations once exposed to adequate training.

The Fig. 5 demonstrates the impact of varying levels of gradient noise on model performance and gradient distributions during training. In the left panel, the accuracy (%) of a model trained with gradient noise ranging from 0 to 0.095 standard deviations is depicted. As noise levels increase, accuracy initially improves, peaking at approximately 99.2% with a noise level of 0.095, before stabilizing and slightly decreasing beyond 0.05. This indicates a balance between



**Fig. 4.** Validation and Test Accuracy of Global Model (on left), Model Performance Metrics of Global Models (on right).

regularization benefits and noise interference. The right panel shows Kernel Density Estimation (KDE) plots of gradient distributions. The blue curve represents gradients without noise, displaying a smoother distribution centered around zero, while the orange curve, with added noise (0.01 standard deviation), exhibits a broader and more spread-out distribution. Adding noise increases gradient variability, potentially enhancing model generalization by mitigating overfitting.



**Fig. 5.** Impact of Gradient Noise on Model Performance and Gradient Distribution.

#### 4.1 Performance Evaluation and Comparative Analysis

In this comparative analysis of methods for kidney disease detection, the proposed Federated Learning with fused stack and secured encryption named as (FedStackEncFL) performs best among all in terms of classification stats. FedStackEncFL achieves exceptional precision, recall, F1-score, and accuracy, scoring consistently high across all metrics (Precision: 98.21%, Recall: 99.20%, F1-score: 99.19%). This goes beyond other strategies including the LR model with chi-square selection, Transfer Learning (TL) with VGG16 and Layer-Wise Relevance Propagation (LRP), Proportionally Fair Federated Learning (PFFL), Fuzzy logic, and ABPNN. However, as can be seen

from Table 2, FedStackEncFL has better comparative performance in test classifications.

**Table 2.** Classification Performance Comparison of Various Methods for Kidney Disease Detection Using Different Architectures

Ref.	Method	Precision	Recall	f1-score	Accuracy
[24]	LR model + chi-square feature selection ( $K > 14$ ), where K is number of features	98%	97%	98%	97.5%
[2]	Transfer Learning (TL) with VGG16 and Layer-Wise Relevance Propagation (LRP) for Explainable AI enhances kidney stone detection accuracy in KUB X-ray images	97.39%	98.45%	98.00%	97.41%
[10]	Proportionally Fair Federated Learning	-	-	-	84.36%
[12]	Fuzzy logic for integrating outputs from the Adaptive Backpropagation Neural Network (ABPNN)	97.50%	96.01%	-	98.5%
<b>Proposed</b>	FL with fused stack and secured cryption(FedStackEncFL)	99.21%	99.20%	99.19%	99.20%

The proposed method constitutes a centralized system environment and does not possess decentralization as some methods do. This centralized approach may cause data privacy issues in contrast to methods that use decentralized frameworks. In contrast, FedStackEncFL employs Explainable AI (XAI) to elucidate the model's decisions, enhancing its interpretability more effectively compared to methods that incorporate XAI to a lesser extent presented in Table 3.

**Table 3.** Comparative Analysis of Decentralized system, Privacy Concerns, Explainable AI, and Sensitivity Analysis(SA) in Kidney Disease Research

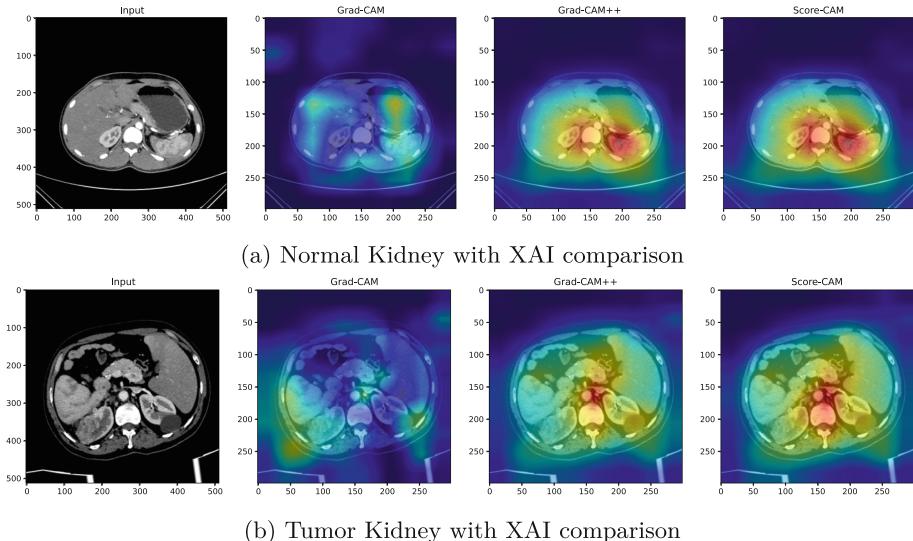
Authors	Decentralized system	Privacy Concerns	XAI	SA
Poonia, R.C. et al. (2022)	✗	✓	✗	✗
Ahmed et al. (2024)	✗	✓	✓	✗
Hosseini et al. (2023)	✓	✗	✗	✗
KR et al. (2024)	✗	✓	✗	✗
<b>Proposed</b>	✓	✗	✓	✓

The entire dataset is formed of 10,000 images. The fused model based on ResNet-101 and InceptionV3 includes about 68. 3 million parameters, which entails roughly 15. 3 GFLOPS. For 20 epochs, the training computation cost is estimated to be approximately  $2.12 \times 10^{12}$  FLOPS. The computation costs for validation and testing are calculated to be  $4.59 \times 10^{10}$  FLOPS and  $2.30 \times 10^{10}$  FLOPS, respectively. With the use of homomorphic encryption, the computation cost is increased by a factor of 10, resulting in an adjusted total computation

cost of  $2.18 \times 10^{13}$  FLOPS. The model's weight is approximately 273.3 MB per client, and the encrypted weight size ranges from 1.37 GB to 2.73 GB per client. Over 20 communication rounds and involving 4 clients, the total communication cost is estimated to be approximately 21.86 GB for plaintext weights and ranges from 109.6 GB to 218.4 GB for encrypted weights.

## 4.2 Explaining AI Model Decisions in Kidney Image Analysis

It is critical to understand which regions of a kidney image relates the most to the model's predictions while analyzing them. Our study compares three such approaches, Grad-CAM [26], Grad-CAM++ [7], and Score-CAM [28] to the raw input photos. Grad-CAM generates a coarse localization map highlighting the image's significant regions by using the gradients of any target idea that flows into the final convolutional layer. Grad-CAM++ is an extension of Grad-CAM that improves localization by taking into account the pixels' positive and negative impacts. Score-CAM does not use gradients, instead relies on the model's score (output probability) for each class to determine the relevance of each region. This technique is very effective for creating more understandable and dependable visual explanations. Figure 6a illustrates the output image for grad-cam, which reveals greater regions of interest but may contain less important areas. However, GradCam++ provides more exact localization, improving the interpretability of critical sections. The most interpretable visualizations are produced by focusing on the regions that have a major impact on the model's output scores using scorecam.



**Fig. 6.** Visualization results of Grad-CAM, Grad-CAM++, and Score-CAM for kidney images.

In Fig. 6b, the raw pictures show kidney tissue with visible tumor areas. The heatmap in Grad-CAM displays wide areas of activity that include both cancer and non-tumor regions. While it identifies regions connected with the tumor, it may also highlight areas that are unrelated to the pathology. Grad-CAM++ provides more refined and exact visualizations than Grad-CAM. The highlighted regions are more specifically concentrated around cancer, making it easier to discriminate between tumors and non-tumor tissues. Finally, Score-CAM provides the clearest and most concentrated visualizations, stressing the regions that have the most impact on the model’s cancer categorization accuracy. Score-CAM, in particular, stands out for its capacity to provide highly focused and meaningful visualizations, making it an effective choice for medical images. The IoU analysis support and showed that Score-CAM’s superior performance, achieving an 81.3% IoU score, compared to 74.5% for Grad-CAM++ and 68.2% for Grad-CAM. This shows that Score-CAM achieves higher segmentation score and produces better localization of the tumor in kidney images compared to ground-truth.

### 4.3 Limitations

Despite its strengths, the proposed FedStackEncFL method has several limitations. The combination of stacking fusion with ResNet-101 and InceptionV3, along with CKKS encryption and Gaussian gradient noise, significantly increases computational complexity, leading to slower processing and higher resource demands. While Gaussian noise improves privacy, it may reduce precision in medical images, where small differences are crucial for accurate diagnosis. Additionally, as the number of clients increases, managing large encrypted datasets can cause bottlenecks and delays in data aggregation. Although advanced XAI techniques like GradCAM, GradCAM++, and ScoreCAM enhance model interpretability, they may fall short in explaining decisions in cases where distinguishing between healthy and pathological tissues is particularly challenging due to the complexities of medical data.

## 5 Conclusion and Future Work

This study presents the FedStackEncFL framework, which combines federated learning with stacking fusion and encryption techniques to improve privacy preservation and model interpretability in medical data analysis. Using ResNet-101 and InceptionV3, our method achieves superior performance in kidney disease classification, with the highest precision, recall, F1-score, and accuracy. The use of CKKS homomorphic encryption ensures secure model updates while protecting patient privacy, and Gaussian gradient noise enhances data confidentiality during training. XAI techniques further increase transparency, fostering trust among healthcare professionals.

Future research should explore alternative homomorphic encryption methods and improved differential privacy techniques to strengthen security in

medical federated learning. This includes adapting federated algorithms for diverse healthcare data storage systems and optimizing internode communication. Advancing XAI to provide more detailed insights into medical imaging decisions and integrating blockchain for enhanced data security in decentralized healthcare systems are also promising directions.

## References

- Agbley, B., et al.: Federated learning-based detection of invasive carcinoma of no special type with histopathological images. *Diagnostics* **12**(7), 1669 (2022)
- Ahmed, F., et al.: Identification of kidney stones in KUB X-ray images using VGG16 empowered with explainable artificial intelligence. *Sci. Rep.* **14**(1), 6173 (2024)
- Ali, M., Naeem, F., Tariq, M., Kaddoum, G.: Federated learning for privacy preservation in smart healthcare systems: a comprehensive survey. *IEEE J. Biomed. Health Inform.* **27**(2), 778–789 (2022)
- Almufareh, M.F., Tariq, N., Humayun, M., Almas, B.: A federated learning approach to breast cancer prediction in a collaborative learning framework. In: *Healthcare*, vol. 11, p. 3185. MDPI (2023)
- Awosika, T., Shukla, R.M., Pranggono, B.: Transparency and privacy: the role of explainable AI and federated learning in financial fraud detection. *IEEE Access* (2024)
- Bercea, C.I., Wiestler, B., Rueckert, D., Albarqouni, S.: Feddis: disentangled federated learning for unsupervised brain pathology segmentation. arXiv preprint [arXiv:2103.03705](https://arxiv.org/abs/2103.03705) (2021)
- Chattopadhyay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-cam++: generalized gradient-based visual explanations for deep convolutional networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 839–847. IEEE (2018)
- Google Cloud: Advanced Guide to Inception v3 (2024). <https://cloud.google.com/tpu/docs/inception-v3-advanced>
- Hamood Alsamhi, S., Hawbani, A., Shvetsov, A.V., Kumar, S.: Advancing pandemic preparedness in healthcare 5.0: a survey of federated learning applications. *Adv. Hum.-Comput. Interact.* **2023**(1), 9992393 (2023)
- Hosseini, S.M., Sikaroudi, M., Babaie, M., Tizhoosh, H.: Proportionally fair hospital collaborations in federated learning of histopathology images. *IEEE Trans. Med. Imaging* (2023)
- Islam, M.N., Hasan, M., Hossain, M.K., Alam, M., Uddin, M.Z., Soylu, A.: Vision transformer and explainable transfer learning models for auto detection of kidney cyst, stone and tumor from CT-radiography. *Sci. Rep.* **12**(1), 1–14 (2022)
- Vineetha, K.R., Maharajan, M.S., Bhagyashree, K., Sivakumar, N.: Classification of adaptive back propagation neural network along with fuzzy logic in chronic kidney disease. *e-Prime Adv. Electr. Eng. Electron. Energy* **7**, 100463 (2024). <https://doi.org/10.1016/j.prime.2024.100463>. <https://www.sciencedirect.com/science/article/pii/S2772671124000457>
- Ku, H., Susilo, W., Zhang, Y., Liu, W., Zhang, M.: Privacy-preserving federated learning in medical diagnosis with homomorphic re-encryption. *Comput. Standards Interfaces* **80**, 103583 (2022)

14. Lee, J.W., et al.: Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access* **10**, 30039–30054 (2022)
15. Li, L., Fan, Y., Tse, M., Lin, K.Y.: A review of applications in federated learning. *Comput. Ind. Eng.* **149**, 106854 (2020)
16. Long, G., Shen, T., Tan, Y., Gerrard, L., Clarke, A., Jiang, J.: Federated learning for privacy-preserving open innovation future on digital health. In: *Humanity Driven AI: Productivity, Well-Being, Sustainability and Partnership*, pp. 113–133. Springer (2021)
17. Lu, M.Y., et al.: Federated learning for computational pathology on gigapixel whole slide images. *Med. Image Anal.* **76**, 102298 (2022)
18. Mahmud, M., et al.: Towards explainable and privacy-preserving artificial intelligence for personalisation in autism spectrum disorder. In: *International Conference on Human-Computer Interaction*, pp. 356–370. Springer (2022)
19. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2063–2079 (2018)
20. Manulis, M., Nguyen, J.: Fully homomorphic encryption beyond IND-CCA1 security: integrity through verifiability. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 63–93. Springer (2024)
21. Mbonihankuye, S., Nkunzimana, A., Ndagiijimana, A.: Healthcare data security technology: HIPAA compliance. *Wirel. Commun. Mob. Comput.* **2019**(1), 1927495 (2019)
22. Nazir, S., Dickson, D.M., Akram, M.U.: Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks. *Comput. Biol. Med.* **156**, 106668 (2023). <https://doi.org/10.1016/j.combiomed.2023.106668>
23. Nilsson, A., Smith, S., Ulm, G., Gustavsson, E., Jirstrand, M.: A performance evaluation of federated learning algorithms. In: *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning*, pp. 1–8 (2018)
24. Poonia, R.C., et al.: Intelligent diagnostic prediction and classification models for detection of kidney disease. *Healthcare* **10**(2) (2022). <https://doi.org/10.3390/healthcare10020371>
25. pytorch: torchvision.models.resnet101 (2024). <https://pytorch.org/vision/main/models/generated/torchvision.models.resnet101.html>. resNet-101 from Deep Residual Learning for Image Recognition
26. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: visual explanations from deep networks via gradient-based localization. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626 (2017)
27. Tan, Y.N., Tinh, V.P., Lam, P.D., Nam, N.H., Khoa, T.A.: A transfer learning approach to breast cancer classification in a federated learning framework. *IEEE Access* **11**, 27462–27476 (2023)
28. Wang, H., et al.: Score-cam: score-weighted visual explanations for convolutional neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 24–25 (2020)
29. Wang, J., Guo, S., Xie, X., Qi, H.: Protect privacy from gradient leakage attack in federated learning. In: *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pp. 580–589. IEEE (2022)

30. Wang, Q., He, M., Guo, L., Chai, H.: AFEI: adaptive optimized vertical federated learning for heterogeneous multi-omics data integration. *Briefings Bioinform.* **24**(5), bbad269 (2023)
31. Wei, W., Liu, L., Wu, Y., Su, G., Iyengar, A.: Gradient-leakage resilient federated learning. In: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), pp. 797–807. IEEE (2021)



# DDFGNN: Dual-Dimensionality Fusion Graph Neural Network for Social Bot Detection

Yuze Bai, Yingjie Sun, Chengping Zheng, and Yizhou Li<sup>(✉)</sup>

School of Cyber Science and Engineering, Sichuan University, Chengdu 610207, China  
 [{baiyuze,sunyingjie,zhengchengping}@stu.scu.edu.cn](mailto:{baiyuze,sunyingjie,zhengchengping}@stu.scu.edu.cn), [liyizhou@scu.edu.cn](mailto:liyizhou@scu.edu.cn)

**Abstract.** With the rapid expansion of social networks, an influx of social bots has poured into these platforms. Their malicious activities, such as manipulating public opinion and invading users' privacy, seriously disrupt the normal operation of social networks. Therefore, detecting social bots becomes one of the important tasks to ensure the security of social networks. Existing studies on the detection of social bots focus on utilizing graph neural networks, which can effectively capture community features, i.e., user attribute features and network topology. However, unlike genuine users in social networks, social bots typically exhibit behavior patterns of high-frequency activity at specific times. In this paper, we first study the differences on behavior patterns among different categories of users, and then propose a novel graph neural network model that integrates dual-dimensional features, including user behavior features and community features. The proposed model introduces user behavior features, which enhances its adaptability to dynamic scenarios. We evaluate the performance of the proposed model on real world datasets. The experimental results show that the proposed model clearly outperform the baseline methods. Especially, the dual-dimensional fusion approach can significantly improve detection accuracy and F1 scores.

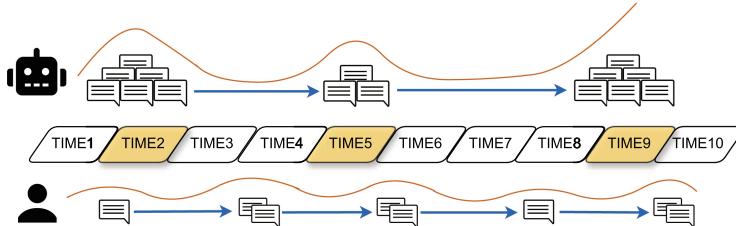
**Keywords:** Social bot detection · Graph neural networks · Social networks · Representation learning

## 1 Introduction

Social networks play significant roles in the production and dissemination of information. The information carried by online social networks (OSNs), such as Twitter, Weibo, and Tiktok, brings positive changes to society in terms of interpersonal interactions, cultural patterns, economic development, and political communication.

However, such open qualities of social networks make them vulnerable to premeditated malicious activities, such as dissemination of misinformation, malicious traffic attacks, manipulation of public opinion, invasion of personal privacy, and even participation in cybercrime [16, 18]. These activities are usually executed by social bots controlled by automated algorithms. In recent years, there

have been many incidents in which social bots have been used to influence political discourse, such as the US election [9] and the Russia-Ukraine conflict [30], which had a significant impact on international events. In addition, the social bots also have spread fake news through channels such as email, e.g., conspiracy theories about COVID-19 [25].



**Fig. 1.** An illustration of the differences in behavior patterns of users, where a time period is uniformly divided into 10 time slots, and a genuine user generates and shares content uniformly and at a stable frequency across the 10 time slots, while a social bot generates and shares a large number of messages in time2, time5, and time9.

The disorderly dissemination of such misinformation increases the cognitive bias, social anxiety, and the risk of personal privacy leakage. Therefore, a large number of studies have been proposed to detect social bots in social networks.

Early approaches for social bot detection focused on analyzing and designing user attributes, which included extracting affective expressions from individual accounts [8] and discerning potential bots through comparative analysis of user metadata [26]. Such feature engineering-based approaches are difficult to capture key information as social bots' behaviors become increasingly complex. Further, as deep learning research evolves, new methods continue to emerge. These methods include employing Long Short-Term Memory Networks (LSTM) to handle intricate mixed information in user metadata for classification, introducing unsupervised learning techniques [4, 23], and harnessing generative adversarial networks [17, 28]. Notably, graph neural networks effectively integrate node features with graph structure [2, 20]. Some researches [12, 15] construct social network graphs based on user relationships, leveraging semantic and attribute data to enhance bot detection capabilities. However, the above classes of methods fail to consider one of the important properties of social bots: the social activity of bots is characterized by irregularity and instability over time.

Figure 1 illustrates the respective behavior patterns of the genuine users and the social bots during 10 evenly separated time slots. One can notice that the behavior of a genuine user in the social networks is typically low-frequency and behaves uniformly over time, whereas a social bot exhibits extremely high activity in certain time slots (e.g., time2, time5, and time9 in Fig. 1), which is characterized by irregularity and non-uniformity over time. This means that by extracting the behavior features of social bots in the time dimension, we can obtain more effective information for identification.

Based on the aforementioned concerns, in this paper, we propose a Dual-dimensionality Fusion Graph Neural Network model (DDFGNN) to detect bots in social networks, which not only considers the community features (i.e., attribute information and network structure) of the users, but also can capture the differences of user’s behaviors in the time dimension. The main contributions of this paper are summarized as follows:

- We study the different behavior patterns between genuine users and social bots. The analysis results on a large-scale dataset show that compared with genuine users, the activities of bots in social networks show greater fluctuations over time. Therefore, we incorporate behavior patterns into the description of user features, which enables our detection model to detect bots from a comprehensive dimension.
- We propose a novel neural network model for social bot detection. It captures user behavior features using transformer encoders and fuses them with the corresponding community features captured by graph neural networks to improve bot detection through high-quality fused features.
- We conduct extensive experiments on two representative datasets, and the results consistently show that our model outperform all baseline methods. Furthermore, the effectiveness of our dual-dimensional fusion approach is confirmed through ablation studies.

## 2 Related Work

### 2.1 Robot Detection Based on Deep Learning

Early researches predominantly relied on manually-designed feature extraction coupled with machine learning to identify malicious social bots. To delve deeper into feature disparities and enhance generalization performance, researchers explored deep neural network methodologies for detecting malicious social bots. Kudugunta et al. [21] extracted text features from tweets and contextual features from accounts, utilizing LSTM architecture for bot detection. Cai et al. [3] employed the Behavior-enhanced Deep (BeDM) model, integrating text and temporal information into a unified sequence vector. Hayawi et al. [19] used LSTM to learn user profile metadata and description field features and used a mixture of features to detect bots. In real social networks, a vast amount of unlabelled and uncategorized complex data exists. Unsupervised learning methods, which do not rely on labelled data, cluster accounts based on common features among account groups. Inspired by biological DNA analysis techniques, Cresci et al. [7] transformed user behaviors into digital DNA sequences. Feng et al. [14] leveraged semantic, attribute, and neighborhood information of specific users to construct the unsupervised representation learning framework SATAR, enhancing the generalization of detection models.

Although the models using deep learning can improve the accuracy and efficiency of detection, most methods only focus on the user node dimension, overlooking the relationships between users and the structural information of social

graphs. Therefore, it is a challenging task to develop effective methods to capture graph structure information and user attributes to detect bots more accurately.

## 2.2 Robot Detection Based on Graph

The graphical structure implies a number of features of users. Early graph-based detection models primarily leveraged the analysis of existing social network structures and established user relationships for classification. Feng et al. [11] constructed an undirected graph based on follower and following relationships, calculating Jaccard coefficients to establish matrices for target users and their associated users, thus facilitating bot detection based on matrix similarity. Recently, the advent of graph neural networks has facilitated a more effective capture and utilization of the topological structure of social networks and the local features of user nodes. Feng et al. [15] introduced a framework named BotRGCN, constructing relationship-based heterogeneous graphs to discern social bots. Yang et al. [29] proposed a novel spatial-temporal enhanced self-supervised GNN architecture, thereby enhancing model detection performance by adaptively determining multi-hop neighborhoods and GNN layers. Chen et al. [5] enhanced model generalization by mining hard-negative and hard-positive samples from community structures and using graph contrast learning and graph convolutional networks. However, the large amount of complex data makes these methods focus only on user features and network structure, while ignoring the subtle differences between genuine users and bots in the time dimension.

## 3 Preparation

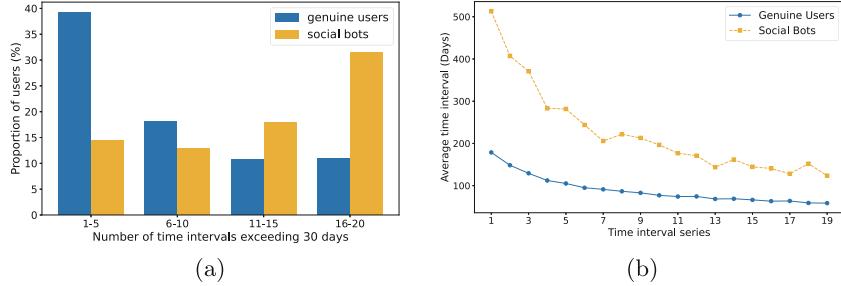
In this section, we engage in an in-depth investigation of social bots on a large-scale social network, based on which we provide a formal description of the research problem.

### 3.1 Data Analysis

We consider the dataset Twibot-22 [13], a large and graph-based Twitter bot detection benchmark, to study the social bot problem. The dataset contains one million nodes and over 100 million edges. For evaluation purpose, we crawl a subnetwork with a hundred thousand nodes.

On various social platforms, users interact with each other through social activities such as posting and commenting. We study the different behavior patterns of genuine users and social bots during the sampled time period. Intuitively, we want to find out the following question: What is the difference in the time interval between each social activity for different categories of users, i.e., genuine users and social bots?

To investigate the frequency of social behaviors of different categories of users, for each user in the dataset, we extract the 20 social activities closest to the current time, and construct a time difference sequence for each user. We



**Fig. 2.** User behavior on the social network, where the time difference between 20 sampled activities generates 19 time intervals. (a) represents the distribution of time intervals between different types of user activities. (b) Represents the average time interval statistics for each of the sampled user activities.

count the time intervals between each social activity to obtain the number of time intervals exceeding 30 days for different types of users, and analyze the distribution of the average time intervals of users' social activities.

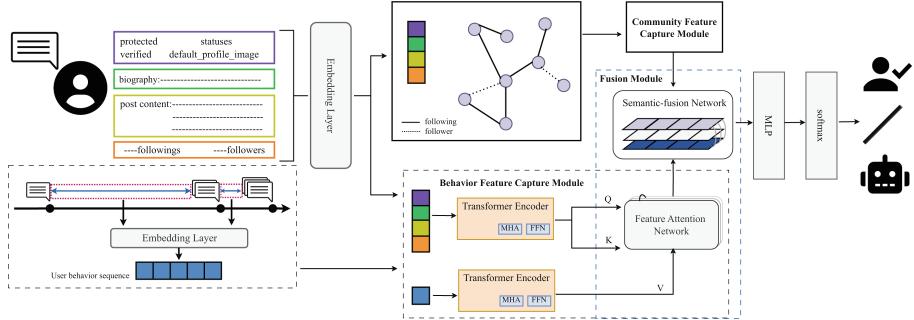
Figure 2(a) shows that when the corresponding activity frequency is low, genuine users account for 39.26% of the sample, while bots account for 14.34% of the sample. As the activity frequency increases, the proportion of genuine users gradually decreases, while the proportion of bots increases. In addition, we can see from Fig. 2(b) that the curve of time difference series of the social bots fluctuates greatly, while the curve of time difference series of the genuine users is relatively smooth. This implies that, unlike genuine users, social bots usually exhibit high activity during certain periods aimed at achieving the manipulator's goals. Under such behavior patterns, the activity intervals of bots may show significant fluctuations, e.g., a sudden burst of high-frequency activity after a long period of silence. Such differences in behavior patterns exhibited by different types of users suggests the effectiveness of using behavior features for social bot detection.

### 3.2 Problem Formulation

In real scenarios, there are large differences between the behavior patterns of social bots and genuine users, so we classify the data in social networks into static community features and behavior features with temporal information.

**Community Feature.** Given a network  $G$ , let  $N_u$  denote the set of users in the network  $G$ . Each user contains a static attribute set  $C_u = \{C_u^{num}, C_u^{cat}, C_u^{des}, C_u^{twi}\}$  representing numerical attributes, categorical attributes, account descriptions and tweets respectively. And we denote  $R_u = \{R_u^{follower}, R_u^{following}\}$  as a set of relationships between follower and following of user  $u$ .

**Behavior Feature.** Additionally, to better characterize user behavior, a time difference mechanism is employed, wherein differential operations on time data extract interval information. We define a set of time differences  $T_u = \{t_{u,1}, \dots, t_{u,i}, \dots, t_{u,j-1}, t_{u,j}\}$  representing the difference in posting times between two tweets. The task of detecting social bots involves utilizing user  $C_u$  and relationship  $R_u$  information to generate community vectors, and then fusing  $C_u$  with time difference information  $T_u$  to generate behavior vectors that jointly identify bots in users.



**Fig. 3.** The framework of DDFGNN with a community feature capture module and a behavior feature capture module.

## 4 Methods

In this section, we primarily elucidate the proposed dual-dimensional fusion graph neural network (DDFGNN) model. Figure 3 illustrates the overall architecture of the model. Specifically, DDFGNN consists of two modules: the behavior feature capture module and the community feature capture module. The community feature capture module captures attribute information from user accounts and structural data from the network by constructing a social graph. The behavior feature capture module reassesses users by using their temporal features as weights through a feature attention network. Eventually, community features and behavior features are fused together through semantic fusion network to obtain high-quality user features for social bot detection.

### 4.1 Community Feature Capture Module

Initially, we employ RGCN to obtain the attribute information and network structure of our users and integrate them into the model as user community features. The initial overall attribute information of users can be represented as:

$$r_i = [r_{d,i}; r_{t,i}; r_{p,i}^{num}; r_{p,i}^{cat}] \quad (1)$$

where  $r_{d,i}$  and  $r_{t,i}$  represent the user's description and tweet features, respectively, pre-trained using RoBERTa [22].  $r_{p,i}^{num}$  and  $r_{p,i}^{cat}$  represent the sum of the user's numerical and categorical features, encoded using z-score normalization and one-hot encoding, respectively. Specifically, we transform the initial attribute features to derive hidden vectors:

$$y_i^{(0)} = \partial(W_1 \cdot r_i + b_1) \quad (2)$$

where  $W_1$  and  $b_1$  are learnable parameter matrices.  $\partial$  represents nonlinearity, and in this paper, we use leaky-ReLU as  $\partial$ . Next, we utilize the RGCN layer to combine the attribute features with the graph relational structure:

$$y_i^{(l+1)} = \theta_{self} \cdot y_i^{(l)} + \sum_{r \in R} \sum_{j \in N_r(i)} \frac{1}{|N_r(i)|} \theta_r y_j^{(l)} \quad (3)$$

where  $\theta$  is the projection matrix. Finally, we use a multi-layer perceptron to obtain the community features:

$$x_{i\text{-Community}} = \partial(W_2 y_i^{(L)} + b_2) \quad (4)$$

where  $W_2$  and  $b_2$  are learnable parameter matrices.

## 4.2 Behavior Feature Capture Module

To achieve a more comprehensive acquisition of account behavior information and overcome the inherent low frequency and unevenness of behavior information, inspired by GPTCN [24], we design a scheme called DETN for social bots. Specifically, we construct two transformer encoders to obtain user attribute information and time difference information respectively. In the feature attention network, we use the encoded time difference information as weights to evaluate the user attribute information and fuse it to obtain the final behaviour features.

Firstly, we map the initial time difference information into a hidden space:

$$t_i^{(0)} = \partial(W_3 \cdot t_i + b_3) \quad (5)$$

where  $W_3$  and  $b_3$  are learnable parameters. Then, we utilize dual transformer encoder to fuse and compress information. The head self-attention in the model is explicitly defined as:

$$MHSA(t_i^{(0)}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \quad (6)$$

$$\text{head}_i = \text{Attention}(t_i^{(0)} W_{t\_i}^Q, t_i^{(0)} W_{t\_i}^K, t_i^{(0)} W_{t\_i}^V) \quad (7)$$

$$MHSA(y_i^{(0)}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O \quad (8)$$

$$\text{head}_i = \text{Attention}(y_i^{(0)} W_{y\_i}^Q, y_i^{(0)} W_{y\_i}^K, y_i^{(0)} W_{y\_i}^V) \quad (9)$$

where the projection matrices  $W_{t\_i}^Q$ ,  $W_{t\_i}^K$ ,  $W_{t\_i}^V$  and  $W_{y\_i}^Q$ ,  $W_{y\_i}^K$ ,  $W_{y\_i}^V$  are weight matrices maintained in the MHSA, representing the trainable parameters in

each attention head of MHSA. The attention function is implemented through dot product operation:

$$\text{Attention}(K, Q, V) = \text{soft max}\left(\frac{QK^T}{\sqrt{d/h}}\right)V \quad (10)$$

$\sqrt{d/h}$  serves as a scale factor to prevent numerical products from becoming too large, while also leveraging a feed-forward neural network to provide non-linear characteristics, as follows:

$$\text{FFN}(t_i^{(0)}) = \partial(t_i^{(0)}W_4 + b_4)W_5 + b_5 \quad (11)$$

$$\text{FFN}(y_i^{(0)}) = \partial(y_i^{(0)}W_6 + b_6)W_7 + b_7 \quad (12)$$

where  $W_4, W_5, W_6, W_7, b_4, b_5, b_6$  and  $b_7$  are all trainable parameters. To better focus on key features in the sequence, we concatenate multiple layers of encoders, with the final output being:

$$\tilde{t}_i^{(n)} = \text{LayerNorm}(\text{MHSA}(t_i^{(n)}) + t_i^{(n)}) \quad (13)$$

$$\hat{t}_i^{(n+1)} = \text{LayerNorm}(\text{FFN}(\tilde{t}_i^{(n)}) + \tilde{t}_i^{(n)}) \quad (14)$$

$$\tilde{y}_i^{(n)} = \text{LayerNorm}(\text{MHSA}(y_i^{(n)}) + y_i^{(n)}) \quad (15)$$

$$\hat{y}_i^{(n+1)} = \text{LayerNorm}(\text{FFN}(\tilde{y}_i^{(n)}) + \tilde{y}_i^{(n)}) \quad (16)$$

where  $\hat{t}_i^{(n+1)}$  and  $\hat{y}_i^{(n+1)}$  are the final output of the  $n$ -th layer time difference encoder and attribute encoder.

**Feature Attention Network.** We employ the cross-attention mechanism to integrate the time dimension as a weighting factor into the attribute feature information. Its precise mathematical definition is as follows:

$$\text{MHA}(\hat{t}_i^{(n)}, \hat{y}_i^{(n)}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (17)$$

$$\text{head}_i = \text{Attention}(\hat{t}_i^{(n)}W_i^Q, \hat{y}_i^{(n)}W_i^K, \hat{y}_i^{(n)}W_i^V) \quad (18)$$

Based on the above content, we define the encoding of behavior features as follows:

$$\tilde{x}_{i\_Behavior}^{(n)} = \text{LayerNorm}(\text{MHA}(\hat{t}_i^{(n)}, \hat{y}_i^{(n)})) \quad (19)$$

$$\hat{x}_{i\_Behavior}^{(n)} = \text{LayerNorm}(\text{FFN}(\tilde{x}_{i\_Behavior}^{(n)})) \quad (20)$$

where  $\hat{x}_{i\_Behavior}^{(n)}$  is the output of the  $n$ -th layer of cross-attention fusion. We use softmax to obtain the behavior feature vector we ultimately construct:

$$x_{i\_Behavior} = \text{soft max}(\hat{x}_{i\_Behavior}^{(n)}M^T) \quad (21)$$

where  $M$  is the weight matrix maintained by the embedding layer.

**Semantic Fusion Network.** After analyzing the community features and behavior features separately, we use semantic fusion network to aggregate multidimensional features in order to achieve information sharing between different semantics. The final user features can be represented as:

$$a_i = \partial(W_8 \cdot [x_{i\_Community}; x_{i\_Behavior}] + b_8) \quad (22)$$

where  $W_8$  and  $b_8$  are learnable parameter matrices.

### 4.3 Loss Function

Based on our fused feature representation, we apply softmax for social bot detection tasks:

$$\hat{y}_i = \text{softmax}(W_o \cdot a_i + b_o) \quad (23)$$

where  $W_o$  and  $b_o$  are learnable parameters. Additionally, we introduce weighted cross-entropy loss to address the issue of class imbalance:

$$L = - \sum_{i \in Y} [\omega_+ y_i \log(\hat{y}_i) + \omega_- (1 - y_i) \log(1 - \hat{y}_i)] + \lambda \sum_{\omega \in \theta} \omega^2 \quad (24)$$

where  $Y$  represents labeled users,  $y_i$  is the true label,  $\theta$  is the learnable parameter in the model architecture, and  $\omega_+$  and  $\omega_-$  are the weights added to positive and negative classes, respectively.

## 5 Experiments

In this section, we present the experimental results obtained by applying our model to the famous dataset Cresci-2015 [6] and the recently proposed large-scale dataset Twibot-22 [13].

### 5.1 Experiment Setup

**Datasets.** Our model incorporates multidimensional features of users, which necessitates user behavior data, attribute data, and a comprehensive social graph structure for subsequent training. The Cresci-2015 [6] dataset consists of two types of users and three types of bots, containing crucial data such as user relationships and posting behavior. Twibot-22 [13] is the largest and most comprehensive recently proposed bot detection dataset, featuring a diverse array of users and information better aligned with real-world social networks. In this paper, we use Cresci-2015 and Twibot-22 to validate the effectiveness of our model. We follow the original baseline delineation in order to make direct comparisons with previous studies.

**Baselines.** We conduct comparisons against the following baselines.

- **Alhosseini et al.** [1] employed graph convolutional networks to learn node and neighbor features for bot detection.
- **Hayawi et al.** [19] leveraged user profile data, integrating LSTM units with dense layers for classification.
- **Kudugunta et al.** [21] merged user metadata with tweet semantic information for tweet-level detection.
- **Feng et al.** [12] considered the heterogeneity of relationships and influences in heterogeneous networks for node representation learning.
- **Efthimion et al.** [10] utilized message variance along with other user features in conjunction with SVM.
- **Wei et al.** [27] did not employ prior knowledge, instead integrating BiLSTM with word embedding techniques to capture features from tweets.
- **BotRGCN** [15] utilized multimodal user semantic and attribute information, aggregating it via RGCN.
- **Chen et al.** [5] mined the community for positive and negative samples and used graph contrast learning and GCN to detect social bots.

**Table 1.** Performance comparison of our method with selected baselines in bot detection

Method	Twibot-22				Cresci-2015			
	ACC	Precision	Recall	F1	ACC	Precision	Recall	F1
BotRGCN [15]	0.7966	0.7481	0.4680	0.5750	0.9652	0.9551	0.9917	0.9730
Alhosseini [1]	0.4772	0.2999	0.5675	0.3810	0.8957	0.8769	0.9716	0.9217
Kudugunta [21]	0.6587	0.4431	<b>0.6198</b>	0.5167	0.7533	<b>1.0000</b>	0.6095	0.7574
Feng [12]	0.7647	0.7503	0.3010	0.4294	0.9715	0.9638	<b>0.9923</b>	0.9778
Hayawi [19]	0.7650	<b>0.8000</b>	0.1499	0.2474	0.8427	0.9296	0.7931	0.8556
Efthimion [10]	0.7408	0.7778	0.1676	0.2785	0.9252	0.9382	0.9438	0.9410
Wei [27]	0.7628	0.6732	0.1965	0.3041	0.9610	0.9170	0.7530	0.8270
Chen [5]	0.7507	0.6546	0.3826	0.4827	0.9588	0.9539	0.9817	0.9674
<b>DDFGNN (Ours)</b>	<b>0.8605</b>	0.7190	0.5468	<b>0.6166</b>	<b>0.9827</b>	0.9846	0.9884	<b>0.9922</b>

## 5.2 Performance in Bot Detection

We select representative and typical methods in recent years to conduct experiments, and the results are presented in Table 1, which clearly demonstrates the superiority of our method. In terms of detection task performance on the TwiBot-22 dataset, BotRGCN showed relatively good performance, whereas Alhosseini’s

method underperformed compared to others. Eftimiont's proposed method integrated multiple data points to analyze user behavior, but the recall and F1 scores are low due to the lack of information about the structure of the network as well as the sensitivity of the traditional machine learning algorithms to the noise and imbalance in large-scale datasets. The methods of Hayawi et al. and Kudugunta et al. utilized LSTM units for detection and perform well in terms of precision and recall. This disparity might stem from different methods of extracting user description and tweet semantic features between the two approaches. We attribute the poor recall and F1 scores performance in Hayawi's method to the potential information loss in LSTM when processing long sequence data. Although Wei et al. employed BiLSTM with lower accuracy, they effectively addressed the lower recall and F1 issues prevalent in previous methods. We believe that bidirectional LSTM may effectively reduce the degree of information loss. Interestingly, while both BotRGCN and Alhosseini's methods utilized GCN, their performances significantly differ. We contend that besides designing comprehensive feature structures, effective weight aggregation among different relationships is crucial for model performance. Although Feng's method accounts for the heterogeneity of relationships in social networks, employing a globally focused graph transformer risks significant information loss in large-data scenarios. Chen's approach enhanced community feature extraction using graph comparison learning, but the absence of behaviour features resulted in low recall and F1 scores.

In terms of the detection task performance on the Cresci-2015 dataset, methods based on user node information generally perform poorly compare to graph-based detection methods such as BotRGCN. For example, Hayawi, Kudugunta, and Wei's methods have relatively low recall with high precision. We believe that besides the information loss in the LSTM unit used for sequence information capture, the lack of user relationship data also contributes to the poor performance of these methods.

Therefore, we design DDFGNN to integrate user behavior information with substantial community information, i.e., user attribute features and network topology, and appropriately account for the influence of relationship weights to achieve comprehensive detection of social bots in multiple dimensions. The method on the classic Cresci-2015 dataset, it is 1.12% more accurate than the best Feng's method, and the F1 score is 1.44% higher. It outperforms the most representative BotRGCN on the Twibot-22 dataset by 6.39%, 7.88% and 4.16% in terms of accuracy, recall and F1 score, respectively. The experiments reveal a force majeure factor in the Twibot-22 dataset in the form of an imbalance between the number of user and bot samples. However, our method is the best in both accuracy and F1 score, so it can also reflect the effectiveness of our model performance.

### 5.3 Ablation Study

To demonstrate the effectiveness of our fusion of dual-dimensional features, we conduct ablation experiments on the Twibot-22 dataset, as shown in Table 2.

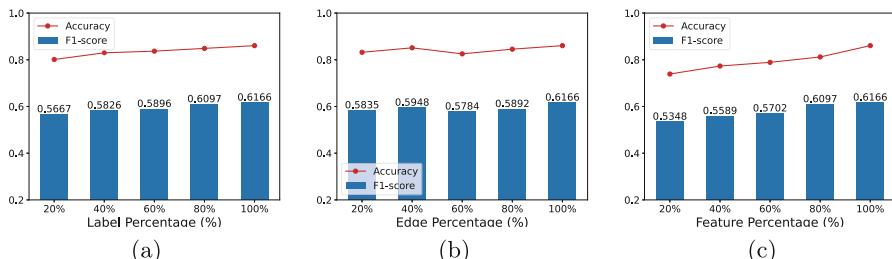
**Table 2.** Ablation experiments between different modules of DDFGNN

Module	Accuracy	Precision	Recall	F1
BFCM	0.8168	0.6962	0.3134	0.4276
CFCM	0.7966	0.7481	0.4680	0.5750
DDFGNN	<b>0.8605</b>	0.7190	<b>0.5468</b>	<b>0.6166</b>

We employ a single behavior feature capture module (BFCM) and a single community feature capture module (CFCM) for bot detection. It is evident that BFCM excels in accuracy metrics by leveraging information from various user behavior patterns, demonstrating the effectiveness of behavior features in enhancing model performance. Conversely, CFCM display superior F1 scores, suggesting that leveraging community features effectively ensures the stability of the model’s overall performance and highlights differences in focus among the various modules. Overall, the performance of the complete model surpasses that of the two independent modules, achieving substantial improvements in accuracy, recall, and F1 scores, thereby confirming the feasibility and effectiveness of the dual-dimensional fusion model we adopted.

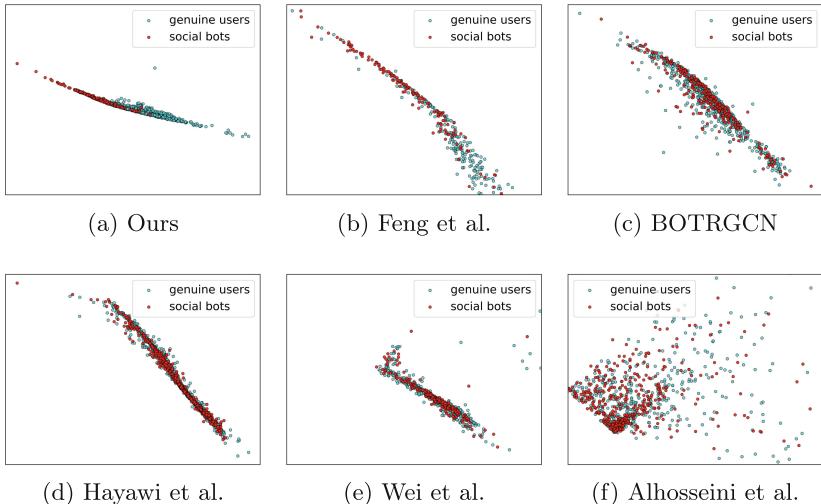
#### 5.4 Data Efficiency Study

Existing bot detection models rely on a large amount of labeled data. To validate the performance of our model in limited data scenarios, we mask different proportions of the training set data labels, edges in the social graph and user features in Twibot-22, respectively. As shown in Fig. 4, our model still outperforms other models with only 40% of the training data. For the edges of the social graph, our model is robust. And the reduction of user features significantly affects the performance of our model, which indicates that in addition to graph structure features our model also relies on user behavior features and other features, such as attribute features contained in community features.

**Fig. 4.** Performance of our model with different scaled data labels, edges, and user features.

## 5.5 User Representation Learning Study

We analyze the user representations learned by our model and some representative models. To better demonstrate the effect of our model representation learning, we reduce the user representations to two dimensional vectors and visualize them. As shown in Fig. 5, we use two colors to distinguish the labels of genuine users and social bots. Compared to the other models we selected, DDFGNN is more focused and effective in separating the extraction of genuine user and bot representations. This suggests that the representations learned by our model are more beneficial for the social bot detection task.



**Fig. 5.** Visualization of user representations learned by our model and different models on the Twibot-22 dataset.

## 6 Conclusions

The detection of social bots has gained significant attention. Our comprehensive analysis of the dataset revealed a notable trend: bots tend to exhibit heightened activity following prolonged periods of silence. Capitalizing on this observation, we introduce DDFGNN, a novel model that integrates user behavior and community features to achieve a more comprehensive analysis and more accurate while detection of social bots, unlike previous approaches that only consider single-dimensional features. We have conducted extensive experiments on two real datasets, and the results show that our approach has significant advantages. Our model significantly improves both accuracy and F1 scores compared to the SOTA baseline. Moving forward, our focus is on refining and enhancing the model’s generalization across various social platforms and optimizing the model’s computational overhead to enhance its potential for real-world applications.

## References

1. Ali Alhosseini, S., Bin Tareaf, R., Najafi, P., Meinel, C.: Detect me if you can: spam bot detection using inductive representation learning. In: Companion Proceedings of the 2019 World Wide Web Conference, pp. 148–153 (2019)
2. Benslimane, S., Azé, J., Bringay, S., Servajean, M., Mollevi, C.: A text and GNN based controversy detection method on social media. *World Wide Web* **26**(2), 799–825 (2023)
3. Cai, C., Li, L., Zengi, D.: Behavior enhanced deep bot detection in social media. In: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 128–130. IEEE (2017)
4. Chavoshi, N., Hamooni, H., Mueen, A.: Debot: Twitter bot detection via warped correlation. In: ICDM, vol. 18, pp. 28–65 (2016)
5. Chen, S., Feng, S., Liang, S., Zong, C.C., Li, J., Li, P.: CACL: community-aware heterogeneous graph contrastive learning for social media bot detection. arXiv preprint [arXiv:2405.10558](https://arxiv.org/abs/2405.10558) (2024)
6. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: Fame for sale: efficient detection of fake twitter followers. *Decis. Support Syst.* **80**, 56–71 (2015). <https://doi.org/10.1016/j.dss.2015.09.003>
7. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: DNA-inspired online behavioral modeling and its application to spambot detection. *IEEE Intell. Syst.* **31**(5), 58–64 (2016)
8. Davis, C.A., Varol, O., Ferrara, E., Flammini, A., Menczer, F.: Botornot: a system to evaluate social bots. In: Proceedings of the 25th International Conference Companion on World Wide Web, pp. 273–274 (2016)
9. Deb, A., Luceri, L., Badaway, A., Ferrara, E.: Perils and challenges of social media and election manipulation analysis: the 2018 us midterms. In: Companion Proceedings of the 2019 World Wide Web Conference, pp. 237–247 (2019)
10. Efthimion, P.G., Payne, S., Proferes, N.: Supervised machine learning bot detection techniques to identify social twitter bots. *SMU Data Sci. Rev.* **1**(2), 5 (2018)
11. Feng, B., Li, Q., Pan, X., Zhang, J., Guo, D.: Groupfound: an effective approach to detect suspicious accounts in online social networks. *Int. J. Distrib. Sens. Netw.* **13**(7), 1550147717722499 (2017)
12. Feng, S., Tan, Z., Li, R., Luo, M.: Heterogeneity-aware twitter bot detection with relational graph transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 3977–3985 (2022)
13. Feng, S., et al.: Twibot-22: towards graph-based twitter bot detection. *Adv. Neural. Inf. Process. Syst.* **35**, 35254–35269 (2022)
14. Feng, S., Wan, H., Wang, N., Li, J., Luo, M.: Satar: a self-supervised approach to twitter account representation learning and its application in bot detection. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 3808–3817 (2021)
15. Feng, S., Wan, H., Wang, N., Luo, M.: Botrgcn: Twitter bot detection with relational graph convolutional networks. In: Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 236–239 (2021)
16. Gayo-Avello, D.: Social media won't free us. *IEEE Internet Comput.* **21**(4), 98–101 (2017)
17. Grimme, C., Assenmacher, D., Adam, L.: Changing perspectives: is it sufficient to detect social bots? In: Meiselwitz, G. (ed.) SCSM 2018. LNCS, vol. 10913, pp. 445–461. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-91521-0\\_32](https://doi.org/10.1007/978-3-319-91521-0_32)

18. Hagen, L., Neely, S., Keller, T.E., Scharf, R., Vasquez, F.E.: Rise of the machines? Examining the influence of social bots on a political discussion network. *Soc. Sci. Comput. Rev.* **40**(2), 264–287 (2022)
19. Hayawi, K., Mathew, S., Venugopal, N., Masud, M.M., Ho, P.H.: Deeprobot: a hybrid deep neural network model for social bot detection based on user profile data. *Soc. Netw. Anal. Min.* **12**(1), 43 (2022)
20. Knoke, D., Yang, S.: Social Network Analysis. SAGE Publications (2019)
21. Kudugunta, S., Ferrara, E.: Deep neural networks for bot detection. *Inf. Sci.* **467**, 312–322 (2018)
22. Liu, Y., et al.: Roberta: a robustly optimized bert pretraining approach. arXiv preprint [arXiv:1907.11692](https://arxiv.org/abs/1907.11692) (2019)
23. Mazza, M., Cresci, S., Avvenuti, M., Quattrociocchi, W., Tesconi, M.: Rtburst: exploiting temporal patterns for botnet detection on twitter. In: Proceedings of the 10th ACM Conference on Web Science, pp. 183–192 (2019)
24. Sun, Y., Zeng, F., Xiao, J., Deng, Y., Ding, Y., Li, Y.: GPTCN: gated parallel transformer convolutional networks for downstream-task user representation learning on app usage. In: ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5175–5179. IEEE (2024)
25. Van Mulukom, V., et al.: Antecedents and consequences of covid-19 conspiracy beliefs: a systematic review. *Soc. Sci. Med.* **301**, 114912 (2022)
26. Wang, G., et al.: Social turing tests: crowdsourcing sybil detection. arXiv preprint [arXiv:1205.3856](https://arxiv.org/abs/1205.3856) (2012)
27. Wei, F., Nguyen, U.T.: Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In: 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), pp. 101–109. IEEE (2019)
28. Wu, B., Liu, L., Yang, Y., Zheng, K., Wang, X.: Using improved conditional generative adversarial networks to detect social bots on twitter. *IEEE Access* **8**, 36664–36680 (2020)
29. Yang, Y., et al.: Rosgas: adaptive social bot detection with reinforced self-supervised GNN architecture search. *ACM Trans. Web* **17**(3), 1–31 (2023)
30. Zhao, B., Ren, W., Zhu, Y., Zhang, H.: Manufacturing conflict or advocating peace? A study of social bots agenda building in the twitter discussion of the Russia-Ukraine war. *J. Inf. Technol. Polit.* **21**(2), 176–194 (2024)



# A Motif-Based Graph Convolution Network for Stock Trend Prediction

Lei Zhou<sup>1</sup> , Yuqi Zhang<sup>1</sup> , Nancy Wang<sup>1</sup> , Jian Yu<sup>1</sup> (✉),  
Guiling Wang<sup>2</sup> , and Xin Zheng<sup>2</sup>

<sup>1</sup> Department of Computer Science, Auckland University of Technology,  
Auckland, New Zealand  
[jian.yu@aut.ac.nz](mailto:jian.yu@aut.ac.nz)

<sup>2</sup> Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data,  
School of Information Science and Technology, North China University of  
Technology, Beijing, China

**Abstract.** The prediction of stock market trends remains a challenging task, garnering significant attention from economists and computer scientists. Recent studies have demonstrated that integrating stock prices with news data improves prediction accuracy. However, current approaches often overlook the complex inter-stock relationships embedded in news data. Deep learning, particularly Graph Convolutional Networks (GCNs), has shown promising results in stock trend prediction by leveraging a message-passing framework that allows nodes to aggregate information from their neighbors. In this paper, we propose a novel method, the Motif-based Graph Convolutional Network for Stock Prediction (MGSP), which addresses the over-smoothing problem by incorporating network motifs into the layer propagation. Our approach constructs a motif graph by correlating stocks with stock news, and employs scaled dot product attention from the transformer architecture to encode stock price and news features. The motif-based GCN is then applied to jointly optimize the embeddings of stock news and time series data, using a transformer encoder to estimate the probability of future price movements. Extensive experiments on U.S. stock data show that our method outperforms several state-of-the-art techniques.

**Keywords:** Graph neural networks · Stock prediction · Network motifs · Transformer

## 1 Introduction

The global stock markets are highly complex and dynamic systems, and efforts have been made to tackle the challenge of stock market prediction. Recently, deep learning technologies have seen extensive application and research in the field of stock price trend prediction [12]. Scholars are exploring the integration of a broader spectrum of data as inputs into deep learning models to enhance predictive accuracy. Various key factors, such as a variety of technical indicators,

market factors, financial reports, online news articles, and social media content, affect the movement of the stock market.

Time series are often treated as independent and identically distributed, an assumption that does not align with the actual dynamics of financial markets [1]. For instance, stocks within the same sector typically exhibit stronger correlations compared to those across diverse sectors [3]. In light of this, recent research has shifted towards utilizing graphs to encapsulate the intricate network of relationships among financial entities [4]. This shift includes the application of graph-based learning techniques for forecasting stock price movements [14]. Such methodologies have proven their merit by incorporating the correlation of stocks into their predictive frameworks [5]. Consequently, employing graph-based strategies offers the potential to derive more nuanced and meaningful representations of the input data, thereby enhancing the accuracy of predictions.

Recent advancements in graph convolutional network (GCN) applications for stock prediction, as shown in studies [15, 22], have highlighted the potential of GCNs in capturing complex interrelationships within financial data. Our study addresses this challenge by integrating network motifs-recurring, significant subgraph patterns that are more prevalent in complex networks than in randomized ones. Our approach leverages motifs in Graph Convolutional Networks to identify latent relationships between stock prices and related news. These recurring subgraph patterns in bipartite graphs, where one set of nodes represents stocks and the other news data, offer a novel way to analyze the influence of news on stock prices. By exploiting these motifs, our method captures complex interactions between stocks and their informational context, improving the accuracy of stock market predictions.

In this work, We introduce a novel approach for creating motif adjacency matrices corresponding to each identified motif. Through these matrices, we develop a motif-oriented graph attention mechanism, designed to integrate complex, higher-order data encapsulated within the motifs. Subsequently, we present the Motif-based Graph Convolutional Network for Stock Prediction (MGSP), a new model conceptualized for forecasting stock market trends. Our proposed model integrates a transformer-based encoder-decoder architecture with a graph convolutional network to efficiently capture and utilize both sequential and structural information for stock prediction.

Our main contributions are summarized as follows:

1. We propose a Motif-graph-based approach for constructing a bipartite graph based on stocks and their associated news. This bipartite graph is utilized to extract potential relationships between different stocks.
2. We propose a framework called MGSP that combines Transformer and motif-based Graph Convolutional Networks (MGCN) for the task of stock trend prediction.
3. We collected the S&P 500 dataset from the US stock market and applied our proposed methodology and model for analysis. In comparison with other deep learning methods that do not utilize Motif graphs, our approach and model demonstrated superior performance, outperforming the alternatives.

This paper is structured in the following manner: Sect. 2 provides an overview of the work related to our research. In Sect. 3, we define motifs and introduce the motif-based graph convolution self-attention approach, along with a detailed presentation of the MGSP model. Section 4 details the experimental setup, comparative results, and in-depth analyses. Finally, Sect. 5 offers concluding remarks and summarizes the paper’s findings.

## 2 Related Work

### 2.1 Traditional Machine Learning Models for Stock Prediction

Machine learning models, including Support Vector Machines (SVM), decision trees, Artificial Neural Networks (ANN), and Random Forests (RF), have demonstrated enhanced performance in capturing the complexities of stock data. These methods have been applied in various configurations to improve the accuracy of stock trend predictions. For instance, Lee et al. [11] study successfully integrated SVM with hybrid feature selection techniques for trend prediction. In [17], they proposed a novel combination of RF, SVM, and ANN for stock prediction tasks. Parray et al. [16] explored the synergistic use of SVM, ANN, and logistic regression to predict next-day stock trends. These advancements highlight the dynamic nature of stock market prediction research and the increasing reliance on machine learning methodologies to navigate its complexities.

### 2.2 Deep Learning Methods for Stock Prediction

Within the current paradigm of predictive analytics for financial markets, there has been a pronounced increase in the utilization of deep learning methodologies. Methods such as Long Short-Term Memory networks (LSTM) [7], Convolutional Neural Networks (CNN) [10], Gated Recurrent Units (GRU) [6], Graph Neural Networks (GNN) [19], and attention-based mechanisms such as Transformer [21] have emerged as frontrunners. Their increasing prominence can be attributed to their efficacy in modeling the intricate, non-linear relationships and spatial-temporal dynamics inherent in stock market data. In [2], they combined wavelet transforms, stacked autoencoders, and LSTM to forecast stock prices. In [20], they introduced a method of converting stock market data into 2-D images, using CNNs for stock trading decisions. The 2-D image size is  $15 \times 15$ , which includes 15 technical indicators over 15 intervals for each day. They transformed stock market data into images, marking the buy and sell points, and labeled the images as buy, sell, or hold. CNNs were then used to determine these points in stock prices. In [13], they presented a model based on a corporate knowledge graph embedding system to extend corporate-related news and combined stock news sentiment with stock market data features using the GRU model to predict stock prices.

### 2.3 Graph Neural Networks for Stock Prediction

Graph-based neural architectures, as delineated in the seminal work by In [19], offer a framework for directly interfacing with data embodied in graph formats. Such architectures deploy mechanisms for aggregating and updating nodal representations by leveraging the interconnected information from adjacent nodes and their corresponding edges [23]. In financial applications, individual stocks are typically conceptualized as nodes within a network, interconnected by edges that signify the existence of inter-stock relationships. The task of forecasting trends across an array of stocks often translates into a node classification challenge, aptly addressed through graph neural network constructs [8]. The intrinsic strength of GNNs lies in their capacity to assimilate inter-stock relationships, enriching the predictive model with relational data nuances. In the context of stock market representation, utilized a graph-structured approach wherein companies are symbolized as nodes, and the interrelations among stocks are articulated through edges [3]. In [9], they present a novel hierarchical attention network designed for stock prediction (HATs), employing relational data to forecast stock market movements. This approach distinctively assimilates information across varied relation types, enhancing the individual representations of companies.

## 3 Materials and Methods

In this section, we will delve into the foundational aspects of our MGSP model, encompassing the data collection methods, the preprocessing techniques applied to the gathered data, and the methodology for constructing a graph that encapsulates the relationships between stocks. We will provide an in-depth exploration of these key components in the following discussion.

### 3.1 Data Representation

For the development and validation of our MGSP model, we sourced daily trading data of the S&P 500 index from Yahoo Finance, spanning the years 2018 to 2019. Our selection criteria for the dataset included trade dates, trade volumes, and closing prices, which served as both the input features and the targets for our predictions. The comprehensive dataset for the S&P 500 encompasses several key attributes: stock code, trade date, opening price, closing price, daily highest price, daily lowest price, and trading volume.

Additionally, we gathered sectorial information for all companies listed in the S&P 500 index, alongside the titles of news articles pertaining to each stock within the same timeframe. This holistic approach enabled us to incorporate both quantitative market data and qualitative news sentiment into our model, enriching the predictive analysis.

Our dataset comprises 354 training samples and 143 testing samples, derived from working days within the observed period. Each sample is characterized by three features: the S&P 500 index price, and a sequence representing the price movements of all listed stocks over the preceding 30 days.

Given the temporal nature of stock market data, our approach frames stock prediction as a sequence classification task. Therefore, each input sample  $x$  is structured as a matrix with dimensions  $T \times d$ , where  $T$  represents the length of the sequence (30 days in our case) and  $d$  signifies the dimensionality of the features for each day. This methodological framework ensures a rigorous analysis of temporal patterns in stock prices, facilitating the generation of accurate and actionable predictions through our proposed model.

Following the collection and initial processing of stock market data, we proceeded to preprocess the related stock news to refine our dataset further. This preprocessing involved filtering news content to ensure relevance to the S&P 500 index constituents. Specifically, we eliminated news articles that did not mention any of the S&P 500 stocks or their respective stock codes. This selection criterion was pivotal in aligning the qualitative news data with the quantitative stock market data, thereby streamlining the integration of both data types into our analysis. The filtered news content is essential for constructing a bipartite graph that maps the intricate relationships between stocks and their news articles.

### 3.2 Construction of Motif Adjacency Matrices from Bipartite Graphs

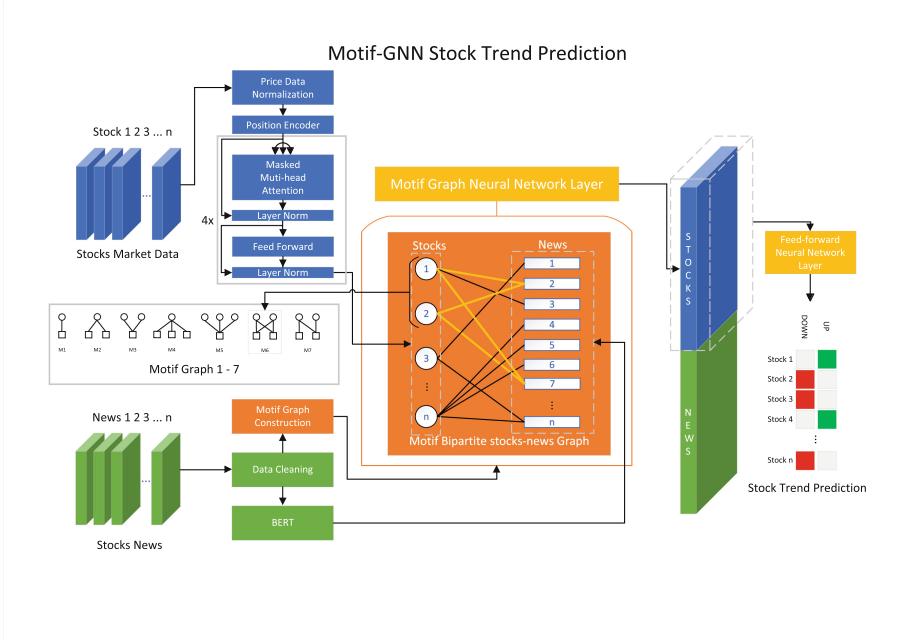
In stock market trend forecasting, we model datasets as bipartite graphs  $G = (U, V, E)$ , where nodes are divided into two disjoint sets  $U$  and  $V$  with edges connecting nodes only between these sets. No edges exist within the same subset. To generate bipartite motif adjacency matrices, we employ the method of Yuqi et al. [24], grouping edges by node type based on the motif structure. For motif  $M_2$ , for instance, if ten news nodes connect to a stock node, these edges form a group, and the adjacency matrix of  $M_2$  connects each pair of news nodes within this group.

To address the lack of direct connections between same-type nodes, we leverage bipartite motifs to capture indirect relationships. This method enhances feature propagation between stock and news nodes. Our Motif-based Graph Convolutional Network (MGCN) extracts and interprets features from stocks within the bipartite graph, enriching the feature set for stock prediction and providing deeper insights into market dynamics through the graph's structural patterns.

### 3.3 MGCN for Stock Prediction Framework

Existing methodologies in the literature [25] integrate stock price features with text features by directly concatenating these data types, followed by encoding through a unified encoder. This prevalent approach, however, does not fully exploit the latent relationships between stock movements and the information contained within textual news data.

We introduce the MGSP model framework, as shown in Fig. 1, which utilizes historical stock prices and stock news to predict future stock movements. To capture the intricate relationships between stocks and news, we construct a



**Fig. 1.** MGCGN Stock Prediction Framework

bipartite graph, deriving motif adjacency matrices that reveal recurrent patterns. We independently process historical price data and stock news, then integrate the encoded features into a unified representation of each stock. This representation is fed into a motif-based Graph Convolutional Network (MGCGN), which efficiently extracts and utilizes relational information from the motif adjacency matrices. This process enables the model to identify complex stock market interactions, improving prediction accuracy.

**Feature Extraction and Representation Learning.** The Transformer model has significantly advanced sequential data processing, particularly in the domain of stock price prediction. At its core, the self-attention mechanism dynamically evaluates the relevance of each sequence position, enhancing the model's capability to process extended sequences and mitigate issues related to gradient vanishing and explosion:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (1)$$

To capture a diverse array of informational cues, the Transformer employs a multi-head attention strategy:

$$\text{MaskedMultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2)$$

$$\text{head}_i = \text{MaskedAttention}(QW_i^Q, KW_i^K, VW_i^V, \text{mask}) \quad (3)$$

where the mask is a lower triangular matrix used to zero out the attention weights for future positions.

The architecture is further refined with the integration of residual connections and layer normalization, enhancing training dynamics:

$$h_1 = h_0 + \text{MaskedMultiHead}(h_0, h_0, h_0) \quad (4)$$

$$h_2 = \text{LayerNorm}(h_1) \quad (5)$$

$$h_3 = h_2 + \text{ReLU}(h_2 W_1 + b_1) W_2 + b_2 \quad (6)$$

$$h_4 = \text{LayerNorm}(h_3) \quad (7)$$

These Transformer-derived outputs ( $h_4$ ) are concatenated with vectors from news content processed through the BERT model, creating an enriched feature set ( $H_{\text{combined}}$ ). This set serves as input for the MGCN, offering a nuanced examination of market interconnectivity:

$$H_{\text{combined}} = \text{Concat}(h_4, H_{\text{BERT-news}}) \quad (8)$$

This strategy integrates the analytical strengths of the Transformer and BERT with the MGCN, representing a paradigm shift in financial market analysis to capture a broad spectrum of factors affecting stock prices.

The integration of network motifs enriches feature extraction by leveraging both direct and motif-mediated relationships:

$$H^{(l+1)} = \tilde{A}_{mc} H_{\text{combined}} \quad (9)$$

The Motif Graph for stocks and stock news is quantitatively constructed and analyzed using the following two key equations, where  $m$  represents different motif types, such as M1 to M7.

The Eq. (10),

$$\tilde{A}_{mc} = \sum_m \alpha_m \tilde{A}_m \quad (10)$$

defines the motif combination matrix,  $\tilde{A}_{mc}$ , as a weighted sum of symmetrically normalized motif adjacency matrices  $\tilde{A}_m$  for each motif type  $m$ , weighted by  $\alpha_m$ . The coefficient  $\alpha_m$  indicates the relative importance or influence of each motif in the combined Motif Graph, allowing for a customized representation of different relational patterns between stocks and stock news.

The Eq. (11),

$$\tilde{A}_m = D_m^{-\frac{1}{2}} A_m D_m^{-\frac{1}{2}} \quad (11)$$

outlines the normalization process for each motif-specific adjacency matrix,  $A_m$ , into its symmetrically normalized form,  $\tilde{A}_m$ , using the degree matrix  $D_m$  of  $A_m$ .

Within the MGCN, the hidden features for stocks ( $h_s^{(l+1)}$ ) and stock-related news ( $h_n^{(l+1)}$ ) are updated based on their motif neighbors:

$$h_s^{(l+1)} = \sum_{m \in M} \sum_{i \in N_m(s)} \frac{1}{c_{ui}} \alpha_m H_{combined_i}^l \quad (12)$$

$$h_n^{(l+1)} = \sum_{m \in M} \sum_{u \in N_m(n)} \frac{1}{c_{ui}} \alpha_m H_{combined_u}^l \quad (13)$$

Here,  $h_s^{(l+1)}$  and  $h_n^{(l+1)}$  represent the updated hidden features for stocks and stock news, respectively, at layer  $l + 1$ , incorporating insights from both market history and current news. This methodology highlights a comprehensive approach to understanding the complex dynamics of financial markets.

**Prediction Layer.** The prediction layer is the apex of the motif-graph architecture, utilizing the refined representations extracted by the MGCN to predict future trends in the stock market. It employs a Fully Connected Neural Network (FNN) for binary classification, distinguishing between potential increases and decreases in stock prices. This distinction is made possible by isolating vector representations specific to stocks from the MGCN's output for processing via the FNN. The methodology for this binary classification is defined as follows:

Given the stock representation vector  $h_{\text{stock}}^{(L)}$ , derived from the MGCN's final layer  $L$ , the prediction layer undertakes binary classification:

$$y_{\text{pred}} = \sigma(\mathbf{W}_{\text{fnn}} h_{\text{stock}}^{(L)} + b_{\text{fnn}}) \quad (14)$$

where  $\sigma$  denotes the sigmoid activation function, appropriate for binary classification tasks, with  $\mathbf{W}_{\text{fnn}}$  and  $b_{\text{fnn}}$  representing the fully connected layer's weight matrix and bias vector, respectively. The model's efficacy in classifying stock price trends is evaluated using the cross-entropy loss:

$$\text{Loss} = \frac{1}{N} \sum_i [-y_i \cdot \log(\tilde{y}_i) + (1 - y_i) \cdot \log(1 - \tilde{y}_i)] \quad (15)$$

This strategy leverages the MGCN's analytical strength, enabling the prediction of stock price movements by interpreting complex interrelations and patterns, thereby providing a sophisticated method for stock trend prediction.

## 4 Experiments

In this section, we detail our experiment setup and present the results of our evaluation. We include comparisons with baseline models to demonstrate the effectiveness of our MGSP framework. Specifically, we select baseline models from traditional machine learning approaches such as LSTM and GRU, as well as methods used in recent studies, including HATs [9], AD-GAT [5], and DANSMP [25]. This comprehensive comparison underscores the capability of our

MGCN methodology to effectively leverage motif information for stock prediction.

#### 4.1 Experimental Settings

We conduct experiments on the Standard and Poor’s 500 dataset (S&P 500). Our dataset spans from January 1, 2018, to the end of 2019, encompassing a total of 500 trading days. The dataset is comprised of 400 training samples and 100 testing samples, corresponding to working days. Each sample encompasses three features: the S&P 500 price and the price sequence of all stocks over the preceding 30 days.

Given that stock prediction can be interpreted as a sequence classification challenge, the input  $x$  for each sample is represented by a matrix of dimensions  $(T, d)$ . Here,  $T$  signifies the length of the sequence, set at 30 days for the S&P 500 dataset, and  $d$  represents the dimension of the features for each day, which includes the individual stock price features, aggregated cluster features, and S&P 500 index features. Each of these feature vectors is concatenated to form a comprehensive representation of the stock’s behavior over the given period.

The labels for each sample are binary, indicating whether the stock price increased or decreased at the end of the prediction window, which is set to 7 days. This approach allows us to evaluate the model’s performance in predicting stock price movements based on historical data and derived features.

**Hyperparameter Settings.** To ensure a fair comparison, we tune the hyperparameters of all the baselines and our model using the same parameters, rather than the original parameters suggested in the papers. The fixed settings for all the models are as follows: the embedding size is set as 240; the batch size is 30.

For our MGSP, the regularization weight  $\lambda$  is set to  $1 \times 10^{-4}$ . The dropout rate is set to 0.6. The feature dimension is set to 50. The number of head attentions is set to 4. We use the Adam optimizer with a learning rate of 0.0001.

#### 4.2 Compared Baselines

Our proposed model is rigorously compared with a diverse set of methods across various categories to ensure a comprehensive evaluation.

**Classical Time-Series Methods.** We include classical time-series methods in our comparison to benchmark the predictive performance of our model against well-established techniques. The methods in this category are:

- **Long Short-Term Memory (LSTM):** Known for its ability to capture long-term dependencies in sequential data, LSTM is a powerful tool for time-series prediction.
- **Gated Recurrent Unit (GRU):** Similar to LSTM but with a simpler architecture, GRU is another popular method for handling sequential data.

- **Transformer:** This method has revolutionized sequential data processing with its self-attention mechanism, enabling it to capture long-range dependencies effectively.
- **Informer:** An extension of the Transformer model, Informer is specifically designed for long-sequence time-series forecasting with improved efficiency.

**Graph-Based Methods.** To further validate our model, we compare it against state-of-the-art graph-based methods that have shown success in stock price prediction:

- **Heterogeneous Attention Networks:** HATS [9] uses relational data to selectively aggregate information on different relation types, enhancing stock market prediction.
- **Attribute-Driven Graph Attention Networks:** AD-GAT [5] enhances the traditional GAT by incorporating adaptive mechanisms to better capture dynamic relationships.
- **Dual Attention Networks for Stock Movement Prediction:** DANSMP [25], leverages a market knowledge graph to model relationships between stocks and predict stock momentum. It integrates stock sequential embedding, stock relational embedding, and prediction layers, using dual attention networks to represent spillover signals.

To ensure a fair comparison, we adapted the core algorithms of various graph-based methods and integrated them into our framework, aligning their objective functions with our stock return prediction goal. We standardized graph structures by using our Graph and Motif Graph configurations, allowing for a direct performance comparison of different methods. By benchmarking our model against both traditional time-series and advanced graph-based approaches, we demonstrate the robustness and superior predictive performance of our proposed Motif-based Graph Convolutional Network (MGCN) framework.

### 4.3 Evaluation Metrics

In the classification of stock trend prediction, researchers often explore binary outcomes, specifically predicting whether stock prices will rise or fall. The advent of deep learning technology has significantly contributed to advancements in stock trend prediction methods [18]. Classification metrics play a crucial role in evaluating the effectiveness of these trend prediction models. The relevant formulas are provided below.

Accuracy is defined as the ratio of correctly predicted instances, encompassing both true positives and true negatives, to the total number of cases examined. The formula for accuracy is given as:

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

AUC, or Area Under the ROC Curve, is a metric employed to assess the performance of binary classification models. In the context of stock price movement, where predictions are dichotomized into ‘up’ or ‘down’, the ROC (Receiver Operating Characteristic) curve plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) across different model prediction thresholds. The AUC, which represents the area beneath the ROC curve and ranges from 0.0 to 1.0, serves as an indicator of model accuracy, with higher values denoting superior performance. The formulas for TPR (True Positive Rate) and FPR (False Positive Rate) are as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TP} + \text{FN}}$$

The AUC is calculated by integrating the TPR over the FPR:

$$\text{AUC} = \int \text{TPR} d(\text{FPR})$$

#### 4.4 Overall Performance Comparison

In this section, we first compare the performance of several classical time-series prediction models that exclusively use historical stock data without incorporating Graph Neural Networks (GNN). Next, we evaluate the performance of models that combine time-series prediction techniques with GNNs utilizing non-Motif Graph structures. Finally, we analyze the prediction models using Motif Graphs, and benchmark their performance against the baseline models.

To maintain consistency and fairness in our comparisons, we slightly modify the objective functions of all models to predict future stock returns. This comprehensive approach not only highlights the effectiveness of various prediction models but also provides insights into the potential benefits of integrating motif-based subgraphs in financial forecasting tasks.

From the experimental results shown in Table 1, we have the following observations:

**GCNs Improve Time-Series Models.** For all time-series baseline models, their performance are improved by simply integrating a two-layer GCN. This improvement can be attributed to the capability of GCNs to capture the relationships between nodes from their interactions, since time-series models treat all stocks as isolated entities and cannot exploit additional information from their interactions.

**Transformer-Like Architectures Remain Powerful.** Looking at the recent stock prediction models HATs, AD-GAT and DANSMP, they have no obvious advantage over pure Transformer and even underperform pure INFORMER. This is possible because these models cannot efficiently extract and combine

**Table 1.** Performance comparison between our MGSP framework and baseline methods on S&P 500 dataset.

Method	ACC(%)	AUC	F1-Score
LSTM	56.01	0.5953	0.6517
GRU	55.07	0.5631	0.6097
TRANSFORMER	59.52	0.6520	0.6671
INFORMER	61.65	0.6443	0.6581
LSTM+GCN	57.11	0.6087	0.6162
GRU+GCN	56.33	0.5676	0.6162
TRANSFORMER+GCN	60.33	0.6399	0.6810
INFORMER+GCN	62.23	0.5377	0.7231
HATs	56.47	0.5878	0.6066
AD-GAT	60.63	0.6379	0.6688
DANSMP	61.23	0.5750	0.7141
MGSP	63.38	0.6582	0.7306

information from time-series data and graph data. Improperly adapting GCNs into a large amount of stocks and news data may cause scalability issues.

#### Comparison with Baseline Models

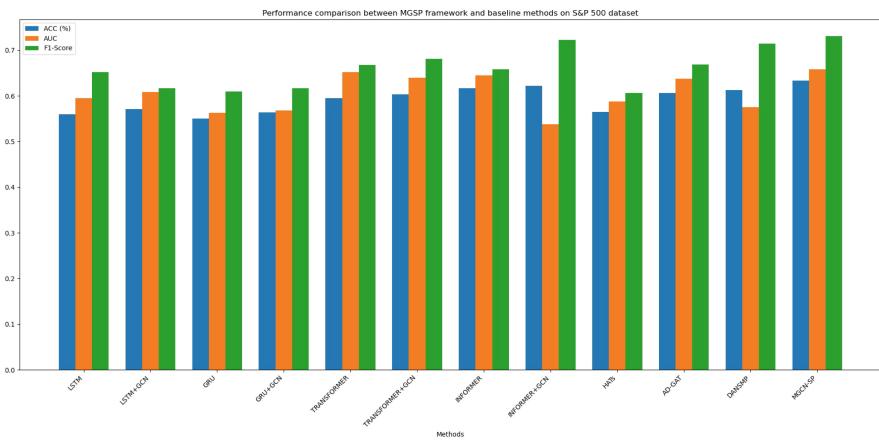
As shown in Fig. 2:

- **LSTM and GRU Models:** These recurrent models, both standalone and combined with GCN, exhibit lower performance across all metrics. This suggests that they are less effective at capturing the complex dependencies in stock-news data.
- **Transformer-based Models:** Standalone transformer models, as well as those combined with GCNs, show improved performance compared to LSTM and GRU models. However, MGSP still outperforms these methods, indicating the added value of motif-based structures in enhancing model performance.
- **Advanced Models (HATs, AD-GAT, DANSMP):** These models perform better than the basic recurrent models but still fall short of MGSP’s performance. For instance, while AD-GAT achieves a high AUC score of 0.6379, it lags behind MGSP in terms of accuracy and F1-score.

**MGSP Outperforms Baseline Models.** Our MGSP framework consistently outperform all baseline models in all evaluation metrics. Comparing with Transformer+GCN which uses two GCN layers, MGSP only have one MGCN layer yet achieves much better performance. This means motifs can efficiently exploit stock-news interactions with less number of layers than GCNs.

## 4.5 Parameter Sensitivity Analysis

In this section, we conduct a parameter sensitivity analysis to assess how different motif configurations impact the performance of the MGSP framework on the S&P 500 dataset. By testing various motifs, we identify which subgraphs most enhance predictive accuracy, AUC, and F1-Score. As shown in Table 2, we evaluate seven motif configurations (Motif 1, Motif 3, and Motif 6), revealing that motif selection significantly affects model performance, with specific motifs leading to superior accuracy, AUC, and F1-Score.



**Fig. 2.** Performance Comparison Between MGSP Framework and baseline models

**Table 2.** Performance comparison of different motif-based configurations in MGSP on the S&P 500 dataset.

Method	ACC(%)	AUC	F1-Score
MGSP (Motif 1)	60.33	0.6399	0.6810
MGSP (Motif 3)	62.61	0.6566	0.7240
MGSP (Motif 6)	63.38	0.6582	0.7306

From the results in Table 2, we observe several key findings:

Among all the tested motifs, MGSP with Motif 6 achieves the best overall performance, with an accuracy of 63.38%, AUC of 0.6582, and an F1-Score of 0.7306. This suggests that Motif 6 is the most effective in capturing the complex stock relationships and improving predictive performance.

Other motifs, such as Motif 3 also show strong performance, with accuracy and F1-Score values close to those of Motif 6. These motifs demonstrate a consistent ability to enhance the model’s ability to capture important stock trends.

Motif 1 perform the worst in terms of all metrics, indicating that these motifs may not be as effective in representing the underlying stock interactions. Their lower accuracy and F1-Score suggest that these motifs are less suitable for improving the model’s predictive capabilities.

**AUC Variation:** While Motif 3 achieve higher AUC values (0.6566 and 0.6585, respectively), their overall accuracy and F1-Score are slightly lower than those of Motif 6. This indicates that while they may be good at distinguishing between classes, they are not as effective in improving overall prediction accuracy.

The parameter sensitivity analysis reveals that selecting the right motif-based subgraph is crucial for maximizing the predictive power of the MGSP framework. Motif 6, in particular, provides the best balance between accuracy, AUC, and F1-Score, making it the optimal choice for this dataset.

## 5 Conclusion

In this paper, we propose a Motif-based Graph convolutional Network for Stock Trend Prediction. Specifically, we generate motif adjacency matrices from stock-news interactions and then use them for propagation of stocks and news embeddings. Extensive experiments on S&P 500 dataset demonstrate the superior performance of our proposed framework against multiple state-of-the-art models. Through analysis on the experimental results, we can conclude that our motif-based method can enhance stock prediction performance via efficiently capturing the relationships stocks and news. In future work, we plan to incorporate dynamic graph learning into our framework to explore the potential of modeling relational dynamics with considering motif structures.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Awartani, B.M., Corradi, V.: Predicting the volatility of the S&P-500 stock index via GARCH models: the role of asymmetries. *Int. J. Forecast.* **21**, 167–183 (2005)
2. Bao, W., Yue, J., Rao, Y.: A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE* **12**(7), e0180944 (2017)
3. Chen, Y., Wei, Z., Huang, X.: Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1655–1658 (2018)

4. Cheng, D., Yang, F., Wang, X., Zhang, Y., Zhang, L.: Knowledge graph-based event embedding framework for financial quantitative investments. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2221–2230 (2020)
5. Cheng, R., Li, Q.: Modeling the momentum spillover effect for stock prediction via attribute-driven graph attention networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 55–62 (2021)
6. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint [arXiv:1412.3555](https://arxiv.org/abs/1412.3555) (2014)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
8. Khemani, B., Patil, S., Kotecha, K., Tanwar, S.: A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. J. Big Data **11**(1), 18 (2024)
9. Kim, R., So, C.H., Jeong, M., Lee, S., Kim, J., Kang, J.: Hats: a hierarchical graph attention network for stock movement prediction. arXiv preprint [arXiv:1908.07999](https://arxiv.org/abs/1908.07999) (2019)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
11. Lee, M.C.: Using support vector machine with a hybrid feature selection method to the stock trend prediction. Expert Syst. Appl. **36**(8), 10896–10904 (2009)
12. Li, Q., Tan, J., Wang, J., Chen, H.: A multimodal event-driven LSTM model for stock prediction using online news. IEEE Trans. Knowl. Data Eng. **33**(10), 3323–3337 (2020)
13. Liu, J., Lu, Z., Du, W.: Combining enterprise knowledge graph and news sentiment analysis for stock price prediction (2019)
14. Liu, Q., Cheng, X., Su, S., Zhu, S.: Hierarchical complementary attention network for predicting stock price movements with news. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 1603–1606 (2018)
15. Long, J., Chen, Z., He, W., Wu, T., Ren, J.: An integrated framework of deep learning and knowledge graph for prediction of stock price trend: an application in chinese stock exchange market. Appl. Soft Comput. **91**, 106205 (2020)
16. Parray, I.R., Khurana, S.S., Kumar, M., Altalbe, A.A.: Time series data analysis of stock price movement using machine learning techniques. Soft. Comput. **24**(21), 16509–16517 (2020). <https://doi.org/10.1007/s00500-020-04957-x>
17. Picasso, A., Merello, S., Ma, Y., Oneto, L., Cambria, E.: Technical analysis and sentiment embeddings for market trend prediction. Expert Syst. Appl. **135**, 60–70 (2019)
18. Riva, F., Tognollo, A., Gardumi, F., Colombo, E.: Long-term energy planning and demand forecast in remote areas of developing countries: classification of case studies and insights from a modelling perspective. Energ. Strat. Rev. **20**, 71–89 (2018)
19. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)
20. Sezer, O.B., Ozbayoglu, A.M.: Algorithmic financial trading with deep convolutional neural networks: time series to image conversion approach. Appl. Soft Comput. **70**, 525–538 (2018)
21. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)

22. Wu, J., Li, Z., Herencsar, N., Vo, B., Lin, J.: A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimedia Syst.* **29**(3), 1751–1770 (2023)
23. Wu, Y., Chen, Y., Yin, Z., Ding, W., King, I.: A survey on graph embedding techniques for biomedical data: methods and applications. *Inf. Fusion* **100**, 101909 (2023)
24. Zhang, Y., Yu, J., Ruan, J., Wang, N., Madanian, S., Wang, G.: Motif-based graph attention network for web service recommendation. In: Proceedings of the 2023 Australasian Computer Science Week, pp. 143–146 (2023)
25. Zhao, Y., et al.: Stock movement prediction based on bi-typed and hybrid-relational market knowledge graph via dual attention networks. arXiv preprint [arXiv:2201.04965](https://arxiv.org/abs/2201.04965) (2022)



# VAGNN: Advancing the Generalization of Graph Neural Networks

Shuming Liang<sup>1</sup>(✉), Yu Ding<sup>2</sup>, Bin Liang<sup>1</sup>, Zhidong Li<sup>1</sup>, Siqi Zhang<sup>3</sup>,  
Yang Wang<sup>1</sup>, and Fang Chen<sup>1</sup>

<sup>1</sup> University of Technology Sydney, Ultimo, Australia

{shuming.liang, bin.liang, zhidong.li, yang.wang, fang.chen}@uts.edu.au

<sup>2</sup> University of Wollongong, Wollongong, Australia

dyu@uow.edu.au

<sup>3</sup> Zhejiang University, Hangzhou, China

siqi.zhang@uts.edu.au

**Abstract.** This paper introduces Virtual Adjacency Graph Neural Network (VAGNN). Compared to existing Graph Neural Networks (GNNs) that employ fixed schemes to construct neighborhoods for information aggregation, VAGNN leverages a virtual adjacency matrix to optimize neighborhoods by selectively excluding 1-hop neighbors while incorporating high-hop local neighbors and global nodes. Also, VAGNN allows for the selection of different attention mechanisms for aggregation and the incorporation of supplementary information into attention weights. The linear computational complexity of VAGNN makes it scalable for handling large graphs. Experimental evaluations on diverse real-world datasets validate the generalization and scalability capabilities of VAGNN. Parameter sensitivity analysis also reveals the importance of carefully balancing the inclusion of local and global information in VAGNN. This work lays the foundation for further exploration in developing more efficient techniques for virtual adjacency matrix construction and weighted aggregation functions, opening up possibilities for the design of more robust GNNs in the future.

**Keywords:** Graph Neural Networks · Graph Transformer · Neighborhood Information Aggregation · Generalization

## 1 Introduction

Over the past few years, a considerable array of Graph Neural Network (GNN) models has emerged [1, 2]. The majority of these models adhere to the neighborhood information aggregation algorithm [3] (also known as the message-passing mechanism [4]). This algorithm can be expressed by Eq. 1<sup>1</sup>, where the representation of a node  $v_i$  is iteratively updated by aggregating representations of its neighboring nodes and the node itself (i.e.,  $\mathbb{N}(v_i) \cup \{v_i\}$ ).

<sup>1</sup> For simplicity, we omit the residual connections, activation functions, etc.

$$\mathbf{h}_i^{(l+1)} = \text{AGGREGATE}^{(l)} \left( \left\{ \mathbf{h}_j^{(l)} \mid v_j \in \mathbb{N}(v_i) \cup \{v_i\} \right\} \right) \quad (1)$$

Various aggregation-based GNNs have been proposed [2], with notable innovations primarily centered around two key aspects: the construction of the neighborhood set  $\mathbb{N}(\cdot)$  and the development of the function  $\text{AGGREGATE}(\cdot)$ . As shown in Eq. 1, earlier GNNs [5,6] typically construct the set  $\mathbb{N}(v_i)$  with the direct utilization of all 1-hop neighbors of the node  $v_i$ . Alternatively, some models [7,8] construct  $\mathbb{N}(v_i)$  by selecting a subset of 1-hop neighbors of  $v_i$  to improve computational efficiency. Furthermore, several works incorporate high-order neighboring nodes of  $v_i$  into  $\mathbb{N}(v_i)$ , resulting in high-hop GNNs [4,9,10]. Drawing inspiration from the Transformer model in Natural Language Processing (NLP) [11],  $\mathbb{N}(v_i)$  is created to encompass all nodes in a graph in several graph Transformers [12–14].

In terms of the design of the function  $\text{AGGREGATE}(\cdot)$ , the straightforward approaches [5,7,15] involve set-based MEAN- or MAX-pooling over the set of node representations (i.e., the set of  $\left\{ \mathbf{h}_j^{(l)} \mid j \in \mathbb{N}(i) \cup \{i\} \right\}$  in Eq. 1). Subsequently, attention mechanisms were integrated into the  $\text{AGGREGATE}(\cdot)$  [6,16]. In the graph Transformer, supplementary information, such as shortest-path distance between nodes and edge features, is encoded as additional weights alongside attention weight to collectively govern the neighborhood aggregation [13,14,17].

The above analysis of diverse GNN architectures motivates us to propose a more general GNN framework called Virtual Adjacency Graph Neural Network (VAGNN). We introduce the term “virtual adjacency matrix” for GNNs to differentiate it from the actual adjacency matrix. In graph theory, the adjacency matrix describes the connections between nodes. In the case of a graph with  $N$  nodes, the adjacency matrix is an  $N \times N$  square matrix, where the  $i, j$ -th entry of the matrix is assigned a value of 1 if node  $v_j$  is a 1-hop neighbor of node  $v_i$ , and 0 otherwise. In GNNs like GCN [5] where  $\mathbb{N}(\cdot)$  only comprises all 1-hop neighbors of a node, the matrix representing all nodes’  $\mathbb{N}(\cdot)$ s aligns precisely with the adjacency matrix. However, for many GNNs [7,8] in which the construction of  $\mathbb{N}(\cdot)$  involves using partial 1-hop neighbors or higher-hop nodes of a node, the corresponding matrix is no longer being the actual adjacency matrix. Therefore, it is appropriate to refer to the matrix governing the neighborhood aggregation in GNNs as a virtual adjacency matrix.

Our proposed VAGNN is a general and scalable GNN framework. Firstly, with the concept of the virtual adjacency matrix, we can construct the neighborhood set  $\mathbb{N}(\cdot)$  from a general perspective. Unlike most existing GNNs that employ fixed methods for constructing the  $\mathbb{N}(\cdot)$ , VAGNN allows for the optimization of the  $\mathbb{N}(\cdot)$  by customizing the removal of 1-hop neighbors and the incorporation of high-hop local neighbors and global nodes. Moreover, VAGNN offers the capability to choose from various attention mechanisms for the aggregation function, and provides the option to incorporate  $y$  information for controlling the aggregation. Theoretically, VAGNN can be transformed into most existing GNNs and graph Transformers [4,6,10,13]. Also, it holds the potential

to achieve superior performance by optimizing the virtual adjacency matrix and aggregation methods tailored specifically for the given graph dataset.

The virtual adjacency matrix in VAGNN acts as a mask that determines whether the nodes require attention calculation and information aggregation or not. When implementing VAGNN based on sparse matrix operations, its computational complexity exhibits a linear relationship with the number of node pairs used for neighborhood aggregation (approximately equal to the sum of the sizes of all nodes'  $\mathbb{N}(\cdot)$ s). This linear complexity demonstrates the scalability of VAGNN, rendering it well-suited for large graphs [18]. Especially, by selectively choosing global nodes, VAGNN effectively overcomes the scalability challenges associated with utilizing global nodes in graph Transformers, reducing the storage complexity from  $O(N^2)$  or even  $O(N^3)$  to  $N$ , with  $N$  representing the total number of nodes in a graph [12, 13, 17].

We conduct extensive experiments on nine diverse real-world datasets sourced from the Open Graph Benchmark (OGB) [18]. The experimental results demonstrate the remarkable generalization and scalability capabilities of the proposed VAGNN. We also investigate the sensitivity of parameters involved in the construction of the virtual adjacency matrix. Our observations reveal that removing a small number of 1-hop neighboring nodes and incorporating a limited number of global nodes yield favorable outcomes, while an excessive utilization of global nodes leads to performance degradation. These findings highlight the importance of carefully balancing the inclusion of local and global information in GNNs.

## 2 Preliminary

Without loss of generality, we demonstrate this work on homogeneous graphs.

### 2.1 Notations

We denote a set using a blackboard bold uppercase letter (e.g.,  $\mathbb{V}$ ), a matrix using a bold uppercase letter, a vector using a bold lowercase letter, and a scalar using a lowercase letter.

Let  $\mathcal{G} = (\mathbb{V}, \mathbb{E}, \mathbf{X})$  denotes a graph  $\mathcal{G}$  with  $N$  nodes, where  $\mathbb{V}$  is the set of nodes,  $|\mathbb{V}| = N$ ,  $\mathbb{E}$  is the set of edges, and  $\mathbf{X} \in \mathbb{R}^{N \times f}$  is the feature matrix for all nodes.  $\mathbf{x}_i \in \mathbb{R}^f$  (i.e., the  $i$ -th row of  $\mathbf{X}$ ) is the feature vector of a node  $v_i$ . A set of nodes connected to a node  $v_i \in \mathbb{V}$  is the set of first-order or 1-hop neighbors of the node  $v_i$ . The adjacency matrix is an  $N \times N$  square matrix, storing the actual connections between nodes. It is denoted as  $\mathbf{A} \in \mathbb{R}^{N \times N}$ , in which the  $i, j$ -th entry (i.e.,  $a_{i,j}$ ) is 1 if an edge exists from node  $v_i$  to  $v_j$ , and 0 otherwise.

### 2.2 Graph Neural Networks

Most modern GNNs follow the neighborhood information aggregation algorithm, as expressed by Eq. 1. This equation describes the update rule of the representation of an individual node in GNNs. In fact, the aggregation-based GNNs can

be formalized based on matrix multiplication as below:

$$\mathbf{H}^{(l+1)} = \sigma \left( \overset{*}{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right), \quad (2)$$

where  $\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d}$  and  $\mathbf{H}^{(l+1)} \in \mathbb{R}^{N \times d'}$  are the input and output node representations in the  $l$ -th layer of a GNN. The  $i$ -th row of  $\mathbf{H}$  is denoted as  $\mathbf{h}_i \in \mathbb{R}^d$  which is a  $d$ -dimensional representation of the node  $v_i$ .  $\mathbf{H}^{(0)} = \mathbf{X}$  is the input feature matrix.  $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d'}$  is a trainable weight matrix.  $\sigma(\cdot)$  is an activation function.  $\overset{*}{\mathbf{A}}$  is a matrix governing the information propagation between nodes. For the plain GNNs like GCN [5],  $\overset{*}{\mathbf{A}}$  is a normalized version of the adjacency matrix  $\mathbf{A}$ . For high-hop GNNs [4, 10],  $\overset{*}{\mathbf{A}}$  would contain the connection information between a node and its high-hop neighbors.

### 3 The Proposed VAGNN

In this section, we introduce our Virtual Adjacency Graph Neural Network (VAGNN). To improve the readability of the subsequent equations, we provide a concise summary of the notations in Table 1.

**Table 1.** A brief summary of the notations used in VAGNN

Notation	Description
$l, L$	the $l$ -th layer and total layers $L$
$m, M$	the $m$ -th head and total heads $M$ in multiple-head attention module $\text{MultiHead}(\cdot)$
$d, d', d_O, d_A$	the dimension of the representation of a node
$\mathbf{A}$	the actual adjacency matrix
$\overset{*}{\mathbf{A}}$	the matrix for information propagation
$\overset{\vee}{\mathbf{A}}$	the virtual adjacency matrix
$\overset{\wedge}{\mathbf{A}}$	normalized version of the virtual adjacency matrix
$\hat{a}_{i,j}^{(l,m)}$	the normalized weight between nodes $v_i, v_j$
$\bar{a}_{i,j}^{(l,m)}$	the weight for information aggregation between $v_i, v_j$
$\check{a}_{i,j}$	the $i, j$ -th entry of the virtual adjacency matrix $\overset{\vee}{\mathbf{A}}$

We begin by providing an overview of the architecture of a VAGNN layer. Formally, the  $l$ -th layer of VAGNN can be described by the equation below:

$$\mathbf{H}^{(l+1)} = \text{MultiHead} \left( \mathbf{H}^{(l)} \right) \mathbf{W}_O^{(l)}, \quad (3)$$

where

$$\text{MultiHead}(\mathbf{H}^{(l)}) = \text{COMBINE}\left(\left\{\mathbf{H}^{(l)}\mathbf{W}^{(l,0)}, \hat{\mathbf{A}}^{(l,m)}\mathbf{H}^{(l)}\mathbf{W}^{(l,m)} : \forall m = 1, \dots, M\right\}\right). \quad (4)$$

$\text{MultiHead}(\mathbf{H}^{(l)})$  is a multi-head attention module where  $M$  is a hyper-parameter defining the number of attention heads. Multiple head attention in GNNs [6, 13] is an extension of the attention mechanism that is commonly used in sequence-based models such as language Transformers [11]. The  $\text{MultiHead}(\mathbf{H}^{(l)})$  allows an individual layer of VAGNN to learn different patterns and relationships within the representations  $\mathbf{H}^{(l)}$  through different attention heads simultaneously, improving the model's expressive power. The module involves three main steps: computing attention coefficients (i.e.,  $\hat{\mathbf{A}}^{(l,m)} \in \mathbb{R}^{N \times N}$  for the  $m$ -th head), aggregating information (i.e., the matrix multiplication  $\hat{\mathbf{A}}^{(l,m)}\mathbf{H}^{(l)}\mathbf{W}^{(l,m)}$ ), and combining representations performed by the function  $\text{COMBINE}(\cdot)$ . The function  $\text{COMBINE}(\cdot)$  can be Concatenation, MEAN, etc. The  $\mathbf{W}^{(l,m)} \in \mathbb{R}^{d \times d_O}$  is a trainable weight matrix. The weight matrix  $\mathbf{W}_O^{(l)}$  satisfies  $\mathbf{W}_O^{(l)} \in \mathbb{R}^{d_O(M+1) \times d'}$  or  $\mathbf{W}_O^{(l)} \in \mathbb{R}^{d_O \times d'}$  if  $\text{COMBINE}(\cdot)$  is Concatenation or MEAN, respectively. The hyper-parameters  $d$ ,  $d'$ , and  $d_O$  define the dimensions of the representations of each node in different learning stages.

In Eq. 4, the  $\mathbf{H}^{(l)}\mathbf{W}_A^{(l,0)}$  is a residual connection from  $\mathbf{H}^{(l)}$  towards  $\mathbf{H}^{(l+1)}$ . The  $\hat{\mathbf{A}}^{(l,m)}\mathbf{H}^{(l)}\mathbf{W}^{(l,m)}$  is the neighborhood aggregation in the  $m$ -th attention head of the  $l$ -th layer of VAGNN, where  $\hat{\mathbf{A}}^{(l,m)} \in \mathbb{R}^{N \times N}$  stores the normalized weights. The  $i, j$ -th entry of  $\hat{\mathbf{A}}^{(l,m)}$  (i.e.,  $\hat{a}_{i,j}^{(l,m)}$ ) is obtained by normalizing  $\bar{a}_{i,j}^{(l,m)}$  using the entries in the  $i$ -th row of  $\bar{\mathbf{A}}^{(l,m)}$  with the softmax function:

$$\hat{a}_{i,j}^{(l,m)} = \text{softmax}_j\left(\bar{a}_{i,j}^{(l,m)}\right) = \frac{\exp\left(\bar{a}_{i,j}^{(l,m)}\right)}{\sum_{k=0}^{N-1} \exp\left(\bar{a}_{i,k}^{(l,m)}\right)}. \quad (5)$$

The  $\bar{a}_{i,j}^{(l,m)}$ , as the  $i, j$ -th entry of  $\bar{\mathbf{A}}^{(l,m)} \in \mathbb{R}^{N \times N}$ , is the weight for information aggregation between node  $v_i$  and  $v_j$ . It is computed by the equations below:

$$\bar{a}_{i,j}^{(l,m)} = \begin{cases} \text{ATTN}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right) + \text{EXT}(v_i, v_j), & \text{if } \check{a}_{i,j} = 1 \\ 0, & \text{if } \check{a}_{i,j} = 0 \end{cases}, \quad (6)$$

where the function  $\text{ATTN}\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}\right) \rightarrow \mathbb{R}$  calculates the attention coefficient between node  $v_i$  and  $v_j$  using their corresponding representation  $\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}$ . The function  $\text{EXT}(v_i, v_j) \rightarrow \mathbb{R}$  introduces an extra weight between node  $v_i$  and  $v_j$ , enabling the incorporation of information that is challenging to be acquired solely by the attention coefficient. For instance, this function can incorporate structural information such as the shortest-path distance between node  $v_i$  and

$v_j$  [13]. The  $\tilde{a}_{i,j}$  is the  $i, j$ -th entity of the matrix  $\overset{\vee}{\mathbf{A}}$ . Herein  $\overset{\vee}{\mathbf{A}}$  is our proposed virtual adjacency matrix for VAGNN. In Eq. 6, we can see that  $\overset{\vee}{\mathbf{A}}$  acts as a mask, determining which node pairs' weights need to be computed.

Equation 6 presents the fundamental elements underlying the proposed VAGNN. This equation summarizes and generalizes the common characteristics of GNNs and graph Transformers [6, 10, 13], endowing VAGNN with powerful expressiveness. In the following sections, we will provide a comprehensive exposition of the key components of this equation.

### 3.1 Attention Methodologies

The attention mechanism enables the GNN model to learn adaptive importance weights between two nodes, allowing every node to differentiate between neighbors during the process of aggregating neighborhood information in the GNN.

As shown in Eq. 6, the  $\text{ATTN}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)})$  is the attention function used to calculate the attention coefficient between nodes  $v_i, v_j$  based on the representations of those two nodes. We summarize several main methods for the function  $\text{ATTN}(\cdot, \cdot)$  as follows.

*Cosine Similarity-Based Attention* [16]. This is the simplest method for computing attention weights. It is typically computed as:

$$\text{ATTN}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}) = \beta^{(l,m)} \cos(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}), \quad (7)$$

where  $\beta^{(l,m)} \in \mathbb{R}$  is a learnable parameter for the  $m$ -th attention head in the  $l$ -th layer,  $\cos(x, y) = xy^\top / |x||y|$ , and  $|x|$  is the Euclidean norm of the vector  $x$ .  $\cos(\cdot, \cdot)$  calculates the cosine of the angle between two vectors, tending to learn higher attention weights for more relevant representation vectors.

*GAT-Type Attention* [19, 20]. In GAT [19],  $\text{ATTN}(\cdot, \cdot)$  can be formalized as:

$$\text{ATTN}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}) = \sigma \left( \mathbf{q}^{(l,m)} \cdot \left( \text{CAT} \left( \mathbf{h}_i^{(l)} \mathbf{W}_Q^{(l,m)}, \mathbf{h}_j^{(l)} \mathbf{W}_Q^{(l,m)} \right) \right)^\top \right), \quad (8)$$

where  $\sigma(\cdot)$  is an activation function, the  $\cdot^\top$  represents transposition,  $\text{CAT}(\dots)$  is the concatenation operation,  $\mathbf{W}_Q^{(l,m)} \in \mathbb{R}^{d \times d_A}$  is a trainable weight matrix shared across all nodes, and  $\mathbf{q}^{(l,m)} \in \mathbb{R}^{2d_A}$  is a weight vector. Compared to cosine similarity-based attention, GAT-type attention employs a shared linear transformation to hidden states across all nodes in the graph, which could improve the expressive power of the model.

*Self-attention Mechanism* [12, 13, 21]. Self-attention has demonstrated great success in Transformers in NLP [11] and computer vision [11, 22]. Recent efforts have been made to extend it to the field of GNNs. For self-attention mechanism,  $\text{ATTN}(\cdot, \cdot)$  can be expressed as:

$$\text{ATTN}(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}) = \frac{\left( \mathbf{h}_i^{(l)} \mathbf{W}_Q^{(l,m)} \right) \cdot \left( \mathbf{h}_j^{(l)} \mathbf{W}_K^{(l,m)} \right)^\top}{\sqrt{d_A}}, \quad (9)$$

where  $\mathbf{W}_Q^{(l,m)} \in \mathbb{R}^{d \times d_A}$  and  $\mathbf{W}_K^{(l,m)} \in \mathbb{R}^{d \times d_A}$  are trainable weight matrices shared across all nodes for the  $m$ -th attention head in the  $l$ -th layer,  $\frac{1}{\sqrt{d_A}}$  is a scaling factor [11, 13].

### 3.2 The Extra Weight

In Eq. 6, we can see that the learning of the attention weight between two nodes  $v_i, v_j$  primarily relies on the node-wise representations, i.e.,  $h_i$  and  $h_j$ . However, when characterizing the relationship between two nodes in graph data, besides the reliance on node representations, there exist other factors, such as the shortest-path distance between two nodes, the weight of the edge, etc. To effectively capture these extra pieces of information, we introduce a function  $\text{EXT}(v_i, v_j)$  and add it to the attention weight, thus providing a better description of the relationship between the two nodes.

$\text{EXT}(v_i, v_j)$  serves as a valuable complement to the attention weight, introducing an explicit component in the GNN design. We provide a specific form of  $\text{EXT}(v_i, v_j)$  as follows:

$$\text{EXT}(v_i, v_j) = \alpha^{(l,m)} \text{SPD}(v_i, v_j) + \dots + \text{EW}(v_i, v_j), \quad (10)$$

where  $\text{SPD}(v_i, v_j)$  calculate the shortest-path distance between two nodes  $v_i, v_j$ ,  $\alpha^{(l,m)} \in \mathbb{R}$  is a trainable weight for the function  $\text{SPD}(\cdot, \cdot)$ ,  $\text{EW}(v_i, v_j)$  can be a function to aggregate the features of edges between  $v_i, v_j$ . Previous works [12–14, 17] have explored incorporating information independent of node features to the attention weight. VAGNN can generalize most of them. For example, Graphomer [13] proposes a method of encoding edge features. That method can be formalized as  $\text{EW}(v_i, v_j) = \frac{1}{E} \sum_{e=1}^E \mathbf{x}_e (\mathbf{w}_e^{(l,m)})^\top$ , where  $E$  is the number of edges in (one of) the shortest path between  $v_i, v_j$ , the edges is indexed from  $v_i$  to  $v_j$ ,  $\mathbf{x}_e$  is a feature vector of the  $e$ -th edge, and  $\mathbf{w}_e^{(l,m)}$  is the  $e$ -th trainable weight vector for  $\text{EW}(\cdot, \cdot)$ .

### 3.3 Virtual Adjacency Matrix

In this work, we investigate various existing GNNs and identify a common characteristic among these models. Specifically, we observe that many GNNs modify the original adjacency matrix for neighborhood information aggregation, with the purpose of improving model expressiveness or computational efficiency. For example, GraphSAGE [7] removes some of the actual edges (i.e., 1-hop neighbors) to enhance the computational efficiency of GNNs. High-hop GNNs, such as DEGNN [10], MixHop [9], and KPGNN [4], incorporate higher-hop neighboring nodes along with 1-hop nodes for neighborhood aggregation. Graph Transformers [12–14, 23] employ the self-attention mechanism that takes into account all nodes, where the matrix for the neighborhood aggregation is equivalent to  $\{1\}^{N \times N}$ . In graph theory, the adjacency matrix describes the actual connections between nodes in a graph. Nevertheless, these modified adjacency matrices

employed for neighborhood aggregation in GNNs have lost the functionality of the original adjacency matrix. Hence, we propose the term virtual adjacency matrix as a more appropriate descriptor for the modified matrix.

The virtual adjacency matrix is an important concept in our proposed VAGNN. As shown in Eq. 6, the virtual adjacency matrix serves as a mask that controls which node pairs can participate in the neighborhood information aggregation in the GNN. We study the methods for constructing the virtual adjacency matrix and categorize them into three main classes outlined below:

- Removal of actual edges. This class of methods involves converting certain 1-hop neighbors into non-1-hop neighbors.
- 1-hop conversion of high-hop local neighboring nodes. Here, high-hop local neighbors are transformed into 1-hop neighbors.
- 1-hop conversion of global nodes. This class of methods focuses on converting global nodes into 1-hop neighbors.

The latter two classes of methods involve the addition of virtual edges between nodes, enabling high-hop neighbors and global nodes to directly participate in the aggregation process of the target node. It is important to note that high-hop neighboring nodes and global nodes are different. High-hop neighbors are not directly connected to the central node, but they have small shortest-path distances to that node. In contrast, global nodes may have significantly larger shortest-path distances or even no path connection to the target node.

Diverse methods can be developed for the three aforementioned classes of methods for the virtual adjacency matrix construction. For example, regarding the removal of actual edges, potential approaches could be random edge removal or edge removal based on weight. In the case of 1-hop neighbor conversion, the methods could include transforming all 2-hop local neighbors, converting partial global nodes, etc. Note that our work primarily centers around introducing VAGNN and its generalization capability. We defer the investigation of specific construction methods to future research.

### 3.4 The Expressiveness of VAGNN

Compared to previous GNNs, the proposed VAGNN allows us to approach and design the GNN models from a more general perspective. VAGNN is not restricted to a specific architecture. It provides a general framework that is capable of being transformed into various existing GNNs and graph Transformers [5, 6, 13]. This generalization is achieved through customizable options in VAGNN, including designing the functions of  $\text{ATTN}(\cdot, \cdot)$  and  $\text{EXT}(\cdot, \cdot)$ , and constructing the virtual adjacency matrix.

Specifically, when the virtual adjacency matrix is the original adjacency matrix, VAGNN without any attention mechanism becomes pioneering GNN models like GCN [5]. Building upon this, the introduction of a residual connection gives rise to JK-GNN [15], GCNII [24] and DeeperGCN [25], while the inclusion of an attention mechanism leads to the development of GAT [6], AGNN [16],

etc. If the virtual adjacency matrix is constructed by removing certain edges from the original adjacency, it takes the form of GraphSAGE [7], FastGCN [8], etc. Alternatively, If the virtual adjacency matrix incorporates high-hop neighbors, it resembles DEGNN [10], MixHop [9] and KPGNN [4]. Lastly, if the virtual adjacency matrix consists entirely of *ones*, along with the self-attention mechanism and additional weights based on structural information, VAGNN is transformed into the architecture of graph Transformers [12–14, 23].

In addition to its ability to be configured as existing GNNs and graph Transformers, VAGNN offers valuable assistance in designing potentially more effective GNNs. Firstly, it enables a balance between local and global nodes by controlling the proportion of global nodes used. Moreover, unlike traditional graph Transformers, the utilization of a virtual adjacency matrix with partial global nodes allows VAGNN to be implemented based on sparse matrix operations. This can reduce the storage complexity from  $O(N^2)$  and  $O(N^3)$  of the dense matrix addition and multiplication ( $N$  is the number of total nodes) in graph Transformers [13, 17, 23] to a linear scale with the number of selected node pairs, thereby enabling VAGNN to exhibit excellent computational scalability even on large graphs. Regarding the complexity of constructing the virtual adjacency matrix, three classes of methods can be performed based on the original adjacency matrix. For instance, the 1-hop conversion of high-hop local neighbors can be based on the multiplication of the original adjacency matrix (e.g.,  $A^2$  represents all neighbors within 2 hops [9]).

## 4 Experiments

This section presents the experimental evaluation of our proposed VAGNN. The objective of this paper is to introduce the VAGNN model and demonstrate its expressiveness and generalization capabilities. We design the experiments to serve two purposes primarily: 1) Examination of generalization capabilities and 2) Sensitivity study on parameters of the virtual adjacency matrix construction.

### 4.1 Experimental Settings

**Datasets.** All datasets used in our experiments are selected from the Open Graph Benchmark (OGB) [18]. We conduct experiments on three prominent graph tasks, i.e., node classification, link prediction, and graph classification. Accordingly, the datasets are ogbn-products, ogbn-proteins, and ogbn-arxiv for node classification; ogbl-ppa, ogbl-collab, and ogbl-ddi for link prediction; and ogbg-molhiv, ogbg-molpcba, and ogbg-code2 for graph classification.

OGB provides official evaluation protocols [18] for all benchmark datasets. We strictly follow these protocols for the training/validation/test data split and the evaluation metrics. We refer to [18] for further details about OGB datasets. All our experimental results are reported on the test set, with mean and standard deviation computed across 5 trials.

**Table 2.** Results of node classification.

	ogbn-products (Accuracy %)	ogbn-proteins (AUC %)	ogbn-arxiv (Accuracy %)
GCN [5]	75.64 $\pm$ 0.21	76.53 $\pm$ 0.35	71.74 $\pm$ 0.29
VAGNN(GCN)	75.77 $\pm$ 0.19	77.31 $\pm$ 0.41	71.38 $\pm$ 0.31
GraphSAGE [7]	78.29 $\pm$ 0.16	77.68 $\pm$ 0.20	71.49 $\pm$ 0.07
VAGNN(SAGE)	78.03 $\pm$ 0.35	78.04 $\pm$ 0.37	71.25 $\pm$ 0.18
DEGNN [10]	80.14 $\pm$ 0.64	85.01 $\pm$ 0.72	71.62 $\pm$ 0.17
VAGNN(DE)	80.37 $\pm$ 0.91	85.17 $\pm$ 0.65	71.93 $\pm$ 0.21
AGNN [16]	78.84 $\pm$ 0.35	79.11 $\pm$ 0.35	71.81 $\pm$ 0.37
VAGNN(AGNN)	78.92 $\pm$ 0.43	79.02 $\pm$ 0.34	71.74 $\pm$ 0.57
GAT [6]	79.23 $\pm$ 0.78	79.35 $\pm$ 0.44	71.97 $\pm$ 0.35
VAGNN(GAT)	79.19 $\pm$ 0.59	79.41 $\pm$ 0.38	71.86 $\pm$ 0.25
Graphormer [13]	—	—	—
VAGNN(Trans)	81.14 $\pm$ 0.67	80.73 $\pm$ 0.51	72.14 $\pm$ 0.45
MLP	61.06 $\pm$ 0.08	72.04 $\pm$ 0.48	55.50 $\pm$ 0.23
Node2vec [26]	72.49 $\pm$ 0.10	68.81 $\pm$ 0.65	70.07 $\pm$ 0.13
UniMP [27]	82.56 $\pm$ 0.31	86.42 $\pm$ 0.08	73.97 $\pm$ 0.15
<b>VAGNN(CosA)</b>	81.53 $\pm$ 0.57	85.17 $\pm$ 0.28	71.64 $\pm$ 0.29
<b>VAGNN(GatA)</b>	81.91 $\pm$ 0.62	86.29 $\pm$ 0.52	72.06 $\pm$ 0.77
<b>VAGNN(SelfA)</b>	82.26 $\pm$ 0.79	86.71 $\pm$ 0.69	72.15 $\pm$ 0.63

**Implementation.** We implement our algorithm based on PyTorch, PyTorch Geometric [28], and OGB [18]. The model is trained with Adam optimizer [29] with the learning rate decayed using the ExponentialLR method [30]. All experiments are conducted on a Linux machine with two 32-core CPUs, 512G RAM, and two NVIDIA A100 GPUs. Code is available at [github.com/astroming/vagnn](https://github.com/astroming/vagnn).

## 4.2 Examination of VAGNN Generalization Capabilities

In order to experimentally examine the generalization capability of VAGNN, we carefully choose several representative GNN models, including 1) GCN [5]. It uses all 1-hop nodes for neighborhood information aggregation without any attention weights; 2) GraphSAGE [7]. It is similar to GCN, but it differs in the selection of the neighborhood set. Instead of utilizing all 1-hop neighbors as done in GCN, GraphSAGE employs a fixed-size neighborhood set sampled uniformly from the 1-hop neighbors; 3) DEGNN [10]. It is a high-hop GNN framework where the neighboring nodes attending the central node’s information aggregation are selected based on the shortest-path distance between the neighbors and the central node; 4) AGNN [16]. It is a 1-hop GNN and employs a cosine similarity attention mechanism as shown in Eq. 7; 5) GAT [6]. It is a 1-hop GNN that

**Table 3.** Results of link prediction

	ogbl-ppa (Hits@100 %)	ogbl-collab (Hits@50 %)	ogbl-ddi (Hits@20 %)
GCN [5]	54.19 $\pm$ 1.21	49.35 $\pm$ 0.47	88.42 $\pm$ 1.34
VAGNN(GCN)	54.23 $\pm$ 1.17	49.19 $\pm$ 0.53	88.52 $\pm$ 2.83
GraphSAGE [7]	49.07 $\pm$ 2.18	48.10 $\pm$ 0.81	87.37 $\pm$ 3.68
VAGNN(SAGE)	50.15 $\pm$ 2.73	48.39 $\pm$ 0.75	88.09 $\pm$ 4.91
DEGNN [10]	51.33 $\pm$ 0.87	50.29 $\pm$ 0.27	89.94 $\pm$ 5.23
VAGNN(DE)	51.61 $\pm$ 1.31	50.29 $\pm$ 0.32	89.17 $\pm$ 3.94
AGNN [16]	54.58 $\pm$ 2.56	49.24 $\pm$ 0.48	89.61 $\pm$ 3.54
VAGNN(AGNN)	54.37 $\pm$ 2.10	49.17 $\pm$ 0.26	90.02 $\pm$ 3.84
GAT [6]	56.27 $\pm$ 3.25	49.17 $\pm$ 0.55	91.21 $\pm$ 5.98
VAGNN(GAT)	55.98 $\pm$ 2.79	49.33 $\pm$ 0.19	90.79 $\pm$ 7.12
Graphomer [13]	—	—	—
VAGNN(Trans)	53.29 $\pm$ 4.41	50.05 $\pm$ 0.72	92.31 $\pm$ 6.25
MF [31]	32.29 $\pm$ 0.94	38.86 $\pm$ 0.49	13.68 $\pm$ 4.75
DeepWalk [32]	28.88 $\pm$ 1.63	50.37 $\pm$ 0.34	26.42 $\pm$ 6.10
Node2vec [26]	22.26 $\pm$ 0.83	48.88 $\pm$ 0.54	23.26 $\pm$ 2.09
SEAL [33]	48.80 $\pm$ 4.56	54.71 $\pm$ 0.79	30.56 $\pm$ 3.86
<b>VAGNN(CosA)</b>	55.37 $\pm$ 2.46	50.91 $\pm$ 0.37	91.42 $\pm$ 2.57
<b>VAGNN(GatA)</b>	57.68 $\pm$ 3.28	52.62 $\pm$ 0.73	92.68 $\pm$ 5.51
<b>VAGNN(SelfA)</b>	57.73 $\pm$ 3.35	52.59 $\pm$ 0.65	94.07 $\pm$ 3.66

incorporates an attention mechanism as shown in Eq. 8; and 6) Graphomer [13]. It employs self-attention on all nodes.

We use the term VAGNN(GCN) to denote the VAGNN that has been configured exactly according to the GCN. Similarly, other VAGNN variations, such as VAGNN(SAGE), follow the same nomenclature convention. VAGNN(Trans) employs the self-attention mechanism in graph Transformers. However, we utilize partial global nodes for each node in VAGNN(Trans) when using all nodes to calculate the self-attention is challenging on some datasets.

Tables 2, 3, and 4 present the results of node classification, link prediction, and graph classification, respectively. The results of the VAGNN variants closely align with those of the corresponding GNN models across most datasets, demonstrating the effectiveness of the VAGNN in generalizing various GNN architectures within a single framework. These results also confirm that the majority of GNNs and graph Transformers fundamentally operate within the form of neighborhood information aggregation (Eq. 1).

To attain the optimal performance of VAGNN, we carefully select the effective combination of parameters for constructing the virtual adjacency matrix. We also employ the extra weight as shown in Eq. 10 where we follow the

**Table 4.** Results of graph classification

	ogbg-molhiv (AUC %)	ogbg-molpcba (AP %)	ogbg-code2 (F1 Score %)
GCN [5]	76.06 ± 0.97	23.96 ± 0.41	15.07 ± 0.18
VAGNN(GCN)	76.25 ± 0.64	23.72 ± 0.70	15.43 ± 0.35
GraphSAGE [7]	76.35 ± 1.32	25.05 ± 0.17	15.13 ± 0.39
VAGNN(SAGE)	76.18 ± 1.09	25.17 ± 0.31	15.36 ± 0.25
DEGNN [10]	79.84 ± 1.25	28.15 ± 0.41	15.81 ± 0.21
VAGNN(DE)	79.62 ± 1.47	28.26 ± 0.63	15.78 ± 0.25
AGNN [16]	76.36 ± 1.13	26.84 ± 0.22	15.37 ± 0.15
VAGNN(AGNN)	76.41 ± 1.05	27.08 ± 0.31	15.26 ± 0.12
GAT [6]	76.81 ± 1.51	27.03 ± 0.29	15.69 ± 0.10
VAGNN(GAT)	76.83 ± 1.29	27.14 ± 0.33	15.57 ± 0.21
Graphormer [13]	80.51 ± 0.53	31.39 ± 0.32	17.52 ± 0.14
VAGNN(Trans)	80.95 ± 1.07	31.58 ± 0.51	17.71 ± 0.23
GIN [3]	77.78 ± 1.30	22.66 ± 0.28	14.95 ± 0.23
PNA [34]	79.05 ± 1.32	28.38 ± 0.35	15.70 ± 0.32
<b>VAGNN(CosA)</b>	<b>78.91 ± 1.47</b>	<b>29.12 ± 0.27</b>	<b>16.14 ± 0.21</b>
<b>VAGNN(GatA)</b>	<b>79.49 ± 1.62</b>	<b>29.88 ± 0.54</b>	<b>17.11 ± 0.23</b>
<b>VAGNN(SelfA)</b>	<b>80.96 ± 1.14</b>	<b>31.72 ± 0.48</b>	<b>17.72 ± 0.25</b>

methods in [13]. Together with different attention mechanisms, i.e., cosine similarity-based attention, GAT-type attention, and self-attention mechanism, we denote such optimal VAGNN models as VAGNN(CosA), VAGNN(GatA), or VAGNN(SelfA). In Tables 2, 3, and 4, it can be found that the VAGNN(SelfA) consistently achieves superior performance across most datasets, compared to VAGNN(CosA), VAGNN(GatA) and previous models. These outcomes serve as compelling evidence for the potential effectiveness of our VAGNN.

Empirically, VAGNN demonstrates linear computational complexity and scalability with respect to both computational resources and time efficiency. The evaluation processes show that VAGNN consistently maintains low overhead across various graph sizes, confirming its suitability for large graphs.

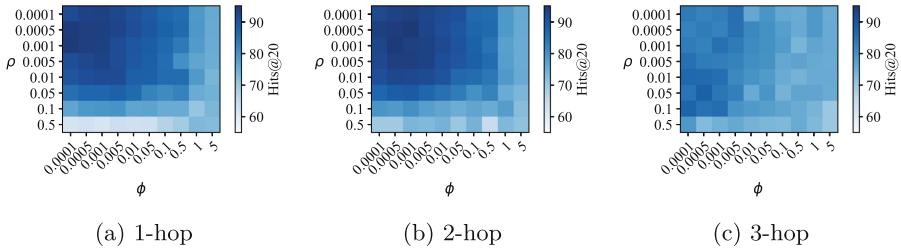
### 4.3 Parameter Sensitivity Study on Virtual Adjacency Matrix

The virtual adjacency matrix serves as a core component of VAGNN, governing the participation of nodes in neighborhood information aggregation. As outlined in Sect. 3.3, the methods used to construct the virtual adjacency matrix can be categorized into three classes, which correspond to the control over the proportions of three types of nodes involved in the aggregation, i.e., actual 1-hop neighboring nodes, high-hop local neighbors, and global nodes. In this parameter

sensitivity analysis, our objective is to investigate how different proportions of these three types of nodes affect the performance of the model.

Experimentally, for the removal of 1-hop neighbors, we randomly eliminate  $\rho|\mathbb{E}|$  actual edges (equivalent to converting 1-hop neighbors to non-1-hop nodes). Herein,  $|\mathbb{E}|$  represents the total number of actual edges. Regarding the 1-hop conversion of high-hop local neighbors, we explore three settings: using the original 1-hop neighbors, converting all local neighboring nodes within 2 hops into 1-hop neighbors, and converting those within 3 hops into 1-hop. Lastly, for the 1-hop conversion of global nodes, we randomly select  $\phi|\mathbb{E}|$  global nodes.

We use the dataset *ogbl-ddi* to investigate the sensitivity of parameters. The results are presented in Fig. 1. For the parameter  $\rho$ , it can be observed that VAGNN attains its optimal performance at relatively small values of  $\rho$ , with the majority of peaks occurring around 0.0005 of  $\rho$ . As  $\rho$  increases, the performance of VAGNN gradually declines and becomes particularly poor when  $\rho \geq 0.1$ .



**Fig. 1.** The results of the parameter sensitivity study on the virtual adjacency matrix construction. Three plots (a) (b) (c) correspond to three different configurations for the 1-hop conversion of high-hop local neighboring nodes. For example, in plot (a), the term “1-hop” represents the virtual adjacency matrix that uses the local neighboring nodes within 1 hop (i.e., only the actual 1-hop nodes). In plot (b), the term “2-hop” refers to the conversion of local neighboring nodes within 2 hops into 1-hop neighbors in the virtual adjacency matrix. The colors of grid cells in the plots indicate the performance (Hits@20) of the proposed VAGNN on the *ogbl-ddi* dataset.

For the 1-hop conversion of high-hop local neighboring nodes, the results on the 1-hop and 2-hop (Fig. 1a and Fig. 1b) are significantly better than the results on the 3-hop (Fig. 1c). Remarkably, the most favorable results are observed on the 1-hop conversion of 2-hop nodes (Fig. 1b), suggesting that the incorporation of high-hop nodes could effectively improve the performance of the VAGNN model. The results on the 1-hop conversion of 3-hop neighbors exhibit minimal variance. This result can be attributed to the characteristics of the *ogbl-ddi* dataset. Specifically, the *ogbl-ddi* is a graph with 4267 nodes and an average node degree of around 500. We observe that for every node in *ogbl-ddi*, its neighbors within 3 hops would approximately account for 95% of the entire graph’s nodes. Consequently, the virtual adjacency matrix based on the 1-hop conversion of 3-hop neighbors on the *ogbl-ddi* dataset is similar to the setting of the standard

graph Transformer, where all nodes are considered as 1-hop neighbors in the virtual adjacency matrix.

For the parameter  $\phi$ , Fig. 1 demonstrates that smaller values of  $\phi$  generally yield superior performance, while larger values of  $\phi$  lead to a noticeable decline. Moreover, when  $\phi \geq 1$ , the model exhibits poor and similar outcomes. This could be attributed to the dense nature of the graph in the ogbl-ddi dataset (4267 nodes and a degree of around 500). When  $\phi \geq 1$ , nearly all global nodes for each node would be converted into its 1-hop neighbors in the virtual adjacency matrix. In real-world graphs, the number of global nodes is often far larger than that of local neighboring nodes for a node. The selection of an appropriate number of global nodes for the virtual adjacency matrix holds crucial importance.

## 5 Conclusion

In this paper, we present VAGNN, a general and scalable GNN framework. VAGNN offers the capability to customize the neighborhood for aggregation, allowing for the exclusion of 1-hop neighbors and the incorporation of high-hop local neighbors and global nodes. By customizing the construction of the virtual adjacency matrix and utilizing various attention mechanisms, VAGNN can be transformed into most existing GNNs and transformers, demonstrating the generalization of VAGNN. Additionally, we perform a parameter sensitivity analysis, highlighting the importance of carefully selecting and balancing the inclusion of local and global information in GNNs.

## References

1. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **32**(1), 4–24 (2020)
2. Hoang, V.T., Jeon, H.-J., You, E.-S., Yoon, Y., Jung, S., Lee, O.-J.: Graph representation learning and its applications: a survey. *Sensors* **23**(8), 4168 (2023)
3. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (2018)
4. Feng, J., Chen, Y., Li, F., Sarkar, A., Zhang, M.: How powerful are k-hop message passing graph neural networks. In: Advances in Neural Information Processing Systems, vol. 35, pp. 4776–4790 (2022)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)
6. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
7. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
8. Chen, J., Ma, T., Xiao, C.: FastGCN: fast learning with graph convolutional networks via importance sampling. In: International Conference on Learning Representations (2018)

9. Abu-El-Haija, S., et al.: MixHop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning, pp. 21–29. PMLR (2019)
10. Li, P., Wang, Y., Wang, H., Leskovec, J.: Distance encoding: design provably more powerful neural networks for graph representation learning. *Adv. Neural. Inf. Process. Syst.* **33**, 4465–4478 (2020)
11. Vaswani, A. et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
13. Ying, C., et al.: Do transformers really perform badly for graph representation? In: Advances in Neural Information Processing Systems, vol. 34 (2021)
14. Chen, D., O’Bray, L., Borgwardt, K.: Structure-aware transformer for graph representation learning. In: International Conference on Machine Learning, pp. 3469–3489. PMLR (2022)
15. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.I., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: International Conference on Machine Learning, pp. 5453–5462. PMLR (2018)
16. Thekumparampil, K.K., Wang, C., Oh, S., Li, L.-J.: Attention-based graph neural network for semi-supervised learning. arXiv preprint [arXiv:1803.03735](https://arxiv.org/abs/1803.03735) (2018)
17. Park, W., Chang, W., Lee, D., Kim, J., et al.: GRPE: relative positional encoding for graph transformer. In: ICLR2022 Machine Learning for Drug Discovery (2022)
18. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs. In: Advances in Neural Information Processing Systems, vol. 33, pp. 22118–22133 (2020)
19. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
20. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: International Conference on Learning Representations (2022)
21. Zhang, J., Zhang, H., Xia, C. and Sun, L.: Graph-BERT: only attention is needed for learning graph representations. arXiv preprint [arXiv:2001.05140](https://arxiv.org/abs/2001.05140) (2020)
22. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. In: International Conference on Learning Representations (2021)
23. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. arXiv preprint [arXiv:2012.09699](https://arxiv.org/abs/2012.09699) (2020)
24. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: International Conference on Machine Learning, pp. 1725–1735. PMLR (2020)
25. Li, G., Xiong, C., Thabet, A., Ghanem, B.: DeeperGCN: all you need to train deeper GCNs. arXiv preprint [arXiv:2006.07739](https://arxiv.org/abs/2006.07739) (2020)
26. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
27. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: unified message passing model for semi-supervised classification. arXiv preprint [arXiv:2009.03509](https://arxiv.org/abs/2009.03509) (2020)
28. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

29. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014)
30. Li, Z., Arora, S.: An exponential learning rate schedule for deep learning. In: International Conference on Learning Representations (2020)
31. Menon, A.K., Elkan, C.: Link prediction via matrix factorization. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011. LNCS (LNAI), vol. 6912, pp. 437–452. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-23783-6\\_28](https://doi.org/10.1007/978-3-642-23783-6_28)
32. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
33. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. *Adv. Neural. Inf. Process. Syst.* **31**, 5165–5175 (2018)
34. Corso, G., Cavalleri, L., Beaini, D., Liò, P., Veličković, P.: Principal neighbourhood aggregation for graph nets. In: Advances in Neural Information Processing Systems, vol. 33, pp. 13260–13271 (2020)



# TrajAngleNet: Transformer-Based Trajectory Prediction Through Multi-task Learning with Angle Prediction

Vibha Bharilya<sup>(✉)</sup> and Neetesh Kumar

Indian Institute of Technology, Roorkee 247667, India  
`{vibha_b,neetesh}@cs.iitr.ac.in`

**Abstract.** Accurate trajectory prediction is crucial for the safe and autonomous navigation of self-driving cars. By analyzing sensor data and historical information, it forecasts the movements of surrounding entities. Nevertheless, the accuracy of trajectory prediction models in autonomous systems is significantly compromised when training alone with trajectory supervision, particularly for complex maneuvers such as lane changes and sharp turns. A multi-task trajectory prediction system called TrajAngleNet is presented to address the aforementioned issue. By incorporating angle prediction as an auxiliary task, the model gains the directional context and orientation of agents, thereby improving the accuracy of future trajectory predictions in dynamic environments. Further, we adopt a classification approach to address the challenges posed by the wrap-around behaviour of angles. This method mitigates numerical instability in regression by discretizing angles into bins and focusing on probability distributions rather than exact values. Moreover, the representation of angles in the input using sine and cosine values proves effective in managing periodicity and discontinuities. The TrajAngleNet model has been evaluated on the widely used and recent Argoverse 2 dataset and outperforms state-of-the-art models by a margin of 1.83–35.58%, 1.74–34.68%, 6.25–64.77% on the minADE<sub>1</sub>, minFDE<sub>1</sub>, and MR<sub>6</sub> respectively.

**Keywords:** Trajectory prediction · multi-task learning · angle prediction · transformer · autonomous driving

## 1 Introduction

The domain of trajectory forecasting is rapidly growing, and it plays a pivotal in advancing the safe autonomous driving. Trajectory forecasting deals with predicting future states or occupancy maps for dynamic actors within a local environment [9, 10]. In the context of autonomous driving, relevant actors include vehicles (both stationary and in motion), pedestrians, cyclists, scooters, and pets. The forecasted futures generated by a forecasting system serve as primary inputs for motion planning, which heavily relies on these forecasts for trajectory selection. Trajectory forecasting is a particularly difficult task because of

the different road networks and the inherent multi-modal driving behaviours of agents. Researchers in early studies primarily utilized rasterized semantic images to depict entire scenes from a top-down perspective integrated with convolutional neural networks [23, 29]. Recent advances favour vectorized representations, preserving information integrity, and enabling graph neural networks and transformers for scene encoding [26, 27]. Transformer frameworks are now directly integrated into trajectory prediction tasks, yielding promising results [11]. Hierarchically stacked transformer blocks are also employed to jointly process trajectories and road networks [18, 21].

Multi-task learning enhances task generalization by training multiple related tasks simultaneously, sharing representations, and extracting domain-specific information. Widely applied in computer vision, speech recognition, and data mining, it improves efficiency and performance across diverse domains [15]. For autonomous driving, Jiquan et al. [11] propose a model for multi-task regression predicting motion, conditional motion, and goals. Zirui et al. [12] use a GNN-based framework for multitask learning, integrating three-dimensional bounding box prediction, interactive event identification, and trajectory prediction. Biao et al. [13] introduce an interaction prediction task alongside trajectory prediction to capture traffic agent interactions. Linhui et al. [17] enhance multi-task learning with dynamic adaptive anchors for forecasting multimodal trajectories of heterogeneous agents. However, these models have limited contextual understanding and do not explicitly enhance robustness to variations in heading directions. This could affect the model's performance in scenarios where abrupt directional changes occur, such as during sharp turns or lane changes.

To overcome the above limitation, a multi-task trajectory prediction framework named TrajAngleNet is proposed. This framework incorporates an auxiliary task, angle prediction, within a multi-task learning framework to enhance understanding of agents' direction and orientation for accurate trajectory forecasting. Utilising sine and cosine values for angle representation as input effectively manages angular periodicity and discontinuities in models. Angles as continuous values can suffer from numerical instability due to their bounded range and periodicity. Classification bins the angles into a fixed number of categories, avoiding the instability issues arising from the angles' wrap-around behaviour. In general, the major contributions of this work are as follows:

- This work introduces an auxiliary task, such as angle prediction, within a multi-task learning framework to provide additional context about the direction and orientation of agents. This information is crucial for accurately understanding and predicting their future trajectories.
- Regression can be used to predict angles, but its limited range and periodicity cause numerical instability. To address this, we employ classification by binning the angles into a fixed number of categories and focusing on the probability distribution over bins rather than the exact numerical difference.
- As an angle is wrap-around behaviour, the direction indicated by  $0^\circ$  and  $360^\circ$  (or  $-\pi$  and  $\pi$  in radians) is the same. There may be discontinuities at these boundaries if numerical values are used as input directly. To deal with this,

using sine and cosine values for angle representation in a model effectively handles the periodicity and discontinuities of angular data.

- To measure the efficacy of the proposed model, both qualitative and quantitative analyses were conducted on Argoverse 2. The results demonstrate that our model achieves competitive trajectory prediction accuracy compared to recently proposed approaches.

## 2 Related Work

Inspired by the great success of the transformer design [20], self-attention and cross-attention mechanisms are being widely adopted for interaction modeling and information aggregation due to their exceptional ability and effectiveness. Several studies [19, 21] have directly integrated transformers into forecasting tasks, resulting in promising results. Some studies have modeled traffic elements as graphs [33] and utilized transformer structures to leverage relations in graph-based inputs [22, 26, 27]. Conversely, recent state-of-the-art approaches tend to process all input modalities, including trajectories and road networks, jointly with hierarchically stacked transformer blocks [11, 18, 21]. The FFINet model was suggested by Miao et al. [16] to forecast multi-modal trajectories by extracting future interactions and feeding feasible future interactions of agents back to the observation interaction features. Liu et al. [34] introduced the LAFormer model, which leverages a Transformer-based temporally dense lane-aware module. Chib et al. [32] employed Transformers with diagonally masked self-attention to impute incomplete observations. Nguyen et al. [35] developed guided Transformers for multi-agent trajectory prediction. The proposed model effectively utilizes additional angle information to capture interactions between agents and lanes in the dataset.

Multi-task learning enhances each task's generalization capacity by simultaneously training multiple related tasks. This approach utilises shared representations between tasks and further extracts characteristic domain information from the training signal. It is widely used across various fields, including social networking, computer vision, speech recognition, and data mining [15]. In autonomous driving, Jiquan et al. [11] suggest a method allowing multi-task regression to be performed by a single model that can predict motion, conditional motion, and goals. Zirui et al. [12] present a GNN-based multitask learning framework for interactive behaviour modelling based on the GNN, whereby three-dimensional bounding box prediction, interactive event identification, and trajectory prediction are all with an integrated loss function. In order to capture the interactions between traffic agents, Biao et al. [13] suggested a model that combines multi-task learning with the introduction of an extra interaction prediction task with the trajectory prediction task. Using dynamic adaptive anchors, Linhui et al. [17] create multi-task learning branches that forecast the multimodal trajectories of heterogeneous agents.

Although these studies commonly infer sequential agent trajectory coordinates, they lack a model that incorporates yaw angles to provide additional

directional context of agents in the scene. In dynamic environments, the direction of movement is as important as the position. In this work, by leveraging multi-task learning, the model can benefit from shared representations between trajectory and angle predictions. Trajectory prediction along with angle prediction helps the model better handle complex scenarios where the direction change is significant, such as turns or intersections, improving generalization and robustness across the primary task.

### 3 Problem Definition

Predicting agent trajectories is important for autonomous vehicles, helping them evaluate potential risks and proactively plan suitable trajectories. The positions of agents, involve vehicles, pedestrians, cyclists, scooters, and pets with a history of T time steps are represented as X and  $R_m$  denotes the road map.

$$X = [p_1, p_2, \dots, p_t, \dots, p_T | R_m] \quad (1)$$

whereas,

$$p_t = [(x_t^1, y_t^1, v_t^1, o_t^1), \dots, (x_t^n, y_t^n, v_t^n, o_t^n), \dots, (x_t^N, y_t^N, v_t^N, o_t^N)] \quad (2)$$

$p_t$  represents the state of  $N$ -agents at time  $t$ , including agent's relative coordinates ( $x_t, y_t$ ), velocity difference ( $v_t$ ), and an observation status flag ( $o_t$ ) indicating whether the agent has an observation.

$$Y = [(y^1, c^1), \dots, (y^K, c^K), \dots, (y^F, c^F)] \quad (3)$$

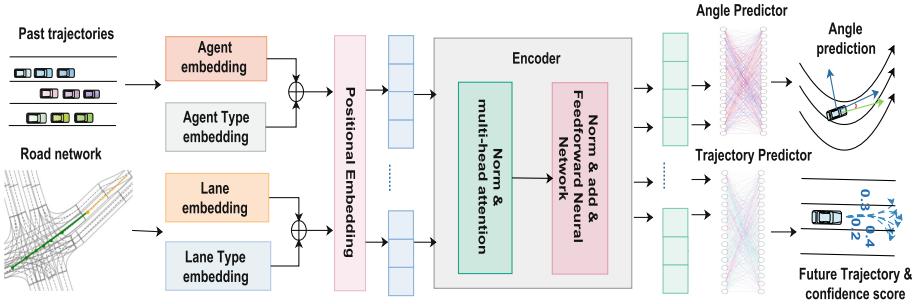
whereas,

$$y^k = [(\hat{x}_{T+1}, \hat{y}_{T+1})^k, \dots, (\hat{x}_{T+t}, \hat{y}_{T+t})^k, \dots, (\hat{x}_{T+F}, \hat{y}_{T+F})^k] \quad (4)$$

denote the predicted coordinates  $(\hat{x}_{T+t}, \hat{y}_{T+t})$  within the future time F. Therefore, the trajectory prediction problem can be summarized as follows: Given the states X of all neighboring agents over a past time horizon T, the goal is to forecast target agent trajectory coordinates Y in the future time horizon F.

### 4 The Proposed TrajAngleNet Model

The proposed architecture of TrajAngleNet is depicted in Fig. 1. First, we define the necessary features for the trajectory prediction task in Sect. 4.1. Next, the embedding layer transforms the sequence into fixed token representations for the transformer inputs, as detailed in Sect. 4.2. The encoder block is defined in Sect. 4.3, and the trajectory prediction module is illustrated in Sect. 4.4. Section 4.5 covers the auxiliary task of angle prediction, and Sect. 4.6 explains the combined loss function used during model training.



**Fig. 1.** The proposed model, TrajAngleNet, takes agent and lane information as input and outputs angle and trajectory predictions.

#### 4.1 Input Representation

Every agent's trajectory and lane segment is represented as a polyline of connected points, following the vectorized representation approach from [27]. Specifically, the past trajectories of  $N$  agents are represented by a tensor  $\mathbf{X}$  with dimensions  $[N, H, C_h]$ , and  $H$  is the number of past frames taken into consideration and  $C_h$  is the number of feature channels in each historical frame. These channels include the agents' coordinates, step-by-step displacement/velocity differences, and a padding flag indicating frame visibility. A local perspective is obtained by transforming the coordinates relative to the target agent, focusing on the environment around the autonomous vehicle [1]. Another input, such as the yaw angle of all agents in the scene, is included alongside their coordinates and velocity. Angles are periodic, meaning they wrap around at  $360^\circ$  or  $2\pi$  radians. Representing angles directly as a single continuous value can cause issues because the model might not understand that  $0^\circ$  and  $360^\circ$  are essentially the same. The transition from  $359^\circ$  to  $0^\circ$  is not smoothly represented on a linear scale. This can be problematic for a neural network that expects input features to vary smoothly. Using sine and cosine to represent angle  $A_{n,t}$  of n-th agent at time t, in two dimensions allows the model to better understand the relationships between different angles. This 2D representation helps maintain the geometric properties of the angle, making it easier for the neural network to learn and generalise.

$$A_{n,t}^{\sin} = \sin(A_{n,t}) \quad (5)$$

$$A_{n,t}^{\cos} = \cos(A_{n,t}) \quad (6)$$

$$A_{\text{concat}} = \text{concat}(A_{n,t}^{\sin}, A_{n,t}^{\cos}) \quad (7)$$

Here,  $A_{n,t}^{\sin}$  and  $A_{n,t}^{\cos}$  denote the sine and cosine values of the angle, respectively.  $A_{\text{concat}}$  represents the concatenation of these values. Additionally, we identify distinct, non-overlapping lane segments  $M$  within the radius around the target agent and represent them using a tensor  $\mathbf{L}$ . This tensor is structured as  $[M, P, C_L]$ , where each lane segment is represented as a series of  $P$  points.

The parameter  $C_L$  denotes the number of feature channels per point or segment, such as coordinates and availability. It is worth noting that all coordinates within each lane polyline are normalized relative to its geometric center, providing a standardized reference point for analysis.

## 4.2 Embedding Layer

The embedding layer transforms sequence features into tokens using a Feature Pyramid Network (FPN) and Neighbourhood Attention Blocks (NATBlock), inspired by LaneGCN [25], to handle multi-scale agent motion. Unlike the standard Transformer, the NAT Block employs 1D neighbourhood attention instead of self-attention, focusing on local motion details. The lane embedding layer adopts PointNet [28] architecture, integrating MLPs and max pooling for a comprehensive road network view. The embedding for agent and lane segment are expressed as:

$$T_h = FPN(X), T_h \in [N, D] \quad (8)$$

$$T_l = MiniPointNet(L), T_l \in [M, D] \quad (9)$$

where,  $D$  represents the embedding dimension, with  $T_h$  for historical features tokens and  $T_l$  for lane token features. Additional attributes like lane types and agent classifications are integrated into tokens using learnable embeddings. Normalized agent coordinates and lane features emphasize embedding tokens with global location information. Positional embedding (PE) is created via a two-layer MLP, detailed in [25], represented as follows:

$$PE = MLP([x, y, \cos(\theta), \sin(\theta)]), PE \in [D] \quad (10)$$

The positional embedding (PE) captures the geometric center pose ( $x, y, \theta$ ) of lane polylines or the latest observed agent posture, enhancing token representations before encoder processing.

## 4.3 Encoder

The encoder block is crucial for extracting and encoding features from lane tokens and road agents, enhancing the model's understanding of their interactions. It follows a standard transformer architecture with Multi-head self-attention mechanism followed by a residual connection and fully connected layers. Past feature tokens ( $T_h$ ) are concatenated with lane tokens ( $T_l$ ), and positional embeddings  $PE(.)$  are added to the input sequence:

$$X_{concat} = \text{concat}(T_h, T_l) + PE \quad (11)$$

The concatenated tokens  $X_{concat}$  are processed by the encoder  $TE(.)$  block with learnable parameters  $W$ :

$$T_E = TE(X_{concat}, MHSA(.), FFN(.), LN(.); W) \quad (12)$$

Here,  $T_E$  is the encoded tokens. The components and functionality of the encoder layer are described in equation (13) and equation (14).

$$X_{input} = LN(MHSA(X_{concat}) + X_{concat}) \quad (13)$$

The Multi-head self-attention  $MHSA(.)$  module computes the relationships between road agents and lane segments. The input  $X_{concat}$  is added to the  $MHSA(.)$  output to retain the original input information and enable more effective training. After applying layer normalization, the training process is stabilized and improved by ensuring that the inputs to subsequent layers have a consistent scale and mean.

$$T_E = LN(FFN(X_{input}) + X_{input}) \quad (14)$$

The Feed-Forward Network  $FFN(.)$  processes the input  $X_{input}$  applying a series of linear transformations and non-linear activation functions to capture and transform features. The  $X_{input}$  is added to the output of the FFN input  $X_{input}$  to retain the original information and aids in effective gradient flow during training. After adding the residual connection, layer normalization is applied to reduce the internal covariate shift, which helps to accelerate the convergence of the model during training.

#### 4.4 Trajectory Predictor

The trajectory predictor utilizes a MultimodalDecoder, which is implemented using a simple MLP architecture. It takes encoded tokens  $T_E$  and processes it through multiple linear layers to predict multimodal outputs.  $Proj(.)$  is the linear layer that projects the input embedding into a space that accounts for k modes, here k = 6.

$$T'_E = Proj(T_E), \quad (15)$$

$Loc(.)$  is sequential layers that predict the future location coordinates (loc) for each mode. It consists of three linear layers with ReLU activations.

$$loc = Loc(T'_E) \quad (16)$$

$Pi(.)$  is sequential layers that predict a scalar uncertainty (pi) for each mode. It also consists of three linear layers with ReLU activations.

$$pi = Pi(T'_E) \quad (17)$$

Both the cross-entropy loss [14] and the widely used Huber loss [24] are assigned equal weights and employed for confidence classification and regression, respectively. Only the best prediction is optimised using a winner-take-all strategy, which minimises the average forecast error in comparison to the ground truth. This loss computation considers every agent in the scene.

$$L_{traj} = L_{reg} + L_{cls} + L_{other} \quad (18)$$

Here,  $L_{traj}$  denotes the total regression loss for trajectory prediction,  $L_{reg}$  represents the regression loss for trajectory prediction, and  $L_{other}$  is the regression loss for all agents present in the scene for a single mode, which minimizes the difference between the predicted positions and the ground truth positions. The  $L_{cls}$  is the classification loss for the best-predicted mode.

#### 4.5 Angle Predictor

Predicting continuous values can be difficult for models to handle, especially when these values have specific constraints or wrap around at certain points (0 to 360). Angles are periodic, and traditional regression methods may not handle this periodicity well. To address this, we convert the angle prediction into a classification problem by dividing the angle range into bins. Binning the angle values between  $-\pi$  and  $\pi$  ensures that the periodic nature is preserved. For each timestamp in the future  $t_f$ , the 2-layer MLP predicts the bin indices  $n_b$  of the angles for every agent present in the scene.

$$\hat{A} = MLP_2(T_E), \hat{A} \in [t_f, n_b] \quad (19)$$

where  $MLP_2$  denotes a multi-layer perceptron (MLP) with 2 layers. By discretizing the angle into bins, the model learns to classify the angle into one of these discrete categories, simplifying the periodicity issue. Cross-entropy loss  $\ell(\hat{A}, A)$  is suitable for classification tasks and effectively helps the model distinguish between different bins. This loss function penalises incorrect predictions more strongly, aiding in better learning. The formulation is described below:

$$L_{angle} = \ell(\hat{A}, A) = - \sum_{n=1}^N \sum_{c=1}^C A_{n,c} \log \left( \frac{\exp(\hat{A}_{n,c})}{\sum_{i=1}^C \exp(\hat{A}_{n,i})} \right) \quad (20)$$

Here,  $L_{angle}$  is the angle loss and  $N$  agents present in the scenes.  $C$  denotes number of bins for the angles.  $A_{n,c}$  is the actual bin indices of angles, which is 1 if agent  $n$  belongs to class  $c$  and 0 otherwise.  $\hat{A}_{n,c}$  is the raw prediction score (logit) for angle bin  $c$  for agent  $n$ .  $\frac{\exp(\hat{A}_{n,c})}{\sum_{i=1}^C \exp(\hat{A}_{n,i})}$  represents the softmax function. It converts the raw prediction scores  $\hat{A}_{n,c}$  (logits) into probabilities for each class  $c$ .  $\sum_{i=1}^C \exp(\hat{A}_{n,i})$  is the sum of exponentials of the scores for all classes for sample  $n$ . This sum acts as a normalization factor, ensuring that the probabilities across all classes sum to 1.

#### 4.6 End-to-End Training

In a multi-task learning setup where we predict both trajectory and angle, the overall loss for the model is a combination of the individual losses for each task. The training process can be outlined as follows:

**Table 1.** Comparisons are made with previous results on the Argoverse 2 test set. Lower values indicate better performance for all metrics. The best results are highlighted in **bold**, and second-best are underline.

Method	minADE <sub>1</sub>	minFDE <sub>1</sub>	MR <sub>1</sub>	minADE <sub>6</sub>	minFDE <sub>6</sub>	MR <sub>6</sub>	b-FDE <sub>6</sub>
THOMAS [30]	1.96	4.71	0.64	0.88	1.51	0.20	2.16
GoReLa [2]	1.82	4.62	0.61	0.76	1.48	0.22	2.01
GANet [3]	<u>1.78</u>	<u>4.48</u>	0.60	0.73	<b>1.35</b>	<b>0.17</b>	1.97
QML w/ ensemble [4]	1.84	4.98	<b>0.59</b>	<b>0.69</b>	1.39	0.19	<u>1.95</u>
BANet w/ ensemble [5]	1.79	4.61	0.60	0.71	<u>1.36</u>	0.19	<b>1.92</b>
BANet w/o ensemble [5]	1.84	4.69	0.615	0.733	1.388	0.179	2.033
HPTR [6]	1.84	4.61	0.61	0.73	1.43	0.19	2.03
SIMPL [7]	2.03	5.50	0.65	0.72	1.43	0.19	2.05
FR [8]	2.37	5.93	0.71	0.89	1.81	0.29	2.47
Forecast-MAE [1]	1.845	4.602	0.623	0.727	1.427	<u>0.187</u>	2.062
TrajAngleNet	<b>1.748</b>	<b>4.403</b>	0.607	<u>0.701</u>	<u>1.361</u>	<b>0.176</b>	2.012

**Table 2.** Comparisons are made with previous results on the Argoverse 2 validation set. Lower values indicate better performance for all metrics. The best results are highlighted in **bold**, and second-best are underline. The “-” indicates that their paper did not report this result.

Method	minADE <sub>1</sub>	minFDE <sub>1</sub>	MR <sub>1</sub>	minADE <sub>6</sub>	minFDE <sub>6</sub>	MR <sub>6</sub>	b-FDE <sub>6</sub>
Forecast-MAE [1]	<u>1.813</u>	<u>4.570</u>	<u>0.622</u>	0.811	<u>1.436</u>	0.189	<u>2.074</u>
SIMPL [7]	-	-	-	<u>0.783</u>	1.452	-	-
TrajAngleNet	<b>1.739</b>	<b>4.375</b>	<b>0.605</b>	<b>0.696</b>	<b>1.366</b>	<b>0.177</b>	<b>2.026</b>

$$L_{total} = L_{traj} + L_{angle} \quad (21)$$

In this case, the overall loss function for training the model is denoted by  $L_{total}(.)$ . The regression loss is represented by  $L_{traj}(.)$ , and the angle loss is represented by  $L_{angle}(.)$ . The combined loss for trajectory and angle prediction allows the model to learn both tasks simultaneously, taking advantage of the shared information between trajectory and orientation. This often leads to better overall performance and more robust predictions, especially in complex tasks such as autonomous driving.

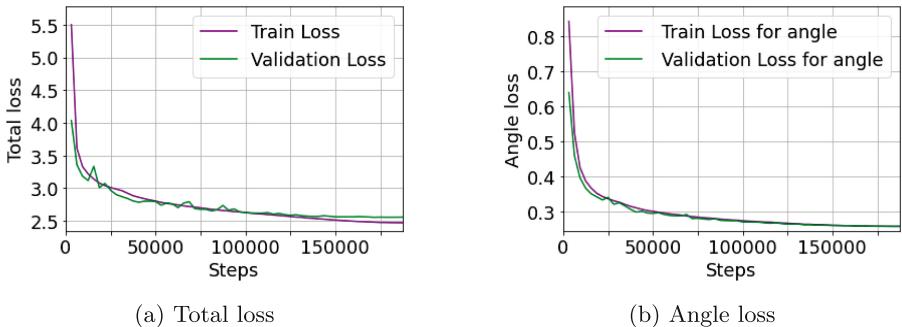
## 5 Experimental Study

### 5.1 Experimental Configuration

**Dataset.** The proposed framework utilises the recently released large Argoverse 2 (AV2) dataset [31]. This dataset contains 250,000 unique scenarios, divided into 199,908 samples for training (approximately 80%), 24,988 for validation (approximately 10%), and 24,984 for testing (approximately 10%). Each sample features a 5-s historical window and requires 6-second future predictions at a 10 Hz frequency. Each Argoverse 2 scenario includes a focal track agent for prediction and high-definition map patches. The dataset’s balance of size and diversity, along with its complexity, makes it an ideal benchmark for evaluating our methods.

**Implementation Details.** Using the PyTorch framework and two Nvidia RTX A5000 GPUs, our model was trained for 60 epochs with a batch size of 64, totaling approximately 14 h. We implemented a warm-up phase of 10 epochs followed by cosine learning rate decay starting at  $1e-3$  and utilizing the  $1e-4$  weight decay AdamW optimizer. Each transformer block featured a dropout rate of 0.2, with experiments incorporating four stacked transformer encoder blocks, each with eight attention heads and latent vector dimensions of 128.

**Metrics.** The model’s performance is assessed using official benchmark metrics, including Brier-minFDE $_k$ , minADE $_k$ , minFDE $_k$ , and miss rate (MR $_k$ ). These metrics evaluate the accuracy of top-ranked predicted trajectory from the  $K$  modes compared to ground truth for a single target agent. Specifically, minFDE $_k$  measures the error at the endpoints, minADE $_k$  calculates the average euclidean distance between the predicted trajectory and the ground truth, and MR $_k$  represents the percentage of sequences where minFDE $_k$  exceeds two meters. The Brier-minFDE $_k$  metric incorporates another Brier score  $(1 - p)^2$ , where  $p$  is the probability of best-predicted trajectory. For more detailed definitions, interested readers can refer to [31].



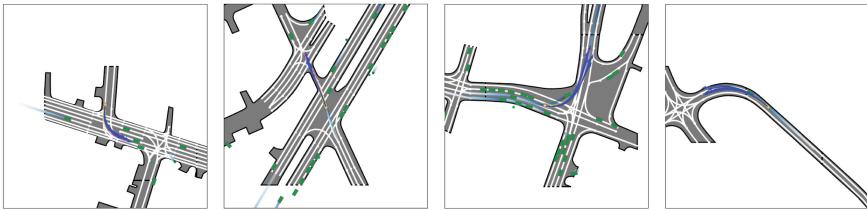
**Fig. 2.** Plots of total loss and angle loss for training and validation

## 5.2 Result

The final leaderboard submission for the argoverse 2 [31] dataset for single agent forecasting uses a model with 36 angle bins, each covering a 10-degree angle. For inference, the model uses 1.9 million parameters for predictions.

**Comparison with State-of-the-Art** The Argoverse 2 [31] leaderboards for Thomas [30], goreala [2], GANet [3], QML [4], BANet [5], HPTR [6], FR [8] and SIMPL [7] are used to benchmark our model against state-of-the-art methods. As shown in Table 1, our model demonstrates remarkable performance on the

leaderboard, despite only using past data points, lane information, and heading angles. In terms of  $\text{minADE}_1$  and  $\text{minFDE}_1$ , our standalone model outperforms the state-of-the-art QML w/ ensemble model by 5% and 11%, respectively. Though the QML w/ ensemble [4] model, which averages predictions from different versions of their models, barely defeated the standalone model TrajAngleNet in terms of  $MR_1$  and  $\text{minADE}_6$ . Every metric of our model outperforms the BANet w/o ensemble model [5]. The results of the model on the Argoverse2 validation dataset are presented in Table 2. Each metric surpasses those of state-of-the-art methods, demonstrating the effectiveness of our approach. Further, Fig. 2 illustrates the training and validation loss curves for total and angle loss, highlighting the model’s learning over time.



**Fig. 3.** Visualization results: The historical trajectory of the agents is displayed in green, ground truth future trajectory in gradient pink, and model’s predictions in blue. The bounding box outlined in orange indicates the target agent. (Color figure online)

**Qualitative Results.** As shown in Fig. 3, we present the qualitative results of our model on the AV2 validation set. The outcomes cover four distinct scenarios: left turns, acute turns, intersection turns, and right turns. The model effectively handles a diverse range of events, including abrupt turns, and other complex maneuvers, facilitated by its heading angle prediction capability. This robustness is essential in dynamic environments like autonomous driving.

### 5.3 Ablation Study

**The Components of Model.** We carried out an ablation analysis to determine the significance of key components in TrajAngleNet on trajectory prediction performance in Table 3. We investigated three configurations: TrajAngleNet without angle loss, TrajAngleNet with angle as regression loss, and TrajAngleNet without the heading angles of agents as input. Comparing these configurations to the full TrajAngleNet model provided valuable insights. The performance metrics are significantly impacted by removing the angle loss module and using angle loss as regression. For  $\text{minFDE}_6$  this leads to a decrease in accuracy of 3.36% and 3.81%, respectively. This suggests that classification effectively manages angular periodicity by discretizing angles into bins, mitigating discontinuity issues at

wrap-around points from  $359^\circ$  to  $0^\circ$ . However, it also indicates that the potential benefits of capturing angular periodicity through regression are not fully utilized in this context. Conversely, removing inputs such as heading angles of agents over previous timestamps and treating angle as a classification problem led to significant performance degradation across all metrics, with a 4.68% decrease in  $\text{minFDE}_6$  accuracy. This highlights the importance of heading angles in understanding an agent’s direction and improving navigation predictions. Effective use of heading angle information and careful loss function design are crucial to TrajAngleNet’s trajectory prediction performance, allowing it to leverage both the direction and path of motion.

**Table 3.** Components of the model result. **Bold** text indicates the best entry for each statistic.

Method	minADE <sub>1</sub>	minADE <sub>6</sub>	minFDE <sub>1</sub>	minFDE <sub>6</sub>	MR <sub>1</sub>	MR <sub>6</sub>	b-FDE <sub>6</sub>
TrajAngleNet w/o angle loss	1.795	0.718	4.489	1.412	0.612	0.183	2.040
TrajAngleNet w/ angle loss as regression	1.793	0.718	4.502	1.418	0.618	0.187	2.045
TrajAngleNet w/o angle as input	1.797	0.723	4.508	1.430	0.619	0.185	2.060
TrajAngleNet (our)	<b>1.739</b>	<b>0.696</b>	<b>4.375</b>	<b>1.366</b>	<b>0.605</b>	<b>0.177</b>	<b>2.026</b>

**Table 4.** Different bin number results. **Bold** text indicates the best entry for each statistic.

Number of Bin	minADE <sub>1</sub>	minADE <sub>6</sub>	minFDE <sub>1</sub>	minFDE <sub>6</sub>	MR <sub>1</sub>	MR <sub>6</sub>	b-FDE <sub>6</sub>
9	1.791	0.720	4.483	1.425	0.620	0.184	2.052
18	1.757	0.715	4.398	1.414	0.612	0.181	2.044
36	<b>1.739</b>	<b>0.696</b>	<b>4.375</b>	<b>1.366</b>	<b>0.605</b>	<b>0.177</b>	<b>2.026</b>
72	1.786	0.722	4.472	1.435	0.622	0.187	2.064

**Hyperparameter Bin.** To evaluate the sensitivity of trajectory prediction performance to varying the number of bins, we conducted an ablation study summarized in Table 4. The metrics reflect changes in predictive accuracy based on different angle bin configurations. Using 9 bins for metrics resulted in poorer performance compared to using 36 bins. The minADE<sub>1</sub> decreased by 2.990% and minADE<sub>6</sub> by 3.44% when using 9 bins. Increasing the number of bins to 18 improved performance relative to 9 bins but still showed a decrease compared to using 36 bins, with minADE<sub>1</sub> decreasing by 1.03% and minADE<sub>6</sub> decreasing by 2.73%. This indicates that if the number of bins is too low, the model may underfit the data, failing to capture important variations in angle that are necessary for accurate trajectory prediction. When employing 36 bins, which

offered finer granularity, the model demonstrated the best performance across all metrics. Employing 72 bins demonstrates minADE<sub>1</sub> decreased by 2.7027%, and the minADE<sub>6</sub> decreased by about 3.73%. More bins provide finer granularity in representing different angle ranges but lead to overfitting. The choice of 36 bins for the TrajAngleNet model strikes an optimal balance between granularity and generalization in trajectory prediction.

## 6 Conclusion and Future Work

This study introduces a novel approach to improve trajectory prediction through multi-task learning with angle prediction. By incorporating angle prediction, this framework improves the model's ability to understand directional context and agent orientation, enhancing trajectory prediction accuracy in dynamic scenarios. We address the challenge of angle periodicity using a classification approach, discretizing angles into bins to mitigate regression issues related to angle wrap-around. This method focuses on angle bin probabilities, ensuring robust performance in trajectory prediction tasks. Leveraging sine and cosine representations further aids in handling angular data's periodic nature, facilitating smooth transitions across angle boundaries for accurate predictions. Experimental validation on the Argoverse 2 dataset demonstrates competitive accuracy against state-of-the-art methods. Future research will explore additional auxiliary tasks and advanced multi-task learning strategies to further enhance model adaptability and robustness in varied real-world environments.

## References

- Cheng, J., Mei, X., Liu, M.: Forecast-MAE: self-supervised pre-training for motion forecasting with masked autoencoders. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 8679–8689 (2023)
- Cui, A., Casas, S., Wong, K., Suo, S., Urtasun, R.: GoRela: go relative for viewpoint-invariant motion forecasting. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 7801–7807. IEEE (2023)
- Wang, M., et al.: GANet: goal area network for motion forecasting. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 1609–1615. IEEE (2023)
- Su, T., Wang, X., Yang, X.: QML for Argoverse 2 motion forecasting challenge. arXiv preprint [arXiv:2207.06553](https://arxiv.org/abs/2207.06553) (2022)
- Zhang, C., Sun, H., Chen, C., Guo, Y.: BANet: motion forecasting with boundary aware network. arXiv preprint [arXiv:2206.07934](https://arxiv.org/abs/2206.07934) (2022)
- Zhang, Z., Liniger, A., Sakaridis, C., Yu, F., Gool, L.V.: Real-time motion prediction via heterogeneous polyline transformer with relative pose encoding. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
- Zhang, L., Li, P., Liu, S., Shen, S.: SIMPL: a simple and efficient multi-agent motion prediction baseline for autonomous driving. IEEE Robot. Autom. Lett. **9**, 3767–3774 (2024)

8. Park, D., Ryu, H., Yang, Y., Cho, J., Kim, J., Yoon, K.J.: Leveraging future relationship reasoning for vehicle trajectory prediction. arXiv preprint [arXiv:2305.14715](https://arxiv.org/abs/2305.14715) (2023)
9. Bharilya, V., Kumar, N.: Machine learning for autonomous vehicle's trajectory prediction: a comprehensive survey, challenges, and future research directions. *Veh. Commun.* **46**, 100733 (2024)
10. Bharilya, V., Kumar, N.: A survey of the state-of-the-art reinforcement learning-based techniques for autonomous vehicle trajectory prediction. In: 2023 International Conference on Electrical, Electronics, Communication and Computers (ELEXCOM), pp. 1–6. IEEE (2023)
11. Ngiam, J., et al.: Scene transformer: a unified multi-task model for behavior prediction and planning. arXiv preprint [arXiv:2106.08417](https://arxiv.org/abs/2106.08417), vol. 2, no. 7 (2021)
12. Li, Z., Gong, J., Chao, L., Yi, Y.: Interactive behavior prediction for heterogeneous traffic participants in the urban road: a graph-neural-network-based multi-task learning framework. *IEEE/ASME Trans. Mechatron.* **26**(3), 1339–1349 (2021)
13. Yang, B., Fan, F., Ni, R., Wang, H., Jafaripournimchahi, A., Hu, H.: A multi-task learning network with a collision-aware graph transformer for traffic-agents trajectory prediction. *IEEE Trans. Intell. Transp. Syst.* **25**, 6677–6690 (2024)
14. Mao, A., Mohri, M., Zhong, Y.: Cross-entropy loss functions: theoretical analysis and applications. In: International Conference on Machine Learning, pp. 23803–23828. PMLR (2023)
15. Zhang, Yu., Yang, Q.: A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.* **34**(12), 5586–5609 (2021)
16. Kang, M., Wang, S., Zhou, S., Ye, K., Jiang, J., Zheng, N.: FFINet: future feedback interaction network for motion forecasting. *IEEE Trans. Intell. Transp. Syst.* **25**, 12285–12296 (2024)
17. Li, L., Wang, X., Yang, D., Ju, Y., Zhang, Z., Lian, J.: Real-time heterogeneous road-agents trajectory prediction using hierarchical convolutional networks and multi-task learning. *IEEE Trans. Intelligent Veh.* **9**, 4055–4069 (2023)
18. Nayakanti, N., Al-Rfou, R., Zhou, A., Goel, K., Refaat, K.S., Sapp, B.: Wayformer: motion forecasting via simple & efficient attention networks. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 2980–2987. IEEE (2023)
19. Zhou, Y., Wang, Z., Ning, N., Jin, Z., Lu, N., Shen, X.: I2T: from intention decoupling to vehicular trajectory prediction based on priorformer networks. *IEEE Trans. Intell. Transp. Syst.* **25**, 9411–9426 (2024)
20. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
21. Zhou, Z., Wang, J., Li, Y.H., Huang, Y.K.: Query-centric trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17863–17873 (2023)
22. Jia, X., Wu, P., Chen, L., Liu, Y., Li, H., Yan, J.: HDGT: heterogeneous driving graph transformer for multi-agent trajectory prediction via scene encoding. *IEEE Trans. Pattern Anal. Mach. Intell.* **11**, 13860–13875 (2023)
23. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M. et al.: CoverNet: multimodal behavior prediction using trajectory sets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14074–14083 (2020)
24. Tong, H.: Functional linear regression with Huber loss. *J. Complex.* **74**, 101696 (2023)

25. Liang, M., et al.: Learning lane graph representations for motion forecasting. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12347, pp. 541–556. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-58536-5\\_32](https://doi.org/10.1007/978-3-030-58536-5_32)
26. Wang, Z., Zhang, J., Chen, J., Zhang, H.: Spatio-temporal context graph transformer design for map-free multi-agent trajectory prediction. IEEE Trans. Intell. Veh. **9**, 1369–1381 (2023)
27. Jiao, Y., et al.: A hierarchical hybrid learning framework for multi-agent trajectory prediction. IEEE Trans. Intell. Transp. Syst. **25**, 10344–10354 (2024)
28. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
29. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint [arXiv:1910.05449](https://arxiv.org/abs/1910.05449) (2019)
30. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Thomas: trajectory heatmap output with learned multi-agent sampling. arXiv preprint [arXiv:2110.06607](https://arxiv.org/abs/2110.06607) (2021)
31. Wilson, B., et al.: Argoverse 2: next generation datasets for self-driving perception and forecasting. arXiv preprint [arXiv:2301.00493](https://arxiv.org/abs/2301.00493) (2023)
32. Chib, P.S., Nath, A., Kabra, P., Gupta, I., Singh, P.: MS-TIP: imputation aware pedestrian trajectory prediction. In: International Conference on Machine Learning, pp. 8389–8402. PMLR (2024)
33. Sachan, A., Chauhan, N.S., Kumar, N.: Congestion minimization using fog-deployed DRL-agent feedback enabled traffic light cooperative framework. In: 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 557–567. IEEE (2023)
34. Liu, M., et al.: LAformer: trajectory prediction for autonomous driving with lane-aware scene constraints. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2039–2049 (2024)
35. Nguyen, N.L., Yoo, M.: Multi-agent trajectory prediction with adaptive perception-guided transformers. IET Intell. Transp. Syst. **18**, 1196–1209 (2024)



# Correlation Disentangling and Spatio-Temporal Cooperative Optimizing Network for Temperature Prediction Revision

Aoao Wei, Xitie Zhang, Suping Wu<sup>(✉)</sup>, Shaohua Yang, Junfeng Zhao,  
and Kehua Ma

Ningxia University, Ningxia 750000, China  
[pswuu@nxu.edu.cn](mailto:pswuu@nxu.edu.cn)

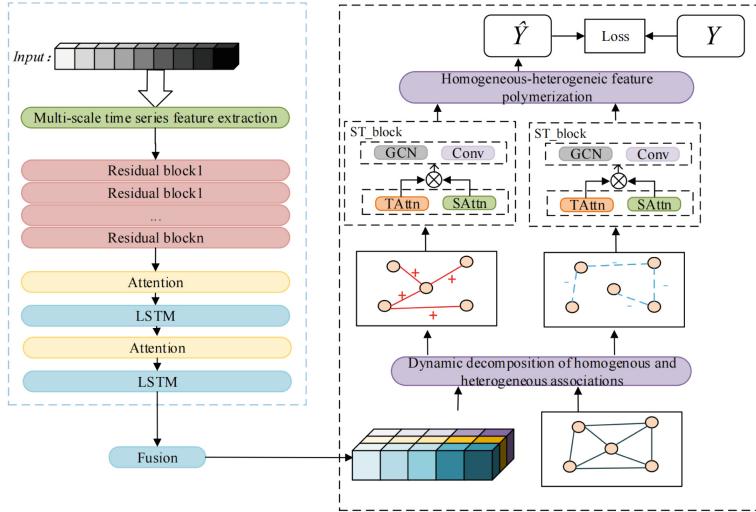
**Abstract.** Temperature is one of the most critical meteorological elements in people's daily lives and production, significantly impacting agriculture, transportation, and other sectors. Error correction in multisite temperature predictions is crucial for the accuracy of temperature forecasting. While existing methods for temperature prediction revision have made progress, they still face challenges related to uneven and imprecise multisite predictions, and a failure to fully and effectively explore the correlation between sites. To address these issues, we propose a correlation disentangling and spatio-temporal cooperative optimizing network(CSTGCN). Specifically, by utilizing temporal cues, we decompose the homogeneous heterogeneous correlations of the input data to process different types of inter-variable relationships across multiple data sites. Analyze and explore the overall correlations among multisites in both temporal and spatial dimensions through the spatiotemporal attention mechanism, which process enhances the impact of crucial temporal and spatial nodes to stabilize the overall prediction accuracy. And mine the temporal and spatial local relationships among multisite data through spatiotemporal map convolution, further deepening the model's comprehension of geographic information and temporal correlation and enhancing the alignment of predictions with actual climate change trends. Extensive experimental results demonstrate the effectiveness of our approach compared to existing methods.

**Keywords:** Temperature forecast error correction · Graph Convolutional Networks · Deep learning

## 1 Introduction

Temperature is one of the most critical meteorological elements affecting people's daily lives and production [2]. It has a significant impact on agricultural production, etc. [12, 15]. Accurate temperature forecasting applications and revision

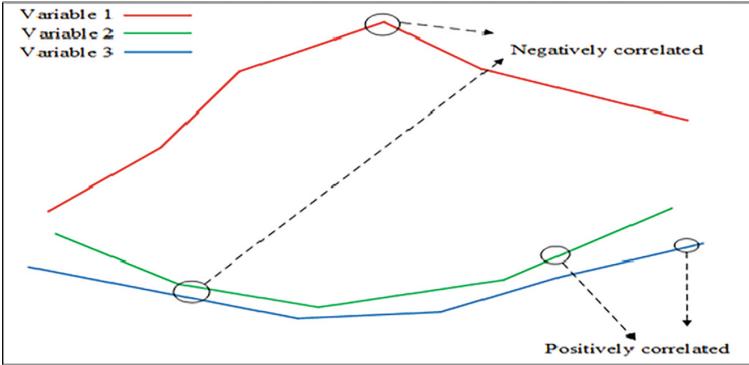
techniques are widely used in agricultural production [25]. However, the complexity and nonlinearity of the atmospheric system, along with the variability of surface conditions, pose numerous challenges to weather forecasting. Therefore, error correction in temperature forecasts has emerged as a key strategy to improve forecast accuracy. The development of numerical weather prediction techniques marks a shift from relying on intuition and traditional methods to modernization based on scientific principles and computational techniques [16]. The parameterization of physical processes and the optimization of the model by Kuang et al. [11] have enabled numerical prediction models to better simulate complex atmospheric processes and weather systems. However, such numerical weather prediction methods often have significant systematic biases. In recent years, with the rapid development of machine learning and deep learning technologies, especially network structures such as convolutional neural networks (CNN) [6, 18], recurrent neural networks (RNN) [4, 18], and U-Net [19, 21], which have shown powerful capabilities in image processing and sequence data processing, significant progress has been made in weather forecasting and correct [1, 8, 10]. Lin et al. [14] established a spatio-temporal temperature deviation prediction model based on convolution and long and short-term memory networks while using the precipitation forecast in NWP to construct a precipitation forecast dataset, which realized the improvement in the accuracy of temperature forecast in NWP and proved that the addition of precipitation elements played a positive role in improving the model prediction accuracy. The BP neural network achieved better results in interpreting numerical forecasts of nearshore seawater temperature. Still, it has the problem of too many training parameters and high computational cost, which affects the prediction efficiency [3]. DLWP-CS [23] used 2 m surface air temperature as input to predict daily precipitation, outperforming linear regression and global prediction systems. The CNN-based architecture [21] aimed to predict extreme heat waves from surface temperature and potential height. ATMConvGRU [28] used an axial attention memory module designed in a quasi state-in-state cascaded manner, facilitating the aggregation and embedding of features within a standard ConvGRU framework. This method enhanced the extraction of spatial-temporal features with stronger correlations and improves the accuracy of weather forecasts. Han et al. [7] revised the predicted temperatures, 2 m relative humidity, as well as 10 m wind speed and wind direction, resulting in improvements in each evaluation metric. Wang [24] sequentially stacked 20 GCMs with single or multiple input/output channels through the SRDRN model, which effectively removes bias based on the relative relationship between different GCMs and observations, and can retain inter-variable dependencies for multivariate bias correction. Shen [22] used a 5-layer structured deep belief network (DBN) to better capture the complex and nonlinear relationships between temperature and different predictor variables, which essentially ensured robust prediction of the spatio-temporal distribution of temperature through a layer-by-layer pre-training process of feature extraction and a fine-tuning process of weight parameter optimization. Le [13] proposed a method based on a convolutional neural network and autoen-



**Fig. 1.** Workflow of the spatio-temporal cooperative optimization network.

coder architecture(ConvAE) to correct pixel-by-pixel satellite productions for bias. The ConvAE model showed superior and stable performance in most comparisons, demonstrating the potential of this model in solving the precipitation bias correction problem. Ying [27] utilized a U-Net model and residual connection model for numerical weather prediction and forecast correction. Maharatha [17] demonstrated exceptional performance in predicting minimum and maximum temperatures using a stacking encoder-decoder architecture with bidirectional LSTM units. Nketiah [20] developed a multivariate recurrent neural network (RNN)-based time series model for atmospheric temperature prediction, using RNN for modeling and Ridge Regularizer (L2) to regulate the training process of the neural network. Additionally, they utilized a Bayesian optimization method to fine-tune hyperparameters, such as the number of network nodes, regularization parameters, and batch size, thereby improving prediction performance.

Although these methods above have achieved some results, still suffer from the problem of uneven and inaccurate multisite predictions, and a failure to fully and effectively explore the correlation between sites. To address the aforementioned issues, we propose a spatio-temporal co-optimization network. Specifically, Fig. (1) illustrates the workflow of our proposed CSTGCN. Input data is extracted by  $n$  residual blocks, attention mechanism, and LSTM for feature extraction. ST\_block denotes the spatio-temporal block, where graph convolutional networks (GCNs) and convolutional neural networks (CNNs) are used for feature extraction. The GCNs process graph-structured data, and through the GCN layers, the intricate features of temperature distribution among various geographic locations and the complex spatial relationships among nodes. On



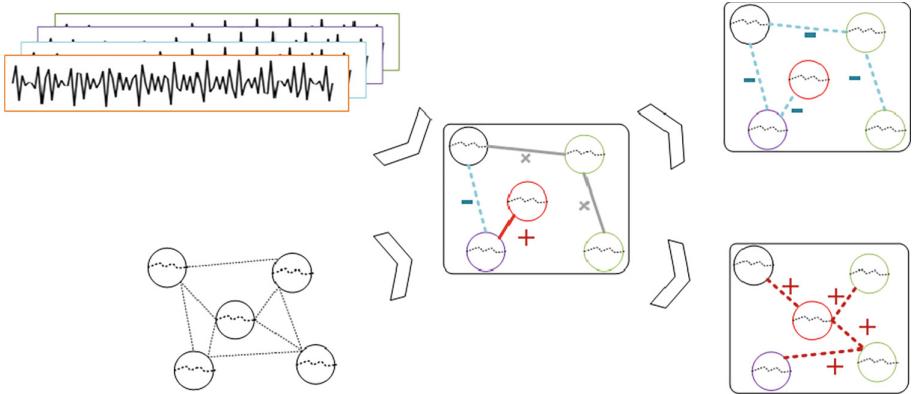
**Fig. 2.** Plot of homogeneous and heterogeneous relationships between variables.

the other hand, the CNN layers extract features within a localized region. By integrating spatial and temporal attention mechanisms, ST\\_block can simultaneously consider complex dependencies on time series and spatial distributions, as well as identify crucial features in different time periods to optimize the accuracy of temperature forecast error revision. Our core contributions to this work are summarized as follows:

- We designed the spatio-temporal co-optimization network to effectively capture the dynamic spatio-temporal correlations in temperature forecast data.
- We design the homogeneous heterogeneous correlation decomposition module, the spatio-temporal attention module, and the spatio-temporal map convolution module, respectively, to analyze and mine the global and local connections from both temporal and spatial dimensions, improving the overall forecast accuracy.
- Extensive experimental results on the dataset show that our method outperforms the comparative methods.

## 2 Method

In this section, we present our methods accordingly, including the homogeneous heterogeneous correlation decomposition module, the spatio-temporal attention module, the spatio-temporal map convolution module, and loss functions for our tasks. As shown in Fig. (1), input data is extracted by  $n$  residual blocks, attention mechanism, and LSTM for feature extraction. Subsequently, the global and local connections are analyzed and mined from both temporal and spatial dimensions by graph convolution and hybrid time and space modules(ST\\_block). Additionally, the homogeneous-heterogeneous correlation decomposition module is combined to guide the model to learn the spatio-temporal distribution law of temperature and improve the accuracy of temperature prediction.



**Fig. 3.** The detailed structure of homogeneous heterogeneous correlation decomposition module.

## 2.1 The Homogeneous Heterogeneous Correlation Decomposition Module

**Homogeneous and Heterogeneous Relationships Between Variables.** In real-world data analysis, interactions between variables exhibit both consistency and variability. These relationships can stabilize over time, as shown in Fig. (2). For the revision of temperature forecast errors across multiple stations or regions, understanding the consistency and discrepancy between these variables is critical to improving the accuracy of the forecasts [9]. Specifically, consistency refers to the fact that the variables show similar patterns or dynamics at different sites; discrepancy reflects the different or idiosyncratic behaviour of the variables across sites. By learning and recognizing these relationships, predictive models can better capture invariant associations between variables and thus be able to make more accurate predictions based on identified patterns and relationships when processing data from new sites. Therefore, we propose a homogeneous-heterogeneous correlation decomposition module to extract consistency and differences in variable relationships across sites, with a view to achieving more accurate predictions in complex and variable environments.

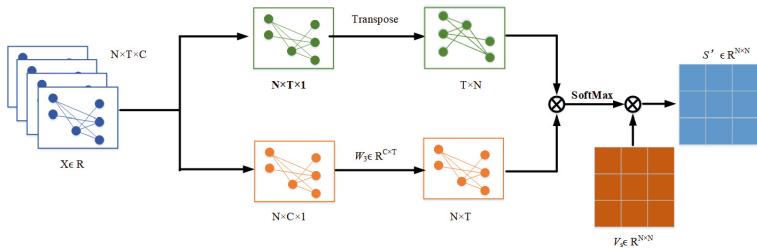
**Homogeneous Heterogeneous Correlation Decomposition Module.** The homogeneous-heterogeneous correlation decomposition module aims to decompose the correlation structure between the extracted temporal feature  $F$  and its neighborhood graph structure into two components: homogeneous and heterogeneous neighborhood graphs, as shown in Fig. (3).  $F$  is the multivariate feature tensor representing the features extracted from  $n$  residual blocks, attention mechanism, and LSTM,  $m$  is the number of nodes (number of sites),  $d$  is the feature dimension, and  $n$  is the sequence length. The overall process of the module can be described as:

$$Z = \text{ReLU}(W_f \cdot [A, F] + b_f) \quad (1)$$

$$A^+ = \text{Sigmoid}(W^+ \cdot Z + b^+) \quad (2)$$

$$A^- = \text{Sigmoid}(W^- \cdot Z + b^-) \quad (3)$$

where  $W_f$ ,  $W^+$ , and  $W^-$  are the weight matrices of the neural network,  $b_f$ ,  $b^+$ , and  $b^-$  are the bias vectors,  $[A, F]$  denotes the splicing operation of the adjacency graph  $A$  and the feature tensor  $F$ ,  $\text{ReLU}$  denotes the modified linear unit activation function,  $\text{Sigmoid}$  denotes the activation function. The decomposed homogeneous  $A^+$  and heterogeneous neighbor-joining graphs  $A^-$ , and the multivariate feature tensor  $F$ , respectively, are input to the homogeneous and heterogeneous modules for spatial feature extraction, which utilize neighbor-joining graphs to capture spatial dependencies between nodes.



**Fig. 4.** The detailed structure of the spatio-temporal attention module.

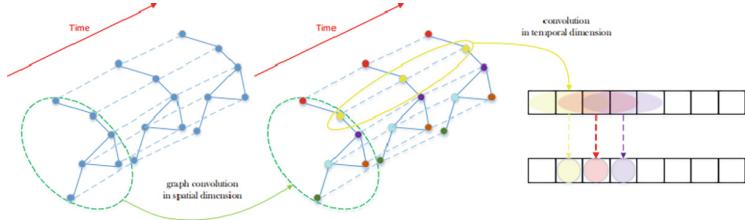
## 2.2 The Spatio-Temporal Attention Module

To accurately capture the dynamics and long-term dependencies of time-series data and model the correlations between different regions, we propose a spatio-temporal attention module. This module encourages the model to focus on the correlation of temperature changes between different stations, leading to more accurate revisions in multisite temperature forecasts. It consists of spatial attention and temporal attention. In the spatial dimension, the temperature changes between sites are interconnected, and this connection is highly dynamic. Therefore, we adaptively capture the dynamic correlation between nodes in the spatial dimension through the spatial attention mechanism, as shown in Fig. (4). The dynamic correlation between individual nodes at a specific time node is described as:

$$\mathbf{S} = \mathbf{V}_s \cdot \sigma((\mathcal{X}_h^{(c-I)} \mathbf{W}_I) \mathbf{W}_2 (\mathbf{W}_3 \mathcal{X}_h^{(c-I)})^T + \mathbf{b}_s) \quad (4)$$

$$\mathbf{S}'_{i,j} = \frac{e \times p(\mathbf{S}_{i,j})}{\sum_{j=1}^N e \times p(\mathbf{S}_{i,j})} \quad (5)$$

Among them, the input data of the  $h$ th block are  $\chi_h^{(c-1)} = (X_1, X_2, \dots, X_{T_-}) \in R^{N \times K_{c-1} \times T_{r-1}}$ ,  $V_e \in R^{T \times T}$ ,  $W_1 \in R^N$ ,  $W_2 \in R^C$ ,  $W_3 \in$



**Fig. 5.** The detailed structure of the spatio-temporal graph convolution module.

$R^{C \times N}$ , and  $b_e \in R^{T \times T}$  are learnable parameters. For the  $c$  layer, the channel size of the input data is  $K_{c-1}$ . Sigmoid  $\sigma$  is used to dynamically compute the attention matrix  $S$  based on the current layer's inputs. Subsequently, Softmax is used to obtain the normalized attention weights of all nodes.

The temperature of a single site has a strong correlation with the temporal dimension, which dynamically changes over time. Through the time-attention mechanism, the influence weights of different time nodes on the current node can be dynamically adjusted. The specific process can be described as follows:

$$\mathbf{E} = \mathbf{V}_e \cdot \sigma(((\mathcal{X}_h^{(c-1)})^T W_1) W_2 (W_3 \mathcal{X}_h^{(c-1)}) + \mathbf{b}_e) \quad (6)$$

$$\mathbf{E}'_{i,j} = \frac{\exp(\mathbf{E}_{i,j})}{\sum_{j=1}^{T_r=1} \exp(\mathbf{E}_{i,j})} \quad (7)$$

where  $V_e \in R^{T \times T}$ ,  $W_1 \in R^N$ ,  $W_2 \in R^C$ ,  $W_3 \in R^{C \times N}$ , and  $b_e \in R^{T \times T}$  are the learning parameters,  $\sigma$  is the activation function,  $\mathbf{E}$  and  $\mathbf{E}_{ij}^{prime}$  are the unstandardized attention matrix and the *SoftMax* function standardized attention matrix respectively,  $\mathbf{E}_{ij}^{prime}$  and  $\sigma$  are the activation function.  $\mathbf{E}'_{ij}$  are the un-normalized attention matrix and the normalized attention matrix of the *SoftMax* function, respectively, and the  $\mathbf{E}_{i,i}$  values denote the dependence weights of time step  $j$  on time step  $i$ .

### 2.3 The Spatio-Temporal Graph Convolution Module

Temperature data from weather stations at different geographical locations show significant spatial correlations. However, traditional models have difficulty capturing such spatial relationships. By treating these stations as nodes in the graph, the spatio-temporal graph convolution module can naturally model the complex spatial relationships between the stations, thereby improving the accuracy of temperature prediction. Additionally, the module can accurately model complex spatio-temporal dependencies in meteorological data by effectively integrating spatio-temporal features. Its structure is shown in Fig. (5). The spatio-temporal convolution module proposed in this paper consists of graph convolution in the spatial dimension, which captures spatial dependencies from the neighborhood, and convolution in the temporal dimension, which captures temporal dependencies from neighboring time.

In multisite analysis, the relationship between sites essentially constitutes a network graph, where the data carried by each node can be regarded as signals on the network graph. To dig deeper into the structural features of the site network, this paper applies a method based on spectral graph theory for graph signal processing in each period. This approach explores the signal correlations among sites at the spatial level through graph convolution operations. Spectral domain graph convolution can transform the network graph into an algebraic form to explore the network structure's properties, including the connectivity between nodes.

In the spectral domain of graph convolution, the Laplace matrix and its eigenvalue analysis are often used to represent the structural properties of graphs. Where  $L = D - A$  is the definition of the Laplace matrix, the normalized form is  $L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ ,  $A$  denotes the adjacency matrix,  $I_N$  denotes the unitary matrix, and the matrix  $D$  refers to the diagonal matrix of the degree of nodes in the graph. By eigenvalue decomposition of the Laplace matrix, i.e.  $L = U\Lambda U^T$ , where  $\Lambda = \text{diag}([\lambda_0, \dots, \lambda_{N-1}^-]) \in R^{N \times N}$  is the diagonal matrix, and  $U$  represents the Fourier basis. At t time point, the signal on the whole graph is  $x = x_t^f \in R^N$ , the graph Fourier transform of this signal is defined as  $\hat{x} = U^T x$ , according to the property of the Laplace matrix,  $U$  is an orthogonal matrix, so the corresponding Fourier inverse transform is  $x = U\hat{x}$ . Graph Convolution By this operation, the conventional convolution operator is replaced by a linear operator diagonalized in the Fourier domain, which applies the kernel function  $g_\theta$  to the signal  $x$  on the graph  $G$  to filter.

$$g_\theta *_G x = g_\theta(\mathbf{L})x = g_\theta(\mathbf{U}\Lambda\mathbf{U}^T)x = \mathbf{U}g_\theta(\Lambda)\mathbf{U}^T x \quad (8)$$

In this case, the graph convolution operation is denoted as  $*_G$ . The process of graph signal convolution involves multiplying the signals in the frequency domain by the graph Fourier transform and then performing the Fourier inverse transform. Therefore, the final result of the convolution operation can be obtained by transforming the Fourier transforms of  $g_\theta$  and  $x$  to the frequency domain and then performing the Fourier inverse transform of their products in the frequency domain. This method first transforms the two signals to the frequency domain, then performs the multiplication operation in this domain, and finally converts the product result back to the original domain through the inverse transformation to complete the convolution process.

After capturing the neighborhood information around each node through graph convolution operations, a regular convolution layer in the time dimension is used, which is responsible for integrating information from consecutive time points to update the signals of different nodes, as shown on the right side of Fig. (5). The operation of layer  $r$  in the nearest component is also used as an example.

$$\mathcal{X}_h^{(r)} = \text{ReLU}(\Phi * (\text{ReLU}(g_\theta *_G x))) \in \mathbb{R}^{C_r \times N \times T_r} \quad (9)$$

where  $*$  is the convolution operation,  $\Phi$  denotes the convolution kernel parameter, and the activation function is ReLU.

## 2.4 Objective Function

We aim to minimize the difference between the predicted temperature value  $f(x)$  and the actual target value  $y$  by using a Mean Squared Error (MSE) loss function. It is shown in the following equation:

$$\text{Loss\_mse} = \frac{\sum_{i=1}^n (f(x) - y)^2}{n} \quad (10)$$

where  $n$  is the number of samples,  $f(x)$  is the model revision result, and  $y$  is the truth value.

## 3 Experiments

In this section, we evaluate the performance of our proposed method on a temperature revision task. We first present the dataset used in our experiments. We then compare our results quantitatively and qualitatively with several other methods.

### 3.1 Datasets

ECMWF (European Centre for Medium-Range Weather Forecasts) forecast data has become increasingly valuable in the fields of deep learning and machine learning, particularly in meteorology and climate prediction. The data used in this study are derived from ECMWF’s fine-grid temperature forecasts for a region, with a horizontal resolution of  $0.5^\circ \times 0.5^\circ$  and a 3-hourly frequency, covering the period from January 1 to December 31, 2019, with forecast time ranging from 12 to 36 h. Station observation data, selected from 13 stations within the region, are used for the current experiment for the period of January 1 to December 31, 2019, to match the model products. The observation hours were 24 times a day, with 1-hourly positive observations, and the observed elements included canopy air temperature, humidity, and air temperature, among others. Characteristic cloudiness and humidity were removed through characteristic correlation analysis. The four features with significant correlations to the observation data—canopy temperature, dew point temperature, 50 cm ground temperature, and 50 cm water content—were selected for modeling. This paper employs ECMWF’s fine-grid temperature forecast data, processed through inverse distance weight interpolation to a horizontal resolution of  $0.1^\circ \times 0.1^\circ$  and a 1-hourly period, to make it consistent with the observed data.

### 3.2 Experimental Details and Evaluation Metrics

We train and test using Python 3.6 and PyTorch on an NVIDIA GTX 2080Ti GPU. The batch size during training was 128, and the Adam optimizer was used for optimization. The initial learning rate was 0.001, with 300 iterations and a decrease of 0.1 after every 100 iterations. The dataset was divided into three

parts: 80% were used as training data, 10% were used for validation, and the remaining 10% were used for testing. The same metrics as in previous methods were used to evaluate the model performance using three metrics:  $RMSE$ (Root Mean Square Error),  $MAE$ (Mean Absolute Error), and  $R^2$ (coefficient of determination).  $RMSE$  denotes the square root of the mean of the squares of the differences between predicted and actual values.  $MAE$  refers to the average of the absolute value of the difference between the predicted value and the actual value.  $R^2$  reflects the degree of correlation between the predicted and actual values of the model. The closer the value of  $R^2$  is to 1, the stronger the model's predictive ability.

### 3.3 Results and Analysis

**Quantitative Evaluation.** To validate the effectiveness of the proposed method, we evaluate our method with  $RMSE$ ,  $MAE$ , and  $R^2$  as evaluation metrics on the dataset based on ECMWF with three prediction times of 12 h, 24 h, and 36 h, respectively. As shown in Table 1, compared to traditional methods such as Support Vector Regression (SVR) and Autoregressive Integrated Moving Average (ARIMA) and several commonly used time series prediction methods, including Recurrent Neural Networks (RNN), Temporal Convolutional Networks (TCN), Long Short-Term Memory networks (LSTM), and Gated Recurrent Units (GRU), the graph convolutional networks (GCNs) demonstrate lower errors in three metrics. However, in cases  $MAE$  and  $R^2$ , STGCN [5] and ASTGCN [26] do not show significant improvements compared to GRU. In contrast, the advantages of our method are more pronounced. Our method shows significant improvements in all three indicators, indicating that our CSTGCN provides highly accurate predictions for time series data.

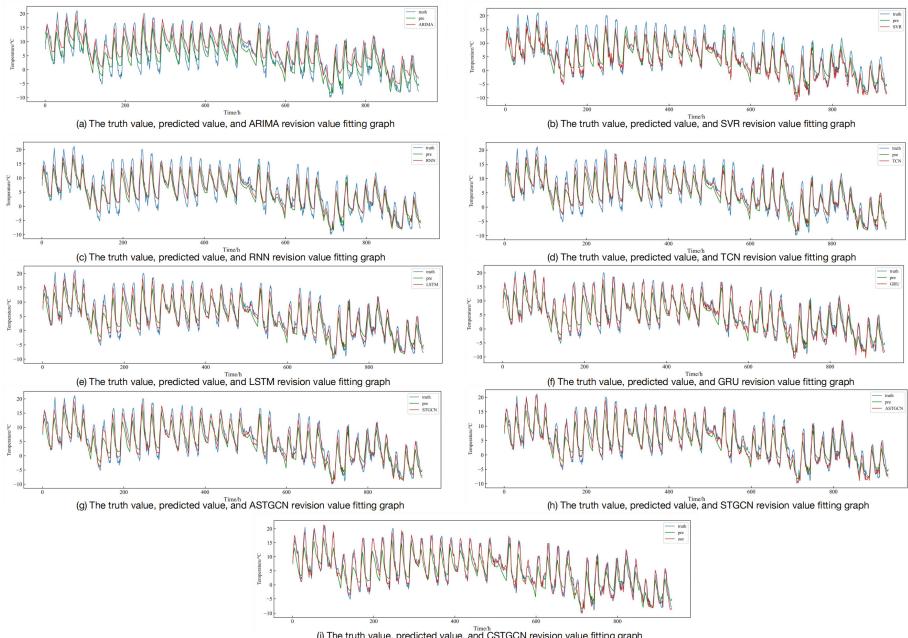
**Table 1.** 12 h, 24 h, 36 h forecasting timescale prediction results of each model evaluation indicators.

Model	12 h			24 h			36 h		
	$RMSE$	$MAE$	$R^2$	$RMSE$	$MAE$	$R^2$	$RMSE$	$MAE$	$R^2$
SVR	2.42	1.83	0.875	2.40	1.87	0.871	2.43	1.86	0.873
ARIMA	3.65	2.94	0.734	3.66	2.95	0.732	3.67	2.94	0.733
RNN	2.40	1.97	0.899	2.42	1.98	0.901	2.44	1.99	0.899
LSTM	1.86	1.37	0.951	1.88	1.38	0.950	1.91	1.37	0.952
TCN	1.93	1.55	0.922	1.91	1.53	0.923	1.95	1.53	0.920
GRU	1.61	1.23	0.952	1.63	1.25	0.953	1.65	1.23	0.953
STGCN	1.55	1.15	0.951	1.52	1.14	0.948	1.54	1.19	0.950
ASTGCN	1.54	1.17	0.954	1.55	1.18	0.955	1.53	1.15	0.954
CSTGCN	<b>1.41</b>	<b>1.01</b>	<b>0.961</b>	<b>1.40</b>	<b>1.01</b>	<b>0.963</b>	<b>1.43</b>	<b>1.03</b>	<b>0.964</b>

**Table 2.** Predicted results of  $RMSE$  evaluation metrics at different step sizes.

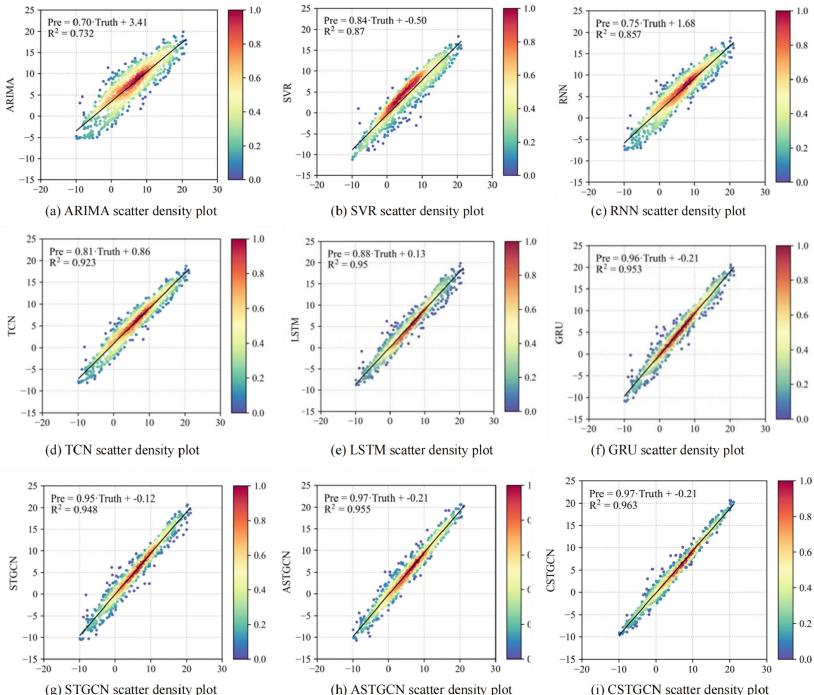
Model	Steps				
	1	2	3	4	5
SVR	2.40	2.51	2.64	2.82	2.97
ARIMA	3.66	3.85	3.97	4.11	4.32
RNN	2.42	2.61	2.82	2.88	3.05
LSTM	1.88	1.89	1.93	1.95	2.13
TCN	1.91	2.03	2.08	2.19	2.28
GRU	1.63	1.64	1.73	1.81	1.89
STGCN	1.52	1.67	1.73	1.79	1.85
ASTGCN	1.55	1.69	1.77	1.86	1.89
CSTGCN	<b>1.40</b>	<b>1.45</b>	<b>1.53</b>	<b>1.62</b>	<b>1.75</b>

In addition, we use  $RMSE$  as an evaluation metric to compare the prediction accuracy of target series with step sizes ranging from 1 to 5. As shown in Table 2, compared with the STGCN model, the  $RMSE$  metrics of our method have a reduction in the  $RMSE$  metrics, indicating a decrease in prediction errors within 1 to 5 prediction steps, respectively. Furthermore, our method consistently

**Fig. 6.** Results of existing methods in temperature revision

delivers the most accurate prediction results as the prediction step increases. The experimental results demonstrate that our method effectively considers the complex dependencies in time series and spatial distributions, while dynamically capturing key features in different time periods, thus realizing the improvement in the accuracy of the revision of the temperature prediction errors.

**Qualitative Experiments.** To demonstrate the effectiveness of our method more intuitively, we make relevant comparisons of the forecast visualization results. As shown in Fig. (6), the red trend line in the line graph represents the corrected forecast value, the blue trend line represents the actual observation data, and the green trend line represents the original temperature forecast data. The traditional methods ARIMA and SVR, along with common recurrent neural networks such as RNN, TCN, LSTM, and GRU, exhibit significant prediction errors. Among these, the ASTGCN model outperforms the previous models, while our proposed CSTGCN model outperforms all the models in terms of prediction effectiveness and fits better with the highest value of the temperature forecast. In addition, we use scatter density plots to analyze the effects of different models on temperature forecasts, including ARIMA, SVR, RNN, TCN, LSTM, GRU, CSTGCN, STGCN, and our CSTGCN. Figure (7) shows the dis-



**Fig. 7.** Scatter density plots for different models.

tribution of the revised data of temperature forecast errors for the nine models. It is evident that the revised temperatures from the STGCN and ASTGCN models still significantly deviate from the observation data at some points, while the revised results of our method are closer to the observation data in comparison, further indicating that the forecasts of our method are more accurate.

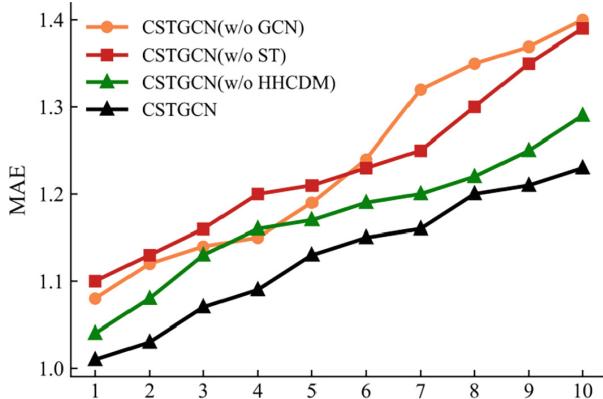
### 3.4 Ablation Study

**Ablation Study of the Modules.** To verify the effectiveness of our proposed method, we investigated the ablation of the spatio-temporal attention module, the spatio-temporal graph convolution, and the homogeneous-heterogeneous correlation decomposition module. With other structures of CSTGCN unchanged, CSTGCN (w/o ST) removes the spatio-temporal attention module, resulting in a considerable increase in all metrics. This demonstrates that the spatio-temporal attention module accurately captures the dynamics of time-series data, long-term dependencies, and correlations between different regions. CSTGCN (w/o GCN) removes the spatio-temporal graph convolution module, leading to an increase in all metrics, indicating that this module effectively captures the structural features of the site network. CSTGCN (w/o HHCDM) removes the homogeneous-heterogeneous correlation decomposition module, and all the metrics also increase, which is because this module can decouple the homogeneous and heterogeneous features, which is more conducive to the model's learning of the temporal features. CSTGCN is the case where all the modules are present, and the model's effectiveness reaches optimization, proving that our proposed module is very effective for the temperature revision task (Table 3).

**Table 3.** The effectiveness of our methods.

Method	RMSE	MAE	R <sup>2</sup>
CSTGCN(w/o ST)	1.48	1.10	0.954
CSTGCN(w/o GCN)	1.47	1.08	0.953
CSTGCN(w/o HHCDM)	1.41	1.04	0.960
CSTGCN	<b>1.40</b>	<b>1.01</b>	<b>0.963</b>

**Ablation Study on the Performance of Different Prediction Time Steps.** In addition, we conduct an ablation study on the performance of different structural models over different prediction time steps, as shown in Fig. (8), with MAE (Mean Absolute Error) as the metric. The horizontal axis represents the prediction time steps from 1 to 10, while the vertical axis represents the MAE values. The red line CSTGCN (w/o ST) indicates the removal of the temporal attention module. In the early stage (time steps 1–4), there is a slight decrease in performance compared to the black folded line of CSTGCN with



**Fig. 8.** Comparison of  $MAE$  values of CSTGCN with different structures.

all modules included. However, in the mid and late stages (time steps 5–10), the decrease is more significant, indicating that the spatiotemporal attention module is particularly effective in long-term prediction. The orange line CSTGCN(w/o GCN) represents the CSTGCN without the spatio-temporal graph convolution module. It performs slightly better than CSTGCN (w/o ST) across all prediction time steps but is still worse than the full CSTGCN, indicating the significant role of the spatio-temporal graph convolution module in enhancing prediction performance. The green line CSTGCN (w/o HHCDM) denotes the CSTGCN with the homogeneous-heterogeneous correlation decomposition module removed. Its performance is relatively close to the full CSTGCN model, especially for short-time steps (1–4), and is nearly equivalent to CSTGCN. However, as the prediction time step increases, a decreasing trend in its performance emerges, underscoring the effectiveness of the homogeneous-heterogeneous correlation decomposition module. All the above experiments prove the effectiveness of our proposed method.

## 4 Conclusion

In this paper, we propose a correlation disentangling and spatio-temporal cooperative optimizing network for temperature prediction revision, which comprises a homogeneous-heterogeneous correlation decomposition module, a spatio-temporal attention module, and a spatio-temporal graph convolution module. The homogeneous-heterogeneous correlation decomposition module can analyze and process different types of inter-variable relationships at the data site, enhancing the model's capability to handle complex data. The spatio-temporal attention mechanism enables the forecast model to more accurately identify and utilize temporal and spatial patterns in temperature change, improving the overall forecast accuracy by reinforcing the effects of important temporal and spatial nodes. Lastly, the spatio-temporal map convolution module can further enhance

the model's comprehension of geographic information and time series, making the forecasts closer to the actual climate change trends. Our proposed network can significantly improve the accuracy and reliability of temperature predictions.

**Acknowledgments.** This work was supported by Ningxia Key Research and Development Program Project under Grant 2022BBF02014.

## References

1. Ahmed, K., Sachindra, D., Shahid, S., Iqbal, Z., Nawaz, N., Khan, N.: Multi-model ensemble predictions of precipitation and temperature using machine learning algorithms. *Atmos. Res.* **236**, 104806 (2020)
2. Bi, K., Xie, L., Zhang, H., Chen, X., Gu, X., Tian, Q.: Accurate medium-range global weather forecasting with 3D neural networks. *Nature* **619**(7970), 533–538 (2023)
3. Chen, J., et al.: Predict the effect of meteorological factors on haze using BP neural network. *Urban Climate* **51**, 101630 (2023)
4. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint [arXiv:1406.1078](https://arxiv.org/abs/1406.1078) (2014)
5. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: International Conference on Machine Learning, pp. 933–941. PMLR (2017)
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
7. Han, L., et al.: A deep learning method for bias correction of ECMWF 24–240 h forecasts. *Adv. Atmos. Sci.* **38**(9), 1444–1459 (2021)
8. Han, Y., et al.: A short-term wind speed prediction method utilizing novel hybrid deep learning algorithms to correct numerical weather forecasting. *Appl. Energy* **312**, 118777 (2022)
9. Huang, Q., et al.: CrossGNN: confronting noisy multivariate time series via cross interaction refinement. *Adv. Neural. Inf. Process. Syst.* **36**, 46885–46902 (2023)
10. Jose, D.M., Vincent, A.M., Dwarakish, G.S.: Improving multiple model ensemble predictions of daily precipitation and temperature through machine learning techniques. *Sci. Rep.* **12**(1), 4678 (2022)
11. Kuang, X.D., Wang, Z.Y., Zhang, M.Y., et al.: An interpretation scheme of numerical near-shore sea-water temperature forecast based on BPNN. *Oceanologia et Limnologia Sin.* **47**(06), 1107–1115 (2016)
12. Lam, R., et al.: Learning skillful medium-range global weather forecasting. *Science* **382**(6677), 1416–1421 (2023)
13. Le, X.H., Lee, G., Jung, K., An, H.u., Lee, S., Jung, Y.: Application of convolutional neural network for spatiotemporal bias correction of daily satellite-based precipitation. *Remote Sens.* **12**(17), 2731 (2020)
14. Lin, H., Hua, Y., Ma, L., Chen, L.: Application of ConvLSTM network in numerical temperature prediction interpretation. In: Proceedings of the 2019 11th International Conference on Machine Learning and Computing, pp. 109–113 (2019)
15. Liu, B., et al.: Assessing forecasting performance of daily reference evapotranspiration using public weather forecast and numerical weather prediction. *J. Hydrol.* **590**, 125547 (2020)

16. Lynch, P.: The origins of computer weather prediction and climate modeling. *J. Comput. Phys.* **227**(7), 3431–3444 (2008)
17. Maharatha, A., Das, R., Mishra, J., Nayak, S.R., Aluvala, S.: Employing sequence-to-sequence stacked LSTM autoencoder architecture to forecast Indian weather. *Procedia Comput. Sci.* **235**, 2258–2268 (2024)
18. Mahmud, M., Kaiser, M.S., McGinnity, T.M., Hussain, A.: Deep learning in mining biological data. *Cogn. Comput.* **13**(1), 1–33 (2021)
19. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. *IEEE Trans. Neural Netw. Learn. Syst.* **29**(6), 2063–2079 (2018)
20. Nketiah, E.A., Chenlong, L., Yingchuan, J., Aram, S.A.: Recurrent neural network modeling of multivariate time series and its application in temperature forecasting. *PLoS ONE* **18**(5), e0285713 (2023)
21. 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](https://doi.org/10.1007/978-3-319-24574-4_28)
22. Shen, H., Jiang, Y., Li, T., Cheng, Q., Zeng, C., Zhang, L.: Deep learning-based air temperature mapping by fusing remote sensing, station, simulation and socioeconomic data. *Remote Sens. Environ.* **240**, 111692 (2020)
23. Singh, M., et al.: A modified deep learning weather prediction using cubed sphere for global precipitation. *Front. Climate* **4**, 1022624 (2023)
24. Wang, F., Tian, D.: On deep learning-based bias correction and downscaling of multiple climate models simulations. *Clim. Dyn.* **59**(11), 3451–3468 (2022)
25. Wu, H., Zhou, H., Long, M., Wang, J.: Interpretable weather forecasting for worldwide stations with a unified deep model. *Nat. Mach. Intell.* **5**(6), 602–611 (2023)
26. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph WaveNet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121* (2019)
27. Yingying, H., Lin, P., Qiguang, W.: Application of deep learning bias correction method to temperature grid forecast of 7–15 days. *J. Appl. Meteorol. Sci.* **34**(4), 426–437 (2023)
28. Yu, T., Kuang, Q., Yang, R.: ATMConvGRU for weather forecasting. *IEEE Geosci. Remote Sens. Lett.* **19**, 1–5 (2021)



# Hierarchical Adaptive Position Encoding-Based Transformer for Point Cloud Analysis

Jiawei Yao<sup>1,2</sup>, Junfeng Yao<sup>1,2,3(✉)</sup>, Yong Yang<sup>2</sup>, and Chunyang Huang<sup>2</sup>

<sup>1</sup> Institute of Artificial Intelligence, Xiamen University, Xiamen 361005, China  
yao0010@xmu.edu.cn

<sup>2</sup> Center for Digital Media Computing, School of Film, School of Informatics, Xiamen University, Xiamen 361005, China

<sup>3</sup> Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, Xiamen, China

**Abstract.** In recent years, transformers have significantly advanced 3D point cloud analysis by capturing long-range dependencies. Current methods focus on complex local feature processing, leading to substantial neighbor preparation and position embedding overhead, and fail to model remote context dependencies directly. Additionally, learned position embeddings struggle to integrate point cloud spatial information with deep features, impairing effective feature learning and target perception. We propose HAPE-Net, which excels in capturing remote context and demonstrates strong generalization and performance. Our method utilizes a Global Position Attention (GPA) module with Hierarchical Adaptive Position Encoding (HAPE), combined with an efficient Local Squeeze Aggregation (LSA) operator for local feature extraction, capturing shape context within and between point sets. HAPE adapts to network depth, promoting spatial and semantic information fusion. LSA and GPA effectively model short-range and long-range spaces, respectively. Our approach requires no additional operations like local position embedding or neighbor preparation. Experiments on ScanObjectNN, ShapeNet-Part, and S3DIS demonstrate HAPE-Net’s effectiveness in point cloud classification and segmentation, surpassing previous methods.

**Keywords:** Transformer · Position embedding · Point cloud analysis

## 1 Introduction

Three-dimensional point cloud data has demonstrated enormous potential for applications in fields such as autonomous vehicles, machine intelligence, and augmented reality. Unlike the ordered array of pixels in 2D images, 3D point clouds consist of unordered sets of discrete points, posing challenges for 3D data processing using convolutional neural networks. To effectively address this issue,

researchers have proposed various advanced methods for processing 3D point cloud data.

Some methods aim to transform irregular point cloud data into ordered structures, such as projecting into multi-view images [2,5] or voxelization [18,37], while others design convolutional operators directly in the point cloud space [13,27,32]. The advent of the self-attention mechanism has provided a new solution for modeling long-range dependencies [29]. Thanks to its self-attention capability, the Transformer can directly capture the correlation between any two elements [7], thus achieving great success in sequence data tasks such as natural language processing.

For 3D point cloud data, some exploratory works have attempted to introduce self-attention in a local or global manner [6,19,33,36], yet there is still a distance from truly solving the core issues of long-range dependencies and scalability. Moreover, it's noteworthy that in 2D images, spatially regular pixels can easily form patch-wise tokens, but 3D points are entirely different with irregular point arrangements, presenting significant challenges to the design of 3D Transformers [12]. In point cloud transformers, individual points are treated as tokens, and researchers typically apply point embedding techniques to every point during local aggregation to supplement crucial geometric prior information for feature extraction, leading to faster convergence and stronger performance during training.

Position embedding is crucial for transformers to capture the sequential order of input tokens, its efficacy has been proven in natural language processing [9,31]. Recently, some researchers have conducted more detailed studies on the position embedding methods used in traditional point cloud processing networks [14]. However, its efficacy in point cloud transformers has not been fully explored. Restricted by its original design, existing position embedding methods cannot well integrate spatial relations into the feature extraction process, instead bringing additional local processing overhead. Overall, how to efficiently and effectively integrate positional information and better apply Transformers to 3D point cloud data remains an important issue to be addressed.

To this end, we developed a simple yet powerful framework for point cloud analysis, HAPE-Net. Specifically, we optimized the learned point cloud positional representations with an improved position embedding method, making the learned spatial relationship features more expressive, and combined it with global cross-attention to better capture shape context. Additionally, we designed LSA module to quickly complete local aggregation operations, enhancing the local detail extraction capability while maintaining the network's computational efficiency.

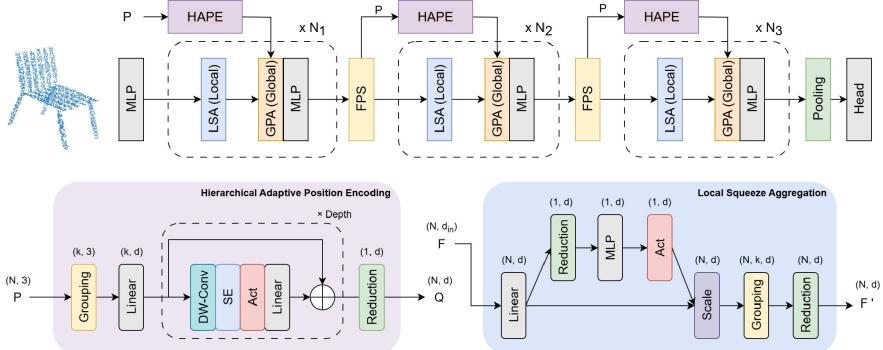
Our contributions are as follows:

- (1) We designed an effective Hierarchical Adaptive Position Encoding (HAPE) from a global perspective to better promote the learning and fusion of spatial information in attention-based point cloud feature extraction modules.
- (2) We proposed a Global Position Attention module using global contextual position embedding as queries, effectively integrating point cloud positional

information. Additionally, we introduced the LSA module to rapidly aggregate essential local information, ensuring a balance between performance and computational efficiency.

- (3) Experiments demonstrate that our model achieves advanced results on widely used point cloud analysis datasets (e.g., ScanObjectNN [28], ShapeNetPart [34] and S3DIS [1]). Extensive ablation studies validate the benefits of each component.

## 2 Methods



**Fig. 1.** Framework overview of HAPE-Net. We employ a hierarchical structure for multi-level feature extraction. At each stage, the point cloud undergoes downsampling, and position encoding is applied based on the current coordinates.

### 2.1 Architecture

Figure 1 illustrates the architecture of HAPE-Net. We propose a hierarchical framework that reduces the total number of processing points through farthest point sampling (FPS) while expanding the embedding dimensions to obtain different stages of feature representations. Initially, a set of input point clouds  $\mathbf{x} \in \mathbb{R}^{N \times D}$  is projected into a C-dimensional embedding space by using  $1 \times 1$  convolution layers. The spatial coordinates  $\mathbf{p} \in \mathbb{R}^{N \times 3}$  at various stages will be used to generate global positional queries.

For classification tasks, we employ several consecutive HAPE blocks, after which we apply max pooling and feed the result into a multi-layer perceptron. In the case of segmentation tasks, we upsample the multi-scale features outputted at different stages to the original number of points and then sum them up, before finally using a multi-layer perceptron to complete the segmentation.

Each stage of HAPE-Net consists of alternating Local Squeeze Aggregation Modules (LSA) and Global Position Attention Modules (GPA), designed for progressively extracting features within and between point sets. Below, we will explain all the modules in this method in detail.

## 2.2 Hierarchical Adaptive Position Encoding

For transformers, capturing the sequential order of input tokens through position embedding is crucial, a general efficacy already proven in natural language processing. Compared to 2D spatially regular pixels, 3D points reside in a more complex continuous space, posing challenges to leveraging spatial positions [12]. Researchers such as [6] believe that since spatial coordinates are already fed into the network at the outset, additional positional embedding seems unnecessary. However, subsequent research by PointMetaBase [14] has systematically demonstrated the efficacy and necessity of positional embedding through empirical evidence. PT [36] employs relative position embedding within each point’s neighborhood, achieving better results but with considerable redundant computation. Stratified Transformer [12] uses a learnable lookup table for relative position embedding, but as features ascend in dimensionality, a clear mismatch emerges between the position embedding and the current feature’s semantic level, potentially resulting in the loss of fine-grained positional information during integration with high-level features as the network deepens.

As the network deepens and the semantic level of features increases, the representational capability obtained through traditional position embedding methods (e.g., trigonometric functions [35] or multi-layer perceptrons [4]) becomes inadequate compared to the keys and values generated by higher-level semantic features. This mismatch can potentially lead to poor integration of spatial information and loss of high-level semantic concepts. Considering that the global structural representation is only provided by a set of position embeddings, the representational capability of position embedding must be sufficiently comprehensive and compatible with the keys and values of the cross-attention module. To ensure that the embedding of spatial information adapts to these changes and prevents conflicts between low-level and high-level features, we propose a Hierarchical Adaptive Position Encoding module as shown in Eq. 1.

$$\gamma_i^s = \mathcal{R}_{j:(i,j) \in \mathcal{N}} \{ \Theta_s (f\{\mathbf{p}_j^s - \mathbf{p}_i^s\}) \} \quad (1)$$

Here,  $p$  represents the point coordinates,  $\gamma$  denotes the positional encoding of point, and the subscript  $s$  indicates the current network stage. It consists of an initial linear layer  $f$ , several cascaded fusion projection blocks  $\Theta_s$ , and a final feature reduction layer  $\mathcal{R}$  (such as max-pooling) that captures spatial information from the neighborhood of point  $i$ . The design of the fusion projection block  $\Theta$  is inspired by the concept of spatial feature contraction in CNN models [11, 26], which impose locality bias and cross-channel interactions on the shape context of the point cloud. The structure of a single fusion projection block  $\Theta$  is as shown in Eq. 2, with its stack number matching the depth corresponding to the stage. For example, for a network with a block depth of  $[n_1, n_2, n_3]$ , the number of Fusion Projection Blocks used in the HAPE would be [1, 2, 3].

$$\begin{aligned}
\hat{x} &= \text{DW-Conv}_{1 \times 1}(x), \\
\hat{x} &= \text{SE}(\hat{x}), \\
\hat{x} &= \text{Act}(\hat{x}), \\
x &= \text{Conv}_{1 \times 1}(\hat{x}) + x,
\end{aligned} \tag{2}$$

The position embedding based on shallow MLPs projects the relative coordinates  $\mathbf{p} \in \mathbb{R}^{N \times K \times 3}$  directly into positional biases  $\mathbf{p}_e \in \mathbb{R}^{N \times K \times D}$ . In contrast, HAPE adaptively generates position embeddings of different semantic levels by stacking improved fusion modules, thus providing spatial semantic information more compatible with the features of the current stage. This facilitates their subsequent cross and fusion. We also compared our method with previous position embedding techniques such as sinusoidal embedding and shallow MLPs. For detailed results, please see Sect. 3.4. In addition, we found that using pre-computed HAPE as global positional queries performs nearly as well as learning positional queries directly from the input features, and offering advantages like reduced parameter count and computational load.

### 2.3 Global Position Based Attention

Features within a point set play a vital role in detecting intricate details in point cloud analysis tasks, while obtaining advanced semantic information among point sets is also crucial. However, the integration of higher-level features is more intricate than that of lower-level feature aggregation. Modeling advanced features requires more precise representations. We propose an efficient global attention mechanism, termed Global Position Relation-based Attention (GPA), to better embed spatial relation priors into the feature extraction process, as illustrated in Algorithm 1. Here, GPA uses positional embedding as queries, with the output features from the Local Aggregation Module (LSA) serving as keys and values to calculate attention weights. The positional embedding is provided by the HAPE module, offering overall shape context to the attention module.

At each stage, the Hierarchical Adaptive Position Encoding (HAPE) component is pre-computed once to serve as a global positional query. GPA leverages these pre-extracted global positional query tokens, interacting with the output features of the Local Squeeze Aggregation (LSA) block as keys and values. This alternating approach of local aggregation and global attention blocks effectively captures spatial information both within and between point sets. The size of the global position query  $q_{pos}$  is  $B \times N \times D$ , where  $B$ ,  $N$ , and  $D$  respectively denote the batch size, number of points, and dimensionality.  $q_{pos}$  is further reshaped into multiple heads to participate in the computation of multi-head attention.

### 2.4 Local Squeeze Aggregation

The local aggregation process generally includes the collection, update, and reduction of neighborhood features. However, directly applying MLPs or self-

**Algorithm 1.** Pseudocode of GPA in a PyTorch-like style

---

```

# Input/output shape: (B, N, D)
# B: Number of batches; N: Number of points;
# D: Dimension size; H: Number of Attention Heads;
def init():
    f = nn.Linear(D, D * 2)
    softmax = nn.Softmax(dim=-1)
    scale = (D // H) ** -0.5

def forward(x, q_pos):
    B, N, D = x.shape
    kv = f(x).reshape(B, N, 2, H, D // H)
    kv = kv.permute(2, 0, 3, 1, 4)
    k, v = kv.unbind(0)
    q_pos = q_pos.reshape(B, H, N, D // H)
    q_k = (q_pos - k).view(B, N, D)
    q_k = mlp(q_k).reshape(B, H, N, D // H)
    attn = softmax(q_k * scale)
    x = (attn * v).reshape(B, N, D)
    return x

```

---

attention to update features for each point’s neighborhood incurs significant computational overhead, making it nearly infeasible. Experimental results from Flatformer [15] also demonstrate that performing various operations on features after neighborhood collection is a primary cause of slowing down model computational speed.

ASSANet [22] proved that without including the relative coordinates term, PreConv set abstraction is equivalent to vanilla set abstraction, suggesting that swapping the order of MLP and grouping operations in local aggregation could reduce the computational burden by a factor of  $k$  (where  $k$  is the number of neighbors). Building on this, we designed a new local aggregation module that captures connections within point sets both accurately and efficiently. While a simplistic token mixer structure utilizing only neighborhood pooling operations may enhance efficiency, it can limit the model’s expressive capability. Therefore, we implemented a Squeeze-and-Excitation-like structure, inspired by convolutional neural networks [8], which introduces channel attention after the linear layer to enhance feature connections within point sets. This design choice aims to improve model performance by effectively utilizing the global position information provided by HAPE while maintaining computational efficiency, compensating for the fact that HAPE does not directly learn the relationships between internal features of point sets.

For a given point cloud feature map  $F = \{x_1, x_2, \dots, x_N\} \in \mathbb{R}^{N \times d}$ , our method is illustrated in the equations below:

$$f_i = Conv(x_i) \quad (3)$$

$$Attn_c(f_i) = \text{Sigmoid}(MLP(P_{avg}(f_i))) \quad (4)$$

$$f'_i = Attn_c(f_i) \odot f_i \quad (5)$$

$$x'_i = P_{max}\{knn(f'_i)\} \quad (6)$$

where  $Conv$  represents a convolutional layer with a  $1 \times 1$  kernel size, used for initial feature recombination of the input features.  $P_{avg}(\cdot)$  denotes an adaptive average pooling function.  $MLP$  consists of two linear layers with a ReLU activation function inserted in between, aimed at reducing and then restoring the original channel dimensions. Finally, the features  $f_i$ , after being weighted by channel attention, undergo neighbor grouping and apply max pooling to extract key local features of point  $x'_i$ .

### 3 Experiments

In this section, we outline the purpose of our experiments, which is to evaluate the effectiveness of our approach against the latest advanced point cloud analysis networks. We designed the experiments to include a variety of challenging datasets, ensuring a comprehensive assessment of our method's capabilities. Experimental outcomes indicate that our approach outperforms existing methods across all tested datasets, highlighting its effectiveness in point cloud analysis.

#### 3.1 Datasets

**Shape Classification.** We validate HAPE-Net on the dataset ScanObjectNN [28]. ScanObjectNN consists of 2,902 real-world objects divided into 15 categories. Existing point cloud analysis methods face significant challenges on ScanObjectNN due to occlusions and noise disturbances. Among various versions of ScanObjectNN, we use the most challenging perturbed version, namely PB\_T50\_RS. We use the training and evaluation split from [28]. Point resampling augmentation was applied on ScanObjectNN following [23].

**Part Segmentation.** For part segmentation task, we conduct experiments on ShapeNetPart [34], a synthetic dataset that contains 16,881 shapes across 16 categories with 50 part labels. We use the same split as [20], where 14,006 samples are used for training and 2,874 samples for validation. Following the approach of networks such as PointNext [23], we employ a voting scheme in the ShapeNetPart task, where we average the results of 10 randomly scaled input point clouds using scaling factors between [0.8, 1.2].

**Scene Segmentation.** To compare with previous methods for scene segmentation, we validate HAPE-Net using the widely utilized benchmark dataset S3DIS [1]. S3DIS is a demanding benchmark comprising six expansive indoor spaces, 271 individual rooms, and a total of 13 semantic categories. In our experiments, we primarily follow the experimental configuration outlined in PointNext [23], utilizing Area-5 for evaluation purposes.

### 3.2 Implementation Details

Our method was trained using the PyTorch framework on an Nvidia RTX 3090. During the training process, we employed data augmentation techniques, including height appending, random scaling and feature dropout (applied to color or normal vector). We choose a base learning rate of 5e-4 and a batch size of 64, implementing a cosine annealing strategy for learning rate decay. The epochs was set to 250, and all models were trained using the AdamW optimizer and cross-entropy loss with label smoothing [25]. Weight decay was set to 0.15, with normalization and activation functions set to BatchNorm [10] and ReLU. For the experimental results, we take the average of five random runs.

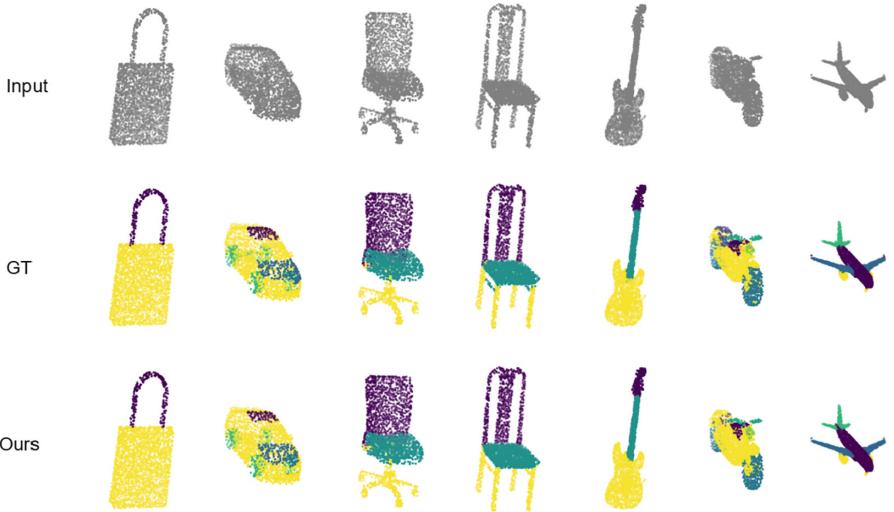
We set the initial embedding dimension and the dimension of attention heads to 64 and 32, respectively, with feature dimensions doubling at each stage. Specifically, for ScanObjectNN, we randomly sampled 1,024 points for both training and evaluation. The network was configured with three stages, constructed with a block depth of [2,2,2]. For ShapeNetPart, we randomly sampled 2,048 points on each shape. Given the larger number of processing points, the network was configured with four stages, constructed with a block depth of [2,2,2,2]. For the S3DIS, we downsample point clouds using voxels with a size of 0.04 m, in line with standard procedures [22,27]. Due to the increased number of processing points, the network was designed with four stages, constructed with a block depth of [2,2,4,2].

### 3.3 Results

Table 1 shows that on ScanObjectNN, HAPE-Net outperformed all baselines. *OA* was improved by up to 1.7% compared to PointNeXt, and compared to SPoTr, it improved by 0.8% *mAcc* and 0.5% *OA*, demonstrating strong competitiveness while requiring only 21% of its FLOPs. Proving the integration method of spatial position relations is crucial for identifying shapes in point clouds.

**Table 1.** Shape classification results on ScanObjectNN.

Methods	Year	mAcc (%)	OA (%)	Params(M)	FLOPs(G)
PointNet++ [21]	2017	75.4	78.9	1.5	1.7
PointCNN [13]	2018	75.1	78.5	0.6	–
DGCNN [30]	2019	73.6	78.1	1.8	2.4
GBNet [24]	2021	77.8	80.5	8.8	–
PRA-Net [3]	2021	77.9	81.0	–	2.3
PointMLP [17]	2022	84.1	85.7	12.6	31.4
PointNeXt [23]	2022	85.8	87.4	1.4	1.6
SPoTr [19]	2023	86.8	88.6	1.7	10.8
<b>HAPE-Net.</b>	2024	<b>87.6</b>	<b>89.1</b>	8.8	2.3



**Fig. 2.** Examples of segmentation on the ShapeNetPart dataset. “GT” refers to the ground truth, while “Ours” represents the segmentation results obtained using the proposed method.

Table 2 and Fig. 2 display the metrics and visualization results of our method on ShapeNetPart, where we evaluated performance using two metrics: instance IoU (*ins. mIoU*) and category IoU (*cls. mIoU*). Our performance was also significantly superior to other models. Although performance on ShapeNetPart has almost reached a bottleneck, HAPE-Net still achieved the best performance breakthrough of 87.4% mIoU, which is 0.2% higher than the previous best method.

**Table 2.** Part segmentation results on ShapeNetPart.

Methods	Year	cls. mIoU	ins. mIoU
PointNet++ [21]	2017	81.9	85.1
PointCNN [13]	2018	84.6	86.1
KPConv [27]	2019	85.1	86.4
PointASNL [33]	2020	–	86.1
PointTransformer [36]	2021	83.7	86.6
PointMLP [17]	2022	84.6	86.1
PointNeXt [23]	2022	85.4	87.1
SPoTr [19]	2023	85.4	87.2
<b>HAPE-Net</b>	<b>2024</b>	<b>85.5</b>	<b>87.4</b>

Table 3 demonstrates that HAPE-Net surpasses all prior methods across all metrics on S3DIS. In the segmentation results of most specific categories, HAPE-Net also performs exceptionally well, particularly for columns. Although HAPE-Net’s performance did not achieve dominance in a few categories, we speculate that this might be due to the network’s tendency to optimize the overall mIoU, leading to overfitting and inevitably neglecting some of the less prevalent structural details. The notable improvement over the previous Transformer architecture [19] (+2.4% mIoU) highlights the critical role of position encoding in the overall understanding and recognition of complex structures in 3D objects.

**Table 3.** Performance comparison of various methods on S3DIS (Area 5). The best result is marked in bold.

Methods	mIoU	OA	mACC	Ceil.	Floor	Wall	Beam	Col.	Wind.	Door	Table	Chair	Sofa	Book.	Board	Clutter
PointCNN [13]	57.3	85.9	63.9	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
PCT [6]	61.3	—	67.7	92.5	98.4	80.6	0.0	19.4	61.6	48.0	76.6	85.2	46.2	67.7	67.9	52.3
KPConv [27]	67.1	—	72.8	93.5	98.4	82.4	0.3	23.9	58.0	69.0	81.5	91.0	58.3	66.7	66.7	57.9
CGA-Net [16]	68.6	—	74.8	94.5	97.8	83.0	0.0	25.3	59.6	71.0	<b>92.2</b>	82.6	66.6	77.7	69.5	61.5
Point Transformer [36]	70.4	—	76.5	94.0	98.5	86.3	0.0	38.0	63.4	74.3	89.1	82.4	74.3	<b>80.2</b>	76.0	59.3
PointNeXt [23]	70.5	90.6	—	—	—	—	—	—	—	—	—	—	—	—	—	—
SPoTr [19]	70.8	90.7	76.4	94.3	98.4	84.0	0.0	34.9	60.2	77.6	83.0	<b>92.5</b>	<b>78.6</b>	76.5	78.8	61.2
<b>HAPE-Net</b>	<b>73.2</b>	<b>92.3</b>	<b>79.0</b>	<b>95.2</b>	<b>98.6</b>	<b>88.9</b>	0.0	<b>57.2</b>	<b>65.2</b>	<b>78.0</b>	86.3	90.9	67.7	76.7	<b>82.9</b>	<b>64.0</b>

### 3.4 Ablation Studies

To strengthen the validity of our conclusions, we carried out comprehensive ablation studies on the ShapeNetPart and ScanObjectNN datasets to ascertain the contribution of each component within our method. The outcomes of these ablation experiments are presented in the table below.

**Position Embedding Method.** To investigate the effects of different position embeddings, we experimented with various combinations on the ShapeNetPart dataset and reported the findings in Table 4. Here, w/o PE indicates the absence of any positional embedding, where the attention mechanism shifts from cross-attention to traditional self-attention. Sin&Cos represents the sinusoidal embedding used in [35], while MLP denotes the classical two-layer MLP structure for positional embedding as utilized in studies like [36]. The experimental results demonstrate that learnable positional embedding methods outperform both the absence of positional embedding and the use of non-learnable positional embedding. Comparatively, HAPE more effectively learns the global positional representations of point clouds, significantly improving the segmentation results of deep models.

**Table 4.** Comparison of different methods of position embedding.

Method	w/o PE	Sin&Cos	MLP	HAPE(ours)
cls. mIoU	84.6	84.8	85.3	<b>85.5</b>
ins. mIoU	86.6	86.4	86.9	<b>87.4</b>

**Table 5.** Types of attention and semantic relationships.

Attention type	Semantic relation	OA (%)
Self	$QK^\top$	87.2
Self	$Q \odot K$	86.9
Self	$Q - K$	88.7
Cross	$QK^\top$	88.1
Cross	$Q \odot K$	87.6
Cross	$Q - K$	<b>89.1</b>

**Table 6.** Ablation study on local aggregation operator

Method	OA (%)	mAcc (%)
w/o LSA	86.3	84.8
w/o Conv	87.4	85.6
w/o CA	88.2	86.4
average	86.5	85.9
max	<b>89.1</b>	<b>87.6</b>
avg+max	88.3	87.0

**Types of Attention and Their Semantic Relationships.** In Table 5, we conducted experiments on ScanObjectNN to compare the impact of different attention types and feature-semantic relationships. The term “self” refers to adding positional embedding directly to the input features at the beginning of a stage and then using regular self-attention to replace the configuration of global position cross-attention, which did not yield promising results. This further validates the effectiveness of the approach we proposed. Namely, employing cross-attention to integrate encoded spatial positional information by HAPE is superior to the method of addition followed by conventional self-attention. This indicates that cross-attention operations maybe more powerful in leveraging point cloud spatial information compared to standard attention mechanisms. Additionally, we experimented with the semantic relationships used in the attention module, showing that the subtractive relationship between query and key is most suitable for modeling the semantic relationships between points.

**Local Aggregation Operator.** In Table 6, the LSA module was ablated to examine the effects of local aggregation operators with different structural designs. We conducted this experiment on ScanObjectNN. The first configuration eliminated the preceding  $1 \times 1$  convolution, the second configuration removed the channel attention following the convolutional layer, and the remaining configurations targeted the choice of reduction function. It is evident that updating the input features before gathering and aggregating neighborhood features is indispensable. Modeling the channel correlations of updated point cloud features through the channel attention(CA) contributed a 0.9% improvement to the network, proving it can aid the model in finding optimal solutions and converging to a better state. Max pooling, as the most commonly used reduction function, still exhibits outstanding performance.

## 4 Conclusion

In this paper, we introduce HAPE-Net, a point cloud transformer that utilizes global positional relations. Our Depth-Adaptive Position Embedding module captures spatial relationships within point clouds, combined with an efficient local aggregation operator and global position cross-attention. This approach effectively captures both intra-set and inter-set shape contexts while maintaining network efficiency, showcasing exceptional results across multiple tasks, such as ScanObjectNN, ShapeNetPart, and S3DIS. We recommend future research to integrate HAPE-Net with additional datasets and explore hybrid models with existing techniques. Additionally, we acknowledge limitations, such as the variability of performance with different data types and the impact of hyperparameters, which necessitate further testing and sensitivity analysis to enhance our findings.

**Acknowledgments.** The paper is supported by the Natural Science Foundation of China (No. 62072388), the public technology service platform project of Xiamen City(No.3502Z20231043) and Fujian Sunshine Charity Foundation.

## References

- Armeni, I., et al.: 3D semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1534–1543 (2016)
- Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1907–1915 (2017)
- Cheng, S., Chen, X., He, X., Liu, Z., Bai, X.: PRA-Net: point relation-aware network for 3D point cloud analysis. *IEEE Trans. Image Process.* **30**, 4436–4448 (2021)
- Choe, J., Park, C., Rameau, F., Park, J., Kweon, I.S.: PointMixer: MLP-mixer for point cloud understanding. In: European Conference on Computer Vision, pp. 620–640. Springer (2022)
- Dai, A., Nießner, M.: 3DMV: joint 3D-multi-view prediction for 3D semantic scene segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 452–468 (2018)
- Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: point cloud transformer. *Comput. Visual Media* **7**, 187–199 (2021)
- Hatamizadeh, A., Yin, H., Heinrich, G., Kautz, J., Molchanov, P.: Global context vision transformers. In: International Conference on Machine Learning, pp. 12633–12646. PMLR (2023)
- Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
- Huang, Z., Liang, D., Xu, P., Xiang, B.: Improve transformer models with better relative position embeddings. arXiv preprint [arXiv:2009.13658](https://arxiv.org/abs/2009.13658) (2020)
- 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. Koonce, B., Koonce, B.: EfficientNet. Convolutional neural networks with swift for Tensorflow: image recognition and dataset categorization, pp. 109–123 (2021)
12. Lai, X., et al.: Stratified transformer for 3D point cloud segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8500–8509 (2022)
13. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: convolution on x-transformed points. *Adv. Neural Inf. Process. Syst.* **31** (2018)
14. Lin, H., et al.: Meta architecture for point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17682–17691 (2023)
15. Liu, Z., Yang, X., Tang, H., Yang, S., Han, S.: FlatFormer: flattened window attention for efficient point cloud transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1200–1211 (2023)
16. Lu, T., Wang, L., Wu, G.: CGA-Net: category guided aggregation for point cloud semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11693–11702 (2021)
17. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual MLP framework. arXiv preprint [arXiv:2202.07123](https://arxiv.org/abs/2202.07123) (2022)
18. Mao, J., et al.: Voxel transformer for 3D object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3164–3173 (2021)
19. Park, J., Lee, S., Kim, S., Xiong, Y., Kim, H.J.: Self-positioning point-based transformer for point cloud understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21814–21823 (2023)
20. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
21. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: deep hierarchical feature learning on point sets in a metric space. *Adv. Neural Inf. Process. Syst.* **30** (2017)
22. Qian, G., Hammoud, H., Li, G., Thabet, A., Ghanem, B.: ASSANet: an anisotropic separable set abstraction for efficient point cloud representation learning. *Adv. Neural. Inf. Process. Syst.* **34**, 28119–28130 (2021)
23. Qian, G., et al.: PointNeXt: revisiting PointNet++ with improved training and scaling strategies. *Adv. Neural. Inf. Process. Syst.* **35**, 23192–23204 (2022)
24. Qiu, S., Anwar, S., Barnes, N.: Geometric back-projection network for point cloud classification. *IEEE Trans. Multimedia* **24**, 1943–1955 (2021)
25. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
26. Tan, M., Le, Q.: EfficientNetV2: smaller models and faster training. In: International Conference on Machine Learning, pp. 10096–10106. PMLR (2021)
27. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6411–6420 (2019)
28. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, T., Yeung, S.K.: Revisiting point cloud classification: a new benchmark dataset and classification model on real-world data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1588–1597 (2019)
29. Vaswani, A., et al.: Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017)

30. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph. (ToG)* **38**(5), 1–12 (2019)
31. Wu, K., Peng, H., Chen, M., Fu, J., Chao, H.: Rethinking and improving relative position encoding for vision transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10033–10041 (2021)
32. Wu, W., Qi, Z., Fuxin, L.: PointConv: deep convolutional networks on 3D point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9621–9630 (2019)
33. Yan, X., Zheng, C., Li, Z., Wang, S., Cui, S.: PointASNL: robust point clouds processing using nonlocal neural networks with adaptive sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5589–5598 (2020)
34. Yi, L., et al.: A scalable active framework for region annotation in 3D shape collections. *ACM Trans. Graph. (ToG)* **35**(6), 1–12 (2016)
35. Zhang, R., Wang, L., Wang, Y., Gao, P., Li, H., Shi, J.: Starting from non-parametric networks for 3D point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5344–5353 (2023)
36. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16259–16268 (2021)
37. Zhou, Y., Tuzel, O.: VoxelNet: end-to-end learning for point cloud based 3D object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499 (2018)



# In-Context Learning for Temperature Field Reconstruction Under Multiple Layouts

Wenzhe Zhang<sup>1,3</sup>, Zixue Xiang<sup>2</sup>, Ming Sun<sup>1</sup>, and Wen Yao<sup>3(✉)</sup>

<sup>1</sup> School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

[sunm@uestc.edu.cn](mailto:sunm@uestc.edu.cn)

<sup>2</sup> College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, China

<sup>3</sup> Defense Innovation Institute, Chinese Academy of Military Science, Beijing, China  
[wendy0782@126.com](mailto:wendy0782@126.com)

**Abstract.** Temperature field reconstruction (TFR) is crucial for the thermal management of complex electronic devices. It involves inferring the entire temperature state from sensor observations. Although previous deep learning-based methods perform well in single layout of heat source scenarios, they often overlook the diverse layouts in the ever-changing real world. To fill this gap, this paper proposes a two-stage training framework to reconstruct the temperature field across multiple layouts. First, to acquire effective reconstruction features, we designed a **Layouts in-context Encoder (LiE)** that operates only on the masked temperature field images, which is essential for feature extraction in the demonstration example. Second, we introduce a feature fusion encoder-decoder architecture. Here, a supervised learning encoder is employed to capture the features of sparse sensor data. By combining the features of the supervised encoder with the layout contextual features of LiE, the decoder can successfully restore the temperature field information. Extensive experiments demonstrate that our method outperforms previous TFR methods in multiple layouts. Furthermore, our approach enables efficient TFR in out-of-distribution (OOD) scenarios.

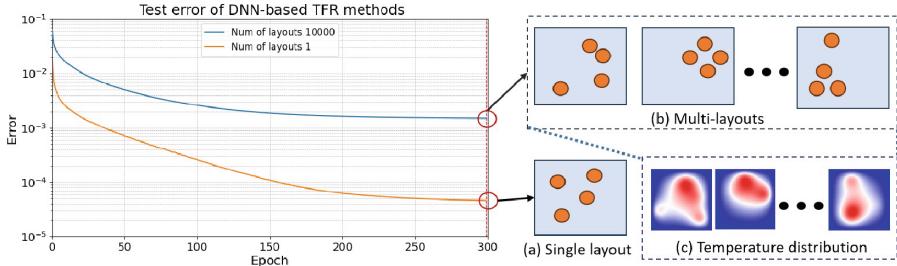
**Keywords:** In-context learning · Temperature field reconstruction · Out-of-distribution scenario

## 1 Introduction

Temperature Field Reconstruction (TFR) is critical for thermal management to ensure optimal performance and prolong the service life of electronic devices [1, 2]. These devices are often equipped with compact, high-power-density components that generate significant amounts of internal heat, qualifying them as “heat sources.” Each specific placement of heat sources constitutes a unique

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2025  
M. Mahmud et al. (Eds.): ICONIP 2024, LNCS 15286, pp. 211–225, 2025.

[https://doi.org/10.1007/978-981-96-6576-1\\_15](https://doi.org/10.1007/978-981-96-6576-1_15)



**Fig. 1.** Test error of DNN-based TFR methods across varying numbers of layouts. Each specific placement of heat sources constitutes a unique layout. (a) represents the case with single heat source layout, while (b) shows the case with multiple heat source layouts. (c) displays the temperature fields for the corresponding heat source layouts in (b).

layout. The varying intensities and layouts of these heat sources form a heat source system [3, 4]. The main objective of TFR is to accurately reconstruct the temperature distribution of the heat source system from sparse sensor observations [5, 6]. However, due to the complex nonlinear relationships within the heat source system, reconstructing its temperature field is typically very challenging [7–10].

There are several traditional methods to reconstruct the temperature field. Interpolation-based methods estimate temperature values within a field by interpolating among known measurements at sparse sensor positions. Moreover, proper orthogonal decomposition [11, 12] is an effective technique that utilizes sparse observations to establish linear equations that determine the coefficients of these bases, thereby reconstructing the entire field. However, the high performance of these traditional methods relies heavily on the collection of data from a large number of sensors.

Recent advancements in deep neural networks (DNN) for TFR have showcased its proficiency in modeling complex nonlinear relationships within physical fields [5, 13, 14]. These DNN-based TFR methods employ various network architectures (e.g., CNN [5], GNN [14]) to directly learn the mapping between sparse sensor measurements and temperature fields. However, existing DNN-based TFR methods face several issues: **(1) They perform well in single layouts of heat source scenarios but overlook the multiple layouts in the real world.** Fig. 1 illustrates that as the variety of heat source layouts increases from 1 to 10,000, the prediction error of DNN-based TFR escalates by two orders of magnitude (i.e., from  $10^{-4}$  to  $10^{-2}$ ). **(2) It is difficult to fine-tune or retrain existing models to enhance their generalizability across different layouts.** A major limitation in extending previous DNN-based TFR is the pervasive scarcity of data. Data for TFR is typically gathered through numerical simulations. While numerical simulations can provide data for specific layouts, executing high-fidelity simulations of complex systems requires substantial computational resources and time. In conclusion, it is crucial to propose a new solu-

tion that consumes fewer computational resources to enhance the performance of TFR across various layouts.

In-context learning (ICL) [15] is regarded as a resource-efficient strategy for adapting to new tasks, widely used in large-scale models such as GPT [16] and BERT [17]. It allows the model to adapt to new scenarios by merely observing a small number of demonstration examples, eliminating the need for retraining or finetuning the entire model. In this paper, we introduce ICL to the task of TFR, aiming to enhance the model's generalizability across various layouts. Specifically, we propose a two-phase training framework for reconstructing the temperature field. Initially, we employ a self-supervised learning that applies random masks to the temperature field. This strategy facilitates the development of a Layouts in-context Encoder (LiE) that encodes contextual information from demonstration examples. Subsequently, we propose a feature fusion encoder-decoder architecture. Here, a supervised learning encoder is employed to capture the features of sparse sensor measurements. By integrating the supervised encoder's features with LiE's in-context features, the decoder effectively restores temperature field information. Our main contributions are as follows:

- To the best of our knowledge, we first introduce in-context learning to TFR. It effectively reduces TFR prediction error under multiple layouts and the out-of-distribution (OOD) scenarios without any finetuning.
- We introduce a two-stage framework that employs self-supervised learning and supervised learning, where the proposed Layouts in-context Encoder (LiE) effectively extracts the layout contextual information from the demonstration example.
- Extensive experimental evaluations have demonstrated that our method significantly enhances the accuracy of temperature field reconstruction under multiple layout conditions and OOD scenarios, outperforming established baseline methods.

## 2 Related Works

### 2.1 DNN-Based TFR Methods

The aim of temperature field reconstruction (TFR) is to infer the entire temperature state of electronic devices from sparse sensor observations. It is crucial for thermal management in various engineering fields, such as electronic engineering and systems design, potentially extending to applications in geophysics, thermodynamics, and aerodynamics [18–20]. While traditional TFR methods struggle with the sparsity of sensor data, deep learning-based TFR methods [3, 4, 6, 21, 22] are increasingly becoming the mainstream solutions due to their effective mapping of sparse sensor data to temperature field [2, 5]. Fukami et al. [5] proposed CNN-based architectures with nearest distance interpolation to adeptly manage sensor number and placement variations. Santos et al. [23] introduced the “Senseiver”, a Transformer-based model that encodes variably sized sets of sparse

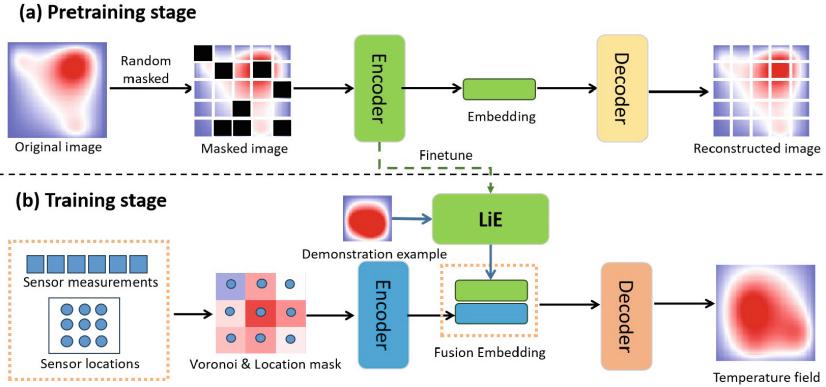
inputs into a latent space using cross-attention, ensuring efficient inference by decoding only the necessary sparse observations. To further improve the accuracy of TFR, researchers continue to explore innovative methods to overcome existing challenges. Zhao et al. [3, 4] integrated Proper Orthogonal Decomposition (POD) into TFR to further enhance the accuracy of DNN-based methods. More research enhanced prediction accuracy by adjusting the positions of sparse sensors [18] and incorporating physical equations as constraints within neural networks [24]. Despite the advancements in DNN-based TFR, these methods face a significant limitation: a notable decline in prediction accuracy with increasing diversity in layout types.

## 2.2 Self-supervised Learning

Self-supervised learning, a subset of unsupervised learning, leverages unlabeled data to enable models to acquire deeper semantic representations. In natural language processing, large language models (LLMs) engage in self-supervised pre-training by predicting next or masked tokens in unlabeled text sequences [25, 26]. In computer vision, contrastive learning as a kind of efficient paradigm of self-supervised learning [27–29] has become popular for assessing image similarity and differences across multiple views. Radford et al. [29] utilize the CLIP framework to decode the intricate relationships between visual content and textual descriptions from large-scale unlabeled image and text data. He et al. [30] propose masked autoencoders (MAE) that are scalable self-supervised learners for computer vision. It masks random patches of the input image and reconstructs the missing pixels. In our pretraining stage, we train a masked autoencoder that employs self-supervised learning for reconstructing the temperature field.

## 2.3 In-Context Learning

In-context learning (ICL) is a new paradigm in domain adaptation that originates from large pre-trained models, such as GPT-3 [31]. Unlike traditional adaptation methods, ICL does not require any updates to model parameters. Instead, it allows the model to adapt to new scenarios by merely observing a small number of demonstration examples [32]. For example, in natural language processing, one might provide a model with a Chinese-English sentence pair and a test sentence in Chinese as input, and then the model produces the English version of the sentence. In computer vision, such a paradigm has only recently begun to be studied. Zhang et al. [33] conduct research on how to better utilize large vision models through the lens of in-context learning. It proposes a prompt retrieval framework specifically for large vision models, allowing the selection of in-context examples to be fully automated. Bar et al. [34] train masked auto-encoders on an unlabeled dataset, which incorporate the in-context information to guide the model’s prediction on the unseen datasets. In this paper, we introduce ICL to the task of temperature field reconstruction, with the aim of enhancing the model’s generalizability on multiple layout datasets and OOD scenario datasets.



**Fig. 2.** Overview of two-stage learning approach for temperature field reconstruction. (a) involves pretraining with masked temperature images for encoder-decoder based image reconstruction. (b) represents that sensor measurements and the demonstration example are encoded to a fusion embedding, which is then decoded to a temperature field. LiE represents the fine-tuned encoder from the pretraining stage.

### 3 Methodology

#### 3.1 Pipeline

**Overall Architecture.** We have developed a two-stage framework that integrates both supervised and self-supervised learning approaches for reconstructing the temperature field. Our architecture comprises two main phases: the pretraining stage and the training stage. In the pretraining stage, self-supervised learning is employed to develop a robust encoder that captures comprehensive semantic information about the temperature field. In the training stage, we introduce In-context learning (ICL) [15] to our task, aiming to enhance the model’s generalizability across diverse layouts. Specifically, we finetune the pretraining stage’s encoder to the **Layout in-context Encoder (LiE)**. It can effectively extract the in-context information of demonstration examples. Moreover, we train a feature fusion encoder-decoder network to learn the mapping from the sparse observations to temperature field. The entire pipeline of the proposed method is depicted in Fig. 2.

**Pre-training Stage.** Motivated by He’s pioneering work [30] on Masked Autoencoders (MAEs), the main objective of our pretraining stage is to train a strong Encoder that employs self-supervised learning for reconstructing the masked temperature field. Unlike previous MAEs that mask patches of natural images, our approach masks individual pixels to enhance the model’s comprehension of temperature field data. This adaptation is crucial due to the significant differences between natural images and temperature field images. The latter contains the subtle temperature variations that are more easily learned through pixel-level reconstruction tasks. As shown in Fig. 2(a), large random

pixels of temperature image  $T$  are masked out, and then the masked image  $T_m$  are processed by encoder-decoder architecture to reconstruct the temperature filed image. The mask operation is defined as:

$$T_m = P \odot T, \quad (1)$$

where  $P$  represents the mask matrix and  $\odot$  denotes the element-wise (Hadamard) product, with  $P$ ,  $T_m$ , and  $T \in \mathbb{R}^{n_x \times n_y}$ . It involves randomly masking a proportion of the image's pixels, calculated as  $n_x \times n_y \times \alpha$ , where  $\alpha$  specifies the mask ratio. Consequently, during subsequent training phases, this encoder requires only minimal finetuning to effectively extract and articulate layout-specific context information from the demonstration example, thus forming our Layouts in-context Encoder (LiE).

**Training Stage.** As shown in Fig. 2(b), our training stage concurrently encompasses two key processes: 1) training the encoder-decoder architecture and 2) finetuning the pretraining stage's encoder to the Layout in-context Encoder (LiE).

During the first process, we employ a supervised learning method on multi-layout datasets to train our backbone encoder-decoder network. For a specific training dataset consisting of  $K$  unique layouts, each containing  $I$  samples per layout. Each sample comprises the sparse sensor measurements and their corresponding location. Since these sparse observations are one-dimension vectors, they need to be transformed into a two-dimension matrix that the encoder can handle. Here, we apply the Voronoi tessellation [5] to achieve this transformation. This process divides the computational domain  $V \in \mathbb{R}^{n_x \times n_y}$  into  $S$  regions based on proximity to sensor locations, assigning a single observation value to each region and setting unknown points within that region to the sensor's value. In this way, we effectively transform the one-dimensional sensor data  $O_{ki}$  into a structured two-dimensional data  $\hat{O}_{ki}$ . Concurrently, a positional mask matrix  $M$  is created to help the encoder identify the exact position of the sensor. In  $M$ , the positions of the sensor are marked with 1, and all other positions are set to 0. Finally, the two-dimensional sparse observations  $\hat{O}_{ki}$  and the position mask  $M$  are combined to form an input for the encoder. The output of the encoder can be expressed as:

$$E_{obs} = Encoder(\hat{O}_{ki}, M). \quad (2)$$

During the second process, we adapt the Encoder that is trained in the pretraining stage to the Layout in-context Encoder (LiE). The finetuning datasets are the same as training datasets. The main objective of LiE is to learn the in-context layout information from the demonstration example  $T_{kj}$ , where  $j \neq i$ ;  $i, j \in [1, I]$ ,  $k \in [1, K]$ . The demonstration example is randomly selected from each layout to ensure a diverse and robust representation of contexts. Then,  $T_{kj}$  are fed into LiE to generate the layout contextual embedding  $E_{layout}$ . Finally, we concatenate the embedding  $E_{layout}$  from demonstration examples and the embedding  $E_{obs}$  from sparse observations together to form a fusion embedding,

which is subsequently fed into the decoder for prediction. The process above can be mathematically represented as follows:

$$\begin{aligned} E_{obs} &= Encoder\left(\hat{O}_{ki}, M; \theta_1\right), \\ E_{layout} &= LiE(T_{kj}; \kappa), \\ \hat{T}_{ki} &= Decoder(E_{layout}, E_{obs}; \theta_2), \end{aligned} \quad (3)$$

where  $\hat{T}_{ki}$  is the final output.  $\theta_1$  and  $\theta_2$  are parameters of the encoder and decoder in the training stage, respectively.  $\kappa$  denotes the trainable parameters of the LiE.

The overall optimization objective for the training stage of our proposed framework can be defined as follows:

$$\operatorname{argmin}_{\theta_1, \theta_2, \kappa} \left| T_{ki} - \text{Decoder}\left(\text{LiE}(\hat{T}_{kj}; \kappa), \text{Encoder}(\hat{O}_{ki}, M; \theta_2); \theta_1\right) \right|, \quad (4)$$

where  $\theta_1$ ,  $\theta_2$ , and  $\kappa$  are the parameters that we need to optimize. It is worth noting that  $\kappa$  set with a small learning rate that aims to finetune the pre-trained encoder to LiE. This finetuning process adapts the encoder's general feature-extracting ability to specific downstream tasks of TFR under multiple layouts. Moreover,  $\theta_1$  and  $\theta_2$  set with higher learning rates to train a backbone from scratch for TFR.

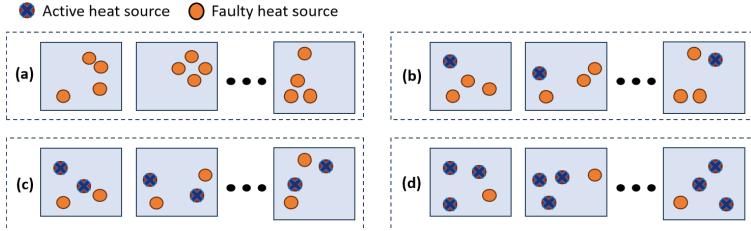
### 3.2 Dataset Construction

Given that previous TFR datasets are limited to single layout, we developed *multi-layout datasets* based on standard settings of heat-source system [3, 4, 21]. Moreover, to validate the robustness of our method in practical development, we also constructed *OOD scenario datasets* that aim to simulate the abnormal cases in the real world. The specific construction is as follows:

**Configuration of Datasets.** We place four Gaussian heat sources on a two-dimensional plane  $\Omega = [0, 2] \times [0, 2]$ , each of which can represent a heat-generating component. The location of the heat source can be expressed as follows:  $(a_i, b_i) \in \Omega, i = 1, 2, 3, 4$ , with  $(a_i, b_i) \sim Uniform(\Omega)$ . Each specific placement of four heat sources constitutes a unique layout. The intensity distribution,  $f$ , of each heat source is mathematically expressed as follows:

$$f(x, y, a_i, b_i, c_i) = -c_i \exp\left(-\frac{(x - a_i)^2 + (y - b_i)^2}{0.05}\right), \quad i = 1, 2, 3, 4 \quad (5)$$

For each intensity distribution  $f$ , the parameter  $c_i$  is a random variable uniformly distributed between 1000 and 6000.



**Fig. 3.** illustration of OOD scenarios datasets. (a) to (d) correspond to cases 1 to 4, respectively, illustrating different states of the heat sources in OOD scenario.

The steady-state heat source systems are governed by the heat conduction equation:

$$\lambda \frac{\partial^2 T}{\partial x^2}(x, y) + \lambda \frac{\partial^2 T}{\partial y^2}(x, y) + \sum_{i=1}^4 f(x, y, a_i, b_i, c_i) = 0 \quad (x, y) \in \Omega, \\ T(x, y) = T_D \quad (x, y) \in \partial\Omega_D, \quad (6)$$

where  $\Omega_D$  is the Dirichlet boundary, and  $\lambda$  is the conductivity coefficient and is set to be 1 in this work. To prepare the training data, the computational domain is divided into  $64 \times 64$  quadrilateral mesh, and Radial Basis Function Generated Finite Differences (RBF-FD) [35] is adopted to solve the partial differential Eq. 6 with varying intensity distribution  $f(x, y, a, b, c)$  under a variety of layouts.

For each sample within our dataset, the data structure consists of a pair of elements: sparse observations  $O$  and the corresponding ground truth temperature field  $T$ . The sparse observations  $O$  are derived from 16 sensors, which are uniformly distributed across the computational domain  $\Omega$ , capturing the sensor measurements in a 16-dimensional vector.

**Multi-layout Datasets.** In the multi-layout datasets, we construct five datasets, each containing varying numbers of layouts: 1, 50, 200, 500, and 10,000. Each layout within these datasets holds an equal number of samples, standardizing the total at 100,000 samples per dataset to ensure comparability. Additionally, 20% of the samples from each layout type are allocated as the validation datasets.

**OOD Scenario Datasets.** In the OOD scenario datasets, we construct four distinct datasets corresponding to four abnormal cases in the real world, as illustrated in Fig. 3. The first dataset comprises 2,000 layouts the model has not previously encountered, with each layout containing an equal number of samples. The subsequent datasets are designed with three, two, and one heat sources, respectively, to simulate real-world scenarios where the components (corresponding to these heat sources) experience failures. Each OOD scenario dataset contains only one validation dataset, which includes 20,000 samples in total, specifically designed to directly assess the model's performance under OOD scenarios.

## 4 Experiments

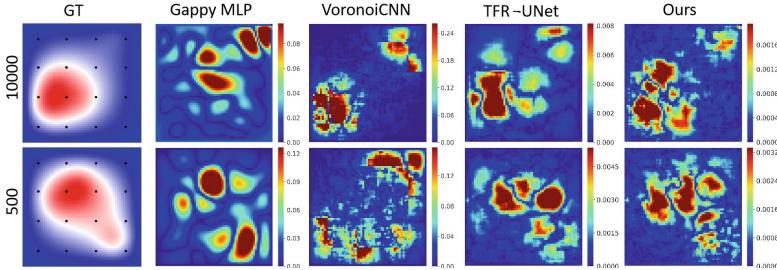
### 4.1 Experiments Setup

**Datasets and Metrics.** We constructed both Multi-Layout Datasets and OOD scenario Datasets for the heat source system to evaluate the performance of various TFR methods across diverse layouts and OOD cases. Details on the dataset designs are provided in Sect. 3.2. For evaluation metrics, we adopt the Mean Absolute Error (MAE) and Maximum Absolute Error (Max-AE) to access the error between the predicted temperature field  $\hat{T}$  and the ground truth temperature field  $T$ . The MAE and Max-AE are defined as  $\varepsilon = \frac{\|\hat{T}-T\|}{|T|}$  and  $\varepsilon_{max} = \frac{\|\hat{T}-T\|_\infty}{|T|}$ , respectively.

**Implementation Details.** Our proposed method is compared with other representative DNN-based TFR methods, including Gappy MLP [3], TFR-UNet [21], Shallow decoder [13], and VoronoiCNN [5]. We adopt the UNet [36] architecture as the backbone for our two-stage framework. Each model is trained over 300 epochs with a batch size of 16. The Adam optimizer with an initial learning rate of  $10^{-3}$  is employed to optimize the network parameters and the exponential decay strategy is adopted to adjust the learning rate with a multiplicative factor of 0.98. After extensive experimentation with a spectrum of mask ratios, the optimal mask ratio for LiE during the pre-training phase was determined to be 0.7 to obtain the best final performance. For fine-tuning during the training phase, the learning rate of LiE was meticulously reduced to  $10^{-5}$  to refine its capacity for layout-specific context extraction and representation. All methods are implemented using PyTorch and trained on one NVIDIA RTX 4090.

### 4.2 Evaluation of Multi-layout Datasets

Table 1 compares the performance of various TFR methods on the datasets across a varying number of layouts. Our method achieves the lowest predicted error across all layouts, demonstrating its superiority over existing methods (i.e., Gappy MLP [3], TFR-UNet [21], Shallow decoder [13], and VoronoiCNN [5]) on multi-layout datasets. We notice that as the number of layouts increases, prediction errors become more noticeable. For example, under a single layout, the Mean Absolute Error (MAE) of TFR-UNet and our method are very close, at 0.046 and 0.045, respectively. But in complex scenarios with up to 10,000 layouts, our method outperforms TFR-UNet by a considerable margin, reducing the MAE by 80.6% from 2.717 to 0.526 and the Max-AE by 70.5% from 3.117 to 0.917. This phenomenon indicates that the previous approaches can effectively learn mapping relationships only under fixed layouts. In contrast, by capturing the layout contextual information from the demonstration example, our approach significantly reduces the prediction error for TFR under multiple layouts without additional training. The gap between the prediction and the exact temperature field is visualized in Fig. 4. The error distribution indicates that our method provides more accurate reconstruction results.



**Fig. 4.** Visualization for the test error of various TFR methods under multi-layout datasets. Red to blue represents the error from high to low. Our method significantly surpasses other TFR methods on 500 and 10000 layout datasets. (Color figure online)

**Table 1.** Comparison of MAE and Max-AE with other methods under different number of layouts.

Methods	Mean Absolute Error(e-3)					Max Absolute Error(e-2)				
	1	50	200	500	10000	1	50	200	500	10000
VoronoiCNN [5]	0.891	19.017	29.567	44.528	79.735	1.720	29.442	44.857	66.759	99.938
Shallow decoder [13]	0.929	9.436	17.514	26.319	28.225	0.596	5.656	13.837	20.940	27.857
Gappy MLP [3]	0.521	8.282	16.172	25.963	27.758	0.700	5.664	13.116	19.962	27.067
TFR-UNet [21]	0.047	0.270	0.366	0.527	2.717	0.046	0.275	0.402	0.512	3.117
Ours	<b>0.045</b>	<b>0.267</b>	<b>0.361</b>	<b>0.470</b>	<b>0.526</b>	<b>0.040</b>	<b>0.273</b>	<b>0.384</b>	<b>0.501</b>	<b>0.917</b>

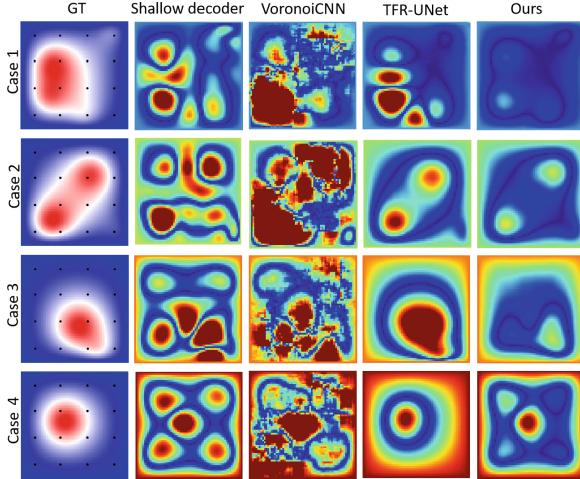
#### 4.3 Evaluation of OOD Scenarios Datasets

In evaluating the performance of various TFR methods in OOD scenarios, we construct four OOD scenario datasets corresponding to different real-world cases with four to one heat sources (refer to Sect. 3.2 for more details), and Fig. 3 shows a visual diagram of corresponding scenarios.

Table 2 compares MAE and Max-AE with representative methods under various OOD scenario datasets. Our method achieves the lowest predicted error across OOD cases, demonstrating its superiority over existing methods (i.e., TFR-UNet [21], Shallow decoder [13], and VoronoiCNN [5]) on OOD scenarios. For example, in Case 1, our method reduces the MAE from 1.306 to 0.158 compared with TFR-UNet. Although TFR-UNet achieves a low MAE of 1.306, it struggles with a high Max-AE of 16.725. In contrast, our method also excels in reducing Max-AE, bringing it down from 16.725 to 1.590. These results underline our method’s superiority in TFR under unseen OOD layouts; the other three cases are designed to simulate downtime in heat-generating components. Despite the increased prediction errors in these complex OOD cases, our method maintains the lowest predicted error than other TFR methods.

Figure 5 visualizes that the predicted error under OOD scenario datasets. The higher temperature areas (red areas in GT) typically indicate the location of the

heat sources, where the model exhibits larger predicted errors due to a lack of specific training for these scenarios. While our method significantly reduces the prediction errors in these areas, demonstrating that our LiE effectively assimilates layout context information from the demonstration example. Compared to other approaches, our proposed framework can markedly decrease prediction errors in complex OOD scenarios.



**Fig. 5.** Visualization for the test error of various TFR methods under OOD scenario datasets. Case 1 through 4 correspond to system containing four to one heat sources. Red to blue represents the error from high to low. Our method significantly surpasses other TFR methods on different OOD cases. (Color figure online)

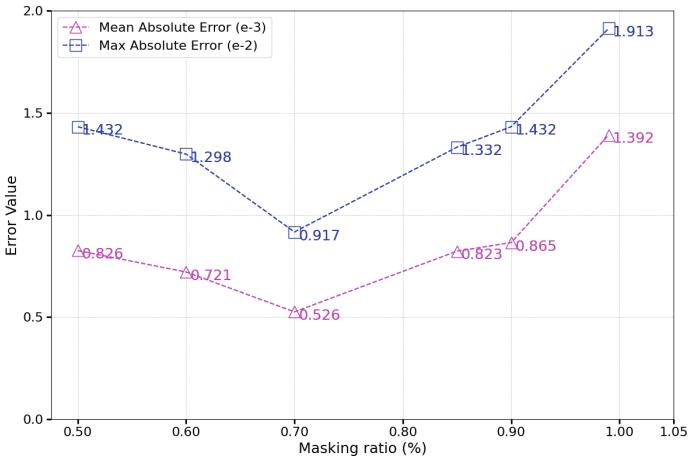
**Table 2.** Comparison of MAE and Max-AE with other methods under various OOD scenario datasets.

Methods	Mean Absolute Error(e-2)				Max Absolute Error(e-1)			
	Case 1	Case 2	Case 3	Case 4	Case 1	Case 2	Case 3	Case 4
VoronoiCNN [5]	8.735	19.526	9.588	11.866	2.779	4.014	4.956	5.388
Shallow decoder [13]	2.878	8.719	10.689	12.536	10.493	18.112	6.379	7.978
TFR-UNet [21]	1.306	6.125	70.115	14.179	16.725	2.039	3.419	4.926
Ours	<b>0.158</b>	<b>4.802</b>	<b>7.801</b>	<b>8.821</b>	<b>1.590</b>	<b>1.459</b>	<b>1.590</b>	<b>2.151</b>

#### 4.4 Ablation Studies

**Masking Ratio.** Figure 6 presents the Mean Absolute Error (MAE) and Maximum Absolute Error (Max-AE) for different masking ratios during the pre-training stage on multi-layout datasets. At a masking ratio of 0.5, the MAE

is 0.826, and the Max-AE is 1.432. When the masking ratio increases to 0.6, the MAE decreases to 0.721 and the Max-AE to 1.298. The lowest errors are observed at a masking ratio of 0.7, with an MAE of 0.526 and a Max-AE of 0.917. However, as the masking ratio continues to rise, the errors start to increase. At a masking ratio of 0.85, the MAE is 0.823, and the Max-AE is 1.432, while at 0.9, the MAE is 0.865, and the Max-AE is 1.332. The highest errors occur at a masking ratio of 0.99, with an MAE of 1.392 and a Max-AE of 1.913. This indicates that while some masking improves the training effectiveness of the pre-training stage, an excessively high masking ratio significantly degrades it.



**Fig. 6.** Comparison of Mean Absolute Error (MAE) and Max Absolute Error (Max-AE) for different masking ratios in the pre-training stage on multi-layout datasets.

**Table 3.** Comparison of MAE and Max-AE for different modules on a dataset with 10,000 layouts. “Pretrained LiE” denotes a Layouts in-context Encoder that has undergone pre-training stage. The masking ratio of the pretraining stage is set to 0.7

LiE	Pretrained	MAE(e-3)	Max-AE(e-2)
—	—	2.717	3.117
✓	—	<b>0.933</b>	<b>1.528</b>
✓	✓	<b>0.526</b>	<b>0.917</b>

**LiE and Pretraining Stage.** To verify the effectiveness of our LiE module, we conducted ablation experiments with and without pre-training. As shown in Table 3, the MAE is reduced from 2.717 to 0.933 and the MAX-AE from 3.117

to 1.528 by introducing the LiE. This demonstrates that using our LiE module can effectively improve the accuracy of the temperature construction task under multiple layouts. Furthermore, after pre-training the LiE, the MAE and Max-AE are further reduced from 1.528 to 0.917. This proves that our pre-training stage significantly enhances the LiE’s understanding of pixel-level semantics in the temperature field, allowing efficient representation of layout context information in the demonstration example.

## 5 Conclusion

In this paper, we propose a two-stage training framework that employs both self-supervised and supervised learning for temperature field reconstruction (TFR). Our method centers on a Layouts In-context Encoder (LiE), which is initially trained on masked temperature field datasets to effectively capture the contextual information. The integration of LiE with a feature fusion encoder-decoder architecture allows the model to combine sparse observations with contextual features, thereby enhancing its ability to effectively reconstruct temperature fields. By implementing in-context learning with a single demonstration example, we can improve prediction accuracy for TFR under multiple layouts and out-of-distribution (OOD) scenarios. Experimental results highlight the potential of in-context learning to address the challenges of data scarcity, presenting a promising avenue for future research in temperature field reconstruction. We expect that this work shall draw greater attention to the complex physical field reconstructions and serve as a baseline that facilitates further research in this direction.

## References

1. Zhang, Y., Gong, Z., Zhao, X., Yao, W.: Uncertainty guided ensemble self-training for semi-supervised global field reconstruction. *Complex Intell. Syst.* **10**(1), 469–483 (2024)
2. Zhao, X., Chen, X., Gong, Z., Zhou, W., Yao, W., Zhang, Y.: RecFNO: a resolution-invariant flow and heat field reconstruction method from sparse observations via fourier neural operator. *Int. J. Therm. Sci.* **195**, 108619 (2024)
3. Zhao, X., Gong, Z., Chen, X., Yao, W., Zhang, Y.: A unified framework of deep neural networks and Gappy proper orthogonal decomposition for global field reconstruction. In: 2023 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2023)
4. Zhao, X., Chen, X., Gong, Z., Yao, W., Zhang, Y.: A hybrid method based on proper orthogonal decomposition and deep neural networks for flow and heat field reconstruction. *Expert Syst. Appl.* **247**, 123137 (2024)
5. Fukami, K., Maulik, R., Ramachandra, N., Fukagata, K., Taira, K.: Global field reconstruction from sparse sensors with Voronoi tessellation-assisted deep learning. *Nat. Mach. Intell.* **3**(11), 945–951 (2021)
6. Xia, Y., Yao, W., Zheng, X., Gong, Z.: Reliability analysis of heat source layout temperature field prediction considering uncertainty in deep neural network surrogate models. *Qual. Reliab. Eng. Int.* **39**(5), 1775–1795 (2023)

7. Fukami, K., Goto, S., Taira, K.: Data-driven nonlinear turbulent flow scaling with Buckingham pi variables. arXiv preprint [arXiv:2402.17990](https://arxiv.org/abs/2402.17990) (2024)
8. Shengfeng, X., Sun, Z., Huang, R., Guo, D., Yang, G., Shengjun, J.: A practical approach to flow field reconstruction with sparse or incomplete data through physics informed neural network. *Acta. Mech. Sin.* **39**(3), 322302 (2023)
9. Morimoto, M., Fukami, K., Zhang, K., Fukagata, K.: Generalization techniques of neural networks for fluid flow estimation. *Neural Comput. Appl.* **34**(5), 3647–3669 (2022)
10. Zhong, Y., Fukami, K., An, B., Taira, K.: Sparse sensor reconstruction of vortex-impinged airfoil wake with machine learning. *Theoret. Comput. Fluid Dyn.* **37**(2), 269–287 (2023)
11. Saini, P., Arndt, C.M., Steinberg, A.M.: Development and evaluation of Gappy-POD as a data reconstruction technique for noisy PIV measurements in gas turbine combustors. *Exp. Fluids* **57**, 1–15 (2016)
12. Henneron, T., Clenet, S.: Model order reduction of non-linear Magnetostatic problems based on POD and DEI methods. *IEEE Trans. Magn.* **50**(2), 33–36 (2014)
13. Erichson, N.B., Mathelin, L., Yao, Z., Brunton, S.L., Mahoney, M.W., Kutz, J.N.: Shallow neural networks for fluid flow reconstruction with limited sensors. *Proc. R. Soc. A* **476**(2238), 20200097 (2020)
14. Li, J., Liu, T., Wang, Y., Xie, Y.: Integrated graph deep learning framework for flow field reconstruction and performance prediction of turbomachinery. *Energy* **254**, 124440 (2022)
15. Dong, Q., et al.: A survey on in-context learning. arXiv preprint [arXiv:2301.00234](https://arxiv.org/abs/2301.00234) (2022)
16. Brown, T., et al.: Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020)
17. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint [arXiv:1810.04805](https://arxiv.org/abs/1810.04805) (2018)
18. Manohar, K., Brunton, B.W., Kutz, J.N., Brunton, S.L.: Data-driven sparse sensor placement for reconstruction: demonstrating the benefits of exploiting known patterns. *IEEE Control Syst. Mag.* **38**(3), 63–86 (2018)
19. Li, B.J., Liu, H.R., Wang, R.Z.: Data-driven sensor placement for efficient thermal field reconstruction. *Sci. China Technol. Sci.* **64**(9), 1981–1994 (2021). <https://doi.org/10.1007/s11431-020-1829-2>
20. Liu, T., Li, Y., Jing, Q., Xie, Y., Zhang, D.: Supervised learning method for the physical field reconstruction in a nanofluid heat transfer problem. *Int. J. Heat Mass Transf.* **165**, 120684 (2021)
21. Peng, X., Li, X., Gong, Z., Zhao, X., Yao, W.: A deep learning method based on partition modeling for reconstructing temperature field. *Int. J. Therm. Sci.* **182**, 107802 (2022)
22. Liu, X., Peng, W., Gong, Z., Zhou, W., Yao, W.: Temperature field inversion of heat-source systems via physics-informed neural networks. *Eng. Appl. Artif. Intell.* **113**, 104902 (2022)
23. Santos, J.E., Fox, Z.R., Mohan, A., O’Malley, D., Viswanathan, H., Lubbers, N.: Development of the senseiver for efficient field reconstruction from sparse observations. *Nat. Mach. Intell.* **5**(11), 1317–1325 (2023)
24. Shengfeng, X., Sun, Z., Huang, R., Guo, D., Yang, G., Shengjun, J.: A practical approach to flow field reconstruction with sparse or incomplete data through physics informed neural network. *Acta. Mech. Sin.* **39**(3), 322302 (2023)

25. McCallen, D., et al.: EQSIM—a multidisciplinary framework for fault-to-structure earthquake simulations on exascale computers part i: Computational models and workflow. *Earthq. Spectra* **37**(2), 707–735 (2021)
26. Radford, A., Jeffrey, W., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. *OpenAI Blog* **1**(8), 9 (2019)
27. Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning? *Adv. Neural. Inf. Process. Syst.* **33**, 6827–6839 (2020)
28. Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F.: A survey on contrastive self-supervised learning. *Technologies* **9**(1), 2 (2020)
29. Radford, A., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp. 8748–8763. PMLR (2021)
30. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16000–16009 (2022)
31. Brown, T., et al.: Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020)
32. Dong, Q., et al.: A survey on in-context learning. arXiv preprint [arXiv:2301.00234](https://arxiv.org/abs/2301.00234) (2022)
33. Zhang, Y., Zhou, K., Liu, Z.: What makes good examples for visual in-context learning? *Adv. Neural Inf. Process. Syst.* **36** (2024)
34. Bar, A., Gandelsman, Y., Darrell, T., Globerson, A., Efros, A.: Visual prompting via image inpainting. *Adv. Neural. Inf. Process. Syst.* **35**, 25005–25017 (2022)
35. Fornberg, B., Flyer, N.: A primer on radial basis functions with applications to the geosciences. SIAM (2015)
36. 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](https://doi.org/10.1007/978-3-319-24574-4_28)



# Loosely Coupled Oscillators as a Correlate of Behavioral Control Circuits Within the Central Complex of the Fruit Fly

Saul Garnell<sup>1</sup>(✉), Mehmet Turkcan<sup>2</sup>, Maryam Daborjeh<sup>1</sup>, Brian Smith<sup>3</sup>,  
and Paul Szyszka<sup>4</sup>

<sup>1</sup> Auckland University of Technology, Auckland 1010, New Zealand  
[saul.garnell@autuni.ac.nz](mailto:saul.garnell@autuni.ac.nz)

<sup>2</sup> Columbia University, New York 10027, USA

<sup>3</sup> Arizona State University, Tempe 85287, USA

<sup>4</sup> University of Otago, Dunedin 9016, New Zealand

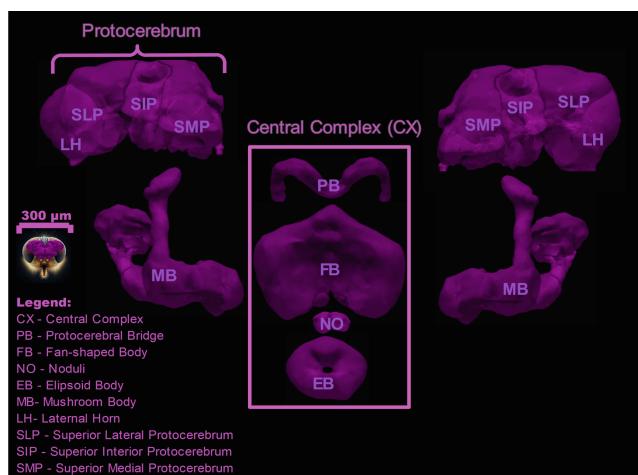
**Abstract.** In this study, we propose a novel methodology to model and analyze a circuit mechanism within the Fan-shaped body (FB), a core neuropil within the Central Complex of the fruit fly. Because the Fan-shaped body plays a crucial role in controlling both sleep and arousal, we hypothesize that specific neuronal circuits within the fruit fly's FB act as loosely coupled oscillators and provide a control mechanism for opposing behaviors like sleep and arousal. To test our hypothesis in silico we created a multi-step framework of methods for identifying neurons of interest that perform as loosely coupled oscillators in the Central Complex using a statistical pathfinding analysis, graph analysis, and simulated signal analysis approaches. Our study also leverages publicly available computational tools to identify, visualize and simulate neuronal signals to advance the understanding of neural circuit dynamics and their implications for behavior and cognitive processes. The final results leverage the Wilson-Cowan model analysis between two neuron types, one excitatory and one inhibitory, from which we elucidate how stable points demonstrated by Wilson-Cowan analysis act as neuronal attractors to drive behavior. These results can be applied to Bayesian Brain theories like the Free Energy Principle. These results also imply experimental approaches that will help interpret the functionality of pathways identified in the fly brain connectome.

**Keywords:** Computational Neuroscience · Neural circuit dynamics · Pathfinding Analysis · Graph Analysis · Fruit Fly · Sleep and arousal

## 1 Introduction

The fruit fly's nervous system offers a unique opportunity to explore the neural dynamics and mechanisms underlying various sensory and behavioral responses,

with the *fruit fly* Central Complex (CX) (Fig. 1) playing a crucial role in controlling various behaviors such as sleep and arousal [1–4]. However, the entire mechanism for sleep and arousal within the CX is still not fully known, as most well-defined areas within the adult fruit fly brain with specific functions and organization (hereon referred to as neuropils) provide output to the CX, but the function of those connections remain remains poorly understood [1]. Key functions such as sensory integration and locomotion coordination are crucial in understanding the end-to-end neural circuitry involved in sleep and arousal regulation [2]. In this respect, arousal mediated by the CX is a multifaceted process influenced by various sensory signals and neural circuits from different modalities. These include visual and olfactory cues that emanate from higher-order regions within the protocerebrum and include the Lateral Horn [3].



**Fig. 1.** The fruit fly brain is composed of several major neuropils, each with distinct functions and structures. Each of these neuropils contributes to the complex functioning of the *Drosophila* brain, enabling it to process sensory information, coordinate motor functions, and perform other critical tasks.

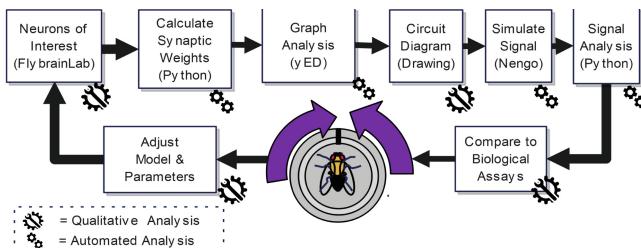
Understanding processes across CNS regions in mammals and invertebrates requires methods that reveal underlying brain activity leading to observed behavior. One method is investigating coupled oscillators within organisms' nervous systems, essential for controlling behaviors like sleep, arousal, and locomotion [4]. Invertebrate brains may have simpler oscillatory activity richness, but local field potential (LFP) oscillations associated with sensory and motor processing indicate the presence of coupled oscillatory mechanisms [5, 6]. We believe certain neural circuits in the fruit fly's FB act as loosely coupled oscillators, controlling behaviors like sleep and arousal. Studying these oscillators involves identifying specific Neurons of Interest (NOIs), which is challenging due to the complexity of an animal's connectome. Mammalian brains can range from around

$10^8$  neurons in mice to  $10^{11}$  neurons in humans, with an even larger number of glial cells [7]. The fruit fly, with an estimated neuron count just under 200,000 [8], is advantageous as a model organism. Advanced techniques like electron microscopy (EM) for detailed brain mapping and connectome analysis have significantly enhanced our understanding of its neural circuitry [9]. EM-based studies have led to detailed datasets like the Hemibrain dataset, enabling computational exploration of the fruit fly connectome. These datasets have allowed for synaptic resolution in neural circuit reconstruction, providing insights into key communication pathways within the fruit fly brain [10]. Despite the detailed physical roadmap of the fruit fly brain provided by datasets like the Hemibrain dataset, gaps in our understanding remain. Even with a detailed connectome, the dataset's complexity makes feature extraction and model building challenging [11].

We believe that easily accessible computational tools are crucial for investigating specific neuronal pathways in the central nervous system, and play a vital role in neuroscience by supporting explanations of brain activity and providing evidence for theories related to animal behavior [12]. Bayesian brain models are a good example of this (see discussion), and posit that the brain performs Bayesian inference, utilizing generative models of the external world to achieve goals [16–18]. We propose a framework to analyze the Fan-shaped body of the fruit fly, simulate feedback loops and steady state oscillators, allowing us to later compare results with neurophysiological data. This will allow us to test and potentially falsify our hypothesis.

## 2 Methodology

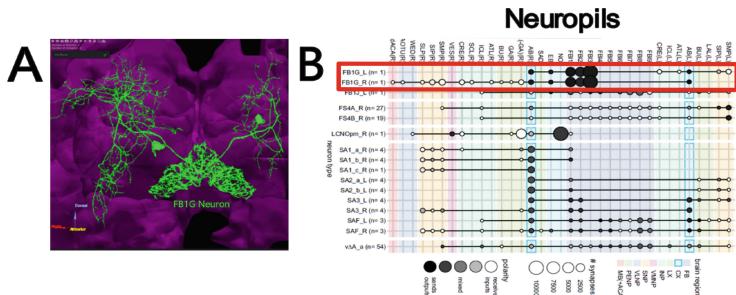
We propose a framework to model and analyze a circuit mechanism within the Fan-shaped Body (FB), a core neuropil within the CX of fruit fly. The multistep framework (Fig. 2) includes methods for: identifying NOIs; calculating synaptic weights, modeling neural connectivity using graphs; developing neural circuit model diagrams; simulating NOIs, and performing signal analysis. Each method within the framework is further elaborated in the following sections.



**Fig. 2.** An overview of the workflow used by the Framework. Top-left: are Neurons of interest are loaded into the FlybrainLab

## 2.1 Initial Survey for Neurons of Interest

We leveraged FlybrainLab, a software package developed by Lazar et al. [13], and is used for investigating the connectome of the fruit fly and offers a powerful approach to explore neuronal pathways [13, 14]. It facilitates the exploration of datasets, particularly the Hemibrain dataset, in a Python-based environment. It integrates neuroanatomical, neurogenetic, and electrophysiological datasets with computational models from various researchers, allowing for validation and comparison within a unified platform [13]. The User Interface offers an intuitive and automated workflow for 3D exploration and visualization of fly brain circuits, enabling interactive exploration of functional logic within selected circuits as part of setting up a general workspace. After the workspace has been loaded (Fig. 3-A), we search for NOIs from which to begin analysis. The workspace shows the GUI 3d output of one specific neuron type named FB1G, which stands for: Fan-shaped Body, layer one, subtype G. Naming of neurons is unique to each dataset even though there is crossover at times. The choice of FB1G as a starting NOI was made by literature review and analysis within FlybrainLab that demonstrated FB1G's innervation that indirectly connects the Superior Protocerebrum (an analog of the mammalian cortex) including the Lateral Horn to the CX. This analysis is important to our search because of the possibility that the Lateral Horn acts as an analog of the human amygdala [15] and outputs an olfactory response to more downstream neuropils (i.e. CX) in parallel to Mushroom Body Output Neurons (MBONs) [15].



**Fig. 3.** Example of neuron innervation for specific neurons in the fruit fly. **a** An example of the FB1G neurons displayed in the FlybrainLab workspace. **b** Fig. 36 B from the paper by Hulse et al., “A connectome of the Drosophila central complex reveals network motifs suitable for navigation and action selection. This graph presents region arborization plots for each neuron type. Highlighted FB1G neurons show innervation to the FB provides a large output to the lower FB layers.

Aiding our search for NOIs was data from the connectome paper by Hulse et al. [12], where the arborization of FB1G into the CX can be seen in Fig. 3-B. The highlighted area shows FB layers 1–3 in the posterior FB have between 7.5k and 10k synapses from FB1G, a synaptic density greater than any other neuron type.

We found this data point a compelling reason to focus on FB1G, even though other factors may exist. For example, it is known that the synaptic contact area, the amount of neurotransmitter released into the synaptic cleft, and the types and number of postsynaptic receptors all have an important role in determining the degree of propagated signal strength between neurons [22]. However, those parameters in numerical form are not obtainable. Neither the Hemibrain dataset nor any other large dataset currently provides a detailed morphology of the synaptic microcircuitry per neuron. Using the number of synapses as an assumed measure of signal strength seems reasonable until more detailed information about the synaptome is published. Simply put, data availability in the underlying dataset is a limiting factor of analysis that can be performed computationally *in silico*.

We loaded our NOIs into the workspace and used FlybrainLab's natural language search engine to explore circuit pathways, allowing us to trace connections of the FB1G neuron toward our target in the CX. This process is qualitative and involves researching literature and reviewing metrics. FlybrainLab's natural language engine speeds up the search process before automated analysis using Python algorithms.

In our example, we used a cutoff of 30 synapses to determine significant pre- and post-synaptic connections, but this number can vary. It's important to control the number of pathways and ensure that the neurons and synapses are reasonable for identifying significant pathways to and from the CX.

## 2.2 Neuron Pathfinding Analysis

In the Neuron Pathfinding Analysis step, we use a Python algorithm to identify significant neuronal connections based on synaptic weights. This automated process analyzes neurons in the FlybrainLab workspace, calculating weighted connections between them. The workspace contains several pre- and post-synaptic neurons, considered as NOIs. At least two neurons are required to assess synaptic strength. We compare the innervation of two neurons to the synaptic distribution across all synaptic partners. For instance, if neurons A and B have 52 post-synapses, we compute how these compare to all of A's potential postsynaptic connections. If A only connects to B, the 52 synapses represent 100% of A's postsynaptic connectivity.

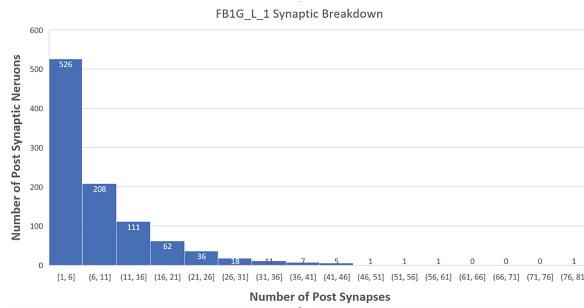
Most connections involve only a small number of synapses and are less than 5 (Fig. 4). Recognizing that greater synaptic connectivity facilitates stronger signal propagation across neurons [16], we focus on larger synaptic connectivity, as defined above, as indicative of meaningful connections within the overall circuit. To objectively determine these higher degrees of connectivity, we employ Z-score analysis by taking a single neuron, listing out all its neuronal partners along with their associated number of synapses, and simply calculating its statistical Z-score. For clarity, a standard statistical Z-score is a measure that tells you how many standard deviations a data point is from the mean. It's a dimensionless quantity used to indicate the signed, fractional, number of standard deviations

by which an event is above the mean value being measured. The formula for calculating a Z-score is:

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

where  $x$  is the raw score (the data point in question), and  $\mu$  is the population mean, and  $\sigma$  is the population standard deviation.

By examining Z-scores for individual neurons in relation to their pre- or post-synaptic partners, we identify a limited set of neurons with notably high Z-scores. Figure 4 shows a right skewed histogram where most of the synaptic connectivity is on the left side of the curve, with higher synaptic connectivity skewed to the right. To refine our investigation, we filter out synaptic connections below an arbitrary Z-score threshold, resulting in a significantly reduced set of meaningful connections.



**Fig. 4.** A histogram of synaptic connectivity with all its related postsynaptic partner neurons. The right skewedness demonstrates that most synaptic innervation occurs with only a small number of synapses. The smallest number of postsynaptic neurons represent a very small percentage of the entire distribution.

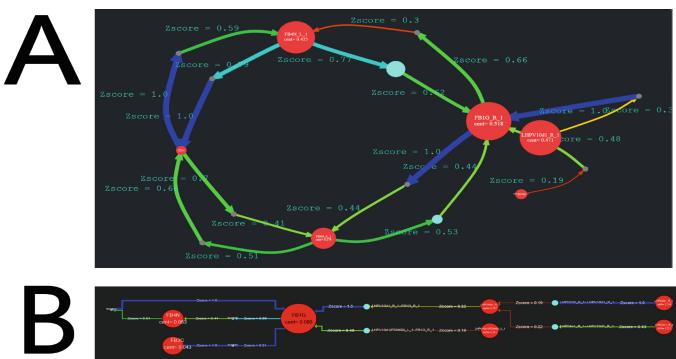
Z-scores are not uniformly distributed across the connectome making the standard calculation hard to compare between neuron pairs. To overcome this, we normalize the Z-scores (a number between zero and one) to facilitate comparison between neurons and draw initial conclusions about their pre- or post-synaptic connection and their proportional contribution to a neuron's overall connectivity. The normalization process involves the following formula:

$$Z_{\text{normalized}} = 1 - \frac{Z_{\max} - Z}{Z_{\max}} \quad (2)$$

### 2.3 Graph Based Analysis

Normalized Z-score calculations are integrated into our graph analysis method for visual utility. Given the Hemibrain dataset's vast neuron and synapse count,

bidirectional graphs are crucial for understanding complex relations [17]. We use Python packages like NetworkX and Graphviz to construct these graphs. They incorporate interneuron Z-score weights into edge descriptions, providing a nuanced view of synaptic strength. Distinct colors for pre- and post-synaptic connections enhance discernment between strong and weak links. Z-score weights indicate connection significance. Figure 5 shows four Z-score combinations: strong-strong; strong-weak; weak-strong; weak-weak. Colors range from red and yellow indicate weaker connections (Z-scores  $\leq 0.5$ ) to green and blue for stronger ones (Z-scores  $\geq 0.5$ ).



**Fig. 5.** Graphs of specific circuits within the fruit fly connectome. **a** shows a directional graph plot using normalized Z-scores to identify the weight of synaptic connectivity between neurons represented by the red nodes. Red edges indicate a low Z-score, while dark blue edges are the strongest. The size of nodes indicates centrality calculations of the red neuron nodes. **b** an example of a hierarchical layout using yED software. (Color figure online)

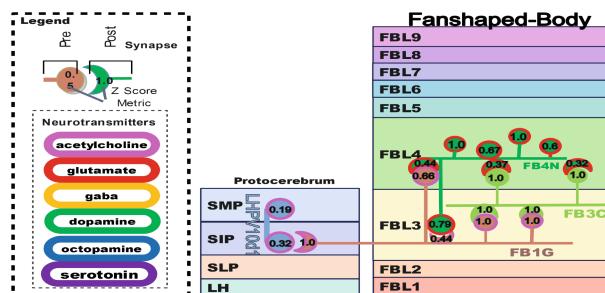
Z-score colored bidirectional graphs present necessary insights into the hierarchical organization of neural circuits, which in turn shed light on the functional architecture we wish to simulate in later steps. But as the number of neurons and synapses increases, maintaining readability becomes a challenge. To address this, we as a last step convert the data from Graphviz's native format to GraphML—a widely used format that facilitates exchange with other software packages and employ yED, a tool developed by yWorks GmbH, which significantly enhances the visual inspection of hierarchical layouts.

Figure 5 illustrates an example graph created using the yED software package where we simplify the model by restricting the number of edges to a single neuron type. Specifically, we group neurons with similar root names—for instance, neurons labeled as FB1G\_R\_1 are interpreted as part of the broader FB1G category. By doing so we generate graphs where each edge represents a neuron type rather than an individual neuron.

## 2.4 Circuit Modeling

After graph generation, we construct an end-to-end circuit model, as shown in Fig. 6. This model, derived from graphs tracing primary inputs from the Lateral Horn and Mushroom Body, is qualitative due to the high node and edge count. Despite some fidelity loss for readability, we gain insights and context within our model organism's biological framework. We incorporate information about the source and target neuropils where neurons arborize, enriching our understanding of connectivity patterns. Instead of color-coding axons based on Z-scores, we label Z-score values at synaptic boutons and spikes, allowing direct assessment of synaptic junction connection strength. Our convention of color-coding neurons by target neuropils aids pathway identification. While color choices may seem arbitrary, they enhance the connectome's visual accessibility.

In the Graph Analysis step, we determined NOIs based on four pre- and post-synaptic Z-score categories: strong-strong, strong-weak, weak-strong, and weak-weak. It's crucial to eliminate connections that don't support the hypothesis, but this isn't always straightforward. Thus, we decided to exclude the weak-weak connections, and this left us with a circuit as seen in Fig. 6, FB1G and FB4N. With two NOIs decided upon, it is possible to move onto the next step of the Framework and test for evidence of oscillatory characteristics between the excitatory neuron FB1G and its postsynaptic inhibitory partner FB4N.



**Fig. 6.** A Circuit Diagram of innervation between the CX and select neuropils in the fruit fly. Each colored area denotes a unique neuropil, with important sources of signal coming from the Superior Protocerebrum. The Fan-shaped Body is where incoming signals are processed.

## 2.5 Simulation for Neurons of Interest

To analyze the characteristics of a coupled oscillator network, it is necessary as a first step to simulate the frequency output of our NOI neural network (NOIs). For this model we employ a Leaky Integrate-and-Fire (LIF) model. The LIF model is a fundamental concept in computational neuroscience that describes in a computationally tractable way how neurons accumulate incoming synaptic

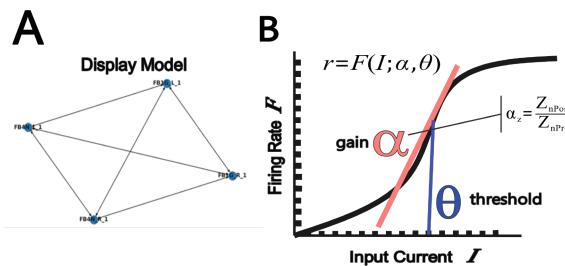
currents over time, incorporating a leak factor, until a threshold is reached, triggering the firing of an action potential [18]. This model is widely used in studying the spiking dynamics of neural networks under various conditions such as stimuli, tasks, or dynamic network states [19]. The basic equation for the LIF model is:

$$\tau_m \frac{dV}{dt} = -(V - E_L) + \frac{I}{g_L} \quad (3)$$

where:  $V$  is the membrane potential,  $g$  is the leak conductance,  $E_L$  is the resting potential,  $I$  is the external input current, and  $\tau_m$  is the membrane time constant.

We've identified two NOIs, FB1G and FB4N, using literature, pathfinding analysis, graph analysis, and circuit modeling. We aim to simulate these within the FlybrainLab environment using the Nengo package. However, our methodology isn't exclusive to Nengo and can adapt to any neural simulation package. The process begins by importing the NOIs into the FlybrainLab workspace (Fig. 2-A), followed by autonomous generation of necessary nodes and edges for simulations.

Before proceeding, it's important to note that unlike the simplifications in the Pathfinding and Graph sections, we use all available neurons in each group to construct Nengo ensembles. Our model isn't limited by neuron and synapse count, but it does restrict the simulation to one excitatory and one inhibitory neuron type, limiting the model to two dimensions. This facilitates Wilson-Cowan analysis, which treats many neurons within a similar ensemble as a single unit constrained to excitatory/inhibitory types. This supports Phase-Plane analysis for visualizing network stability and avoids the need for compactification of extra data dimensions, which could obscure results [20]. Figure 7-A shows the output topology of our model's two neuron types.



**Fig. 7.** Details about the Nengo model. **a** Four node simplified version of the Nengo model graph. FB1G is cholinergic and FB4N being Glutamatergic which creates a four-way node integration. **b** Parameters of the FI curve sigmoid function, alpha and theta. Theta shifts the curve along the x-axis and is typically estimated. The alpha parameter influences the steepness of the curve and is derived from the product of Z-scores.

With the above constraints in place our approach lies in establishing ensemble connections using a transfer function that mirrors, as accurately as possible, the biophysical parameters of our selected neurons. To do this we employ two key parameters in this process, alpha and theta, that are used within the Frequency-Current sigmoid transfer functions (F-I curve), a key component in artificial neural networks (i.e. Nengo), crucial for simulating the nonlinear behavior of neuron ensembles. The F-I curve in neuroscience illustrates the relationship between a neuron's firing rate (F) and the net synaptic current (I) it receives [28]

$$F(I; \alpha, \theta) = \frac{1}{1 + e^{-\alpha(I-\theta)}} - \frac{1}{1 + e^{\alpha\theta}} \quad (4)$$

where:  $I$  is the simulated input to the function, an abstract value of current,  $\theta$  determines the position of maximum slope before the sigmoid curve inverts toward its upper limit,  $\alpha$  alpha determines the value of the overall slope between minimum and maximum. The theta value used within the sigmoid function is challenging to determine and is best fitted against biological assays or backward engineered via deep learning algorithms. Sadly, since we don't have such data at this time, the number must simply be estimated to a value between two random integers. The alpha function we use is novel, and is derived from the Z-score connection weights using the following formula:

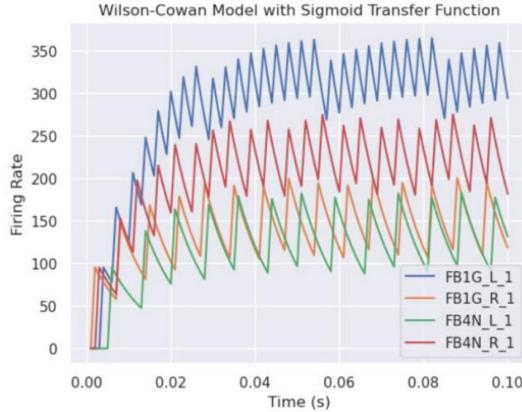
$$\alpha = \begin{cases} \frac{Z_{nPost}}{Z_{nPre}} & \text{if } \frac{Z_{nPost}}{Z_{nPre}} > 1 \\ \frac{Z_{nPost}}{Z_{nPre}} + 1 & \text{if } \frac{Z_{nPost}}{Z_{nPre}} < 1 \end{cases} \quad (5)$$

The product of pre- and post-synaptic Z-scores provides an analog for the alpha value in the sigmoid transfer function, representing gain and loss. Figure 7-B clarifies alpha's role in gain within this function. After configuring transfer functions and establishing neuron connections in Nengo, we input a simulated 10 Hz sine function connected to the FB1G neurons. This choice is informed by the Hemibrain dataset's pre- and post-synaptic directional vector and confirmed by graph analysis. The result is a graph showing the Hz frequency firing rate over time, with each neuron color-coded for identification. Figure 8 is a sample output produced by the Nengo Simulation. A key aspect to observe here is the consistent rise and maintenance of a steady firing rate by the neurons throughout the simulation.

In line with the Wilson-Cowan theory, one should anticipate that all firing rates will reach stable points if they exist. This expectation is based on the theory's premise that neurons within an excitatory-inhibitory network tend to stabilize their firing rates over time [29].

## 2.6 Analysis of Simulated Neural Signals

Coupled oscillator theory, commonly used for signal analysis, studies the dynamics of biological neural network models [21]. Originating from Huygens' work, the theory has evolved and been applied to both biological and non-biological systems. Synchronous rhythms are key in coordinating neural activity across brain



**Fig. 8.** Simulated firing rates of four neuron type ensembles, The Y axis represents an abstract Hz rate which must be converted into values that make sense from a biological example.

networks [22]. The rhythmic activities of various systems, like cardiac pacemaker cells, neural circuits of central pattern generators, and circadian cells, exemplify pulse-coupled oscillators in both invertebrates and vertebrates.

After generating a stable signal with Nengo, the crucial final step is signal analysis. We use the Wilson-Cowan theory and a two-dimensional phase plane analysis. This approach is ideal for our study as it helps understand Loosely Coupled Oscillators. For a detailed discussion on Loosely Coupled Oscillators, please refer to the discussion section. The model is defined by a system of two first-order, nonlinear differential equations:

$$\begin{aligned} \frac{dE}{dt} &= -E + F(w_{EE}E - w_{EI}I + I_E) \\ \frac{dI}{dt} &= -I + F(w_{IE}E - w_{II}I + I_I) \end{aligned} \quad (6)$$

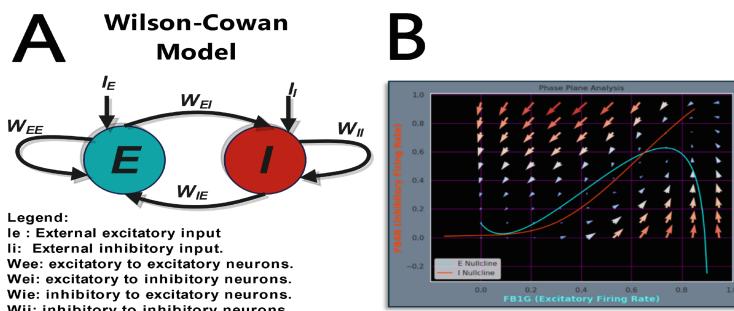
Here,  $E$  and  $I$  represent the average activity levels of the excitatory and inhibitory neurons, respectively. The parameters  $w_{EE}$ ,  $w_{EI}$ ,  $w_{IE}$  and  $w_{II}$  denote the synaptic weights, reflecting the strength of connections within and between the two types of neurons.  $I_E$  and  $I_I$  are external inputs to the excitatory and inhibitory populations, respectively. These weights are used in the simulation and nullclines phase plane analysis. Currently, the weights (i.e.,  $W_{EE}$ ,  $W_{EI}$ ,  $W_{IE}$ ,  $W_{II}$ ) must be estimated due to the lack of clear computational datasets for specific neuron combinations. The function  $F$  is a sigmoidal function that influences the input into a firing rate. Figure 9-A makes the integration of the Wilson-Cowan model easier to understand. The diagram's main goal is to depict the nullclines for both excitatory ( $E$ ) and inhibitory ( $I$ ) neurons, outlining the conditions that keep a neuron's activity constant over time.

### 3 Results

The parameters crucially shape neural network dynamics. Synaptic weights determine neuron influence, and external inputs can drive the network into various states. Understanding these parameters is key to interpreting the Wilson-Cowan model and the represented neural networks.

We used the Nengo simulation engine to model two neuron ensembles, FB1G and FB4N, within the fruit fly's FB. These neurons were identified as computationally significant through literature review, pathfinding techniques, and advanced modeling. They form a major part of the pathway connecting the FB to the Lateral Horn, an important neuropil in arousal [15, 23]. The simulation produced spiking neurons with stable firing rates over time, dynamically representing neural activity in the fruit fly's FB. Further analysis using 2D phase plane analysis revealed at least one stable point for each neuron type, approximately 0.61 for FB4N and 0.6 for FB1G. This suggests the presence of stable coupled oscillators within the network. This key finding allows for future comparisons with in-vivo neuron recordings in the fruit fly CX, enhancing our understanding of simulation parameters that accurately reflect individual neuron biophysics, contributing to computational neuroscience.

Figure 9-B shows the resulting phase plane diagram indicating two nullclines following an expected pattern, with at least one intersecting point at roughly 0.7 and 0.6 for FB1G and FB4N respectively. These intersection points, also known as fixed points, can be either stable or unstable, depending on the local behavior of the system. In our case, calculations reveal one stable point, as confirmed by examining the flow directions of all possible (X, Y) values within the grid.



**Fig. 9.** The resulting phase plane model from the simulated results above. X and Y axis represent abstract frequencies produced by the simulated neurons. The two nullclines, FB1G and FB4N respectively, converge at the stable point. Arrows show the flow of any signal that is not located on the two nullcline.

This result demonstrates that we fail to prove the null hypothesis and that it is not impossible for the two NOIs to form an oscillatory point within the CX of the fruit fly. A key next step would be to compare these results to actual

recordings of the CX and Fan-shaped body. Doing so would clearly be relevant to important questions such the neuromechanistic explanation of cognitive functions within the CX and their relation to other brain theories such functions to Bayesian brain model.

## 4 Discussion

It's not surprising to observe stable points in the frequencies. More accurately, the simulations don't disprove the null hypothesis that the CX contains no stable loosely coupled oscillators. This observation raises questions and insights that could bolster the hypothesis that loosely coupled oscillators form a natural construct for fundamental control within the fruit fly brain. This hypothesis aligns with scientific reasoning. Similar concepts are seen in PI Controllers used as error correction systems, like in homeostatic regulation of body temperature, where the error (temperature difference) is minimized over time. This raises the question of the ubiquity of these control mechanisms in regulation of neural circuits.

The Free Energy Principle is one such control mechanism. It hypothesizes that biological systems, such as the brain, minimize free energy - a measure of surprise or prediction error - to maintain homeostasis [24–26]. This principle has been applied to explain various biological phenomena, including perception, learning, action control, and behavior. We propose that identified loosely coupled oscillators could provide a necessary control mechanism. This potentially offers insight into common cognitive abilities across species and explains the neuromechanical mechanism for affective systems. Damasio and Solms theorize that affective systems are foundational to cognition [27], supported by Fabbro and Panksepp's work suggesting these systems underpin self-awareness and perception [28]. If affective systems are essential for cognition, their presence in the CX aligns with theories locating such systems in the basal ganglia, analogous to the CX in higher organisms [29]. Strausfeld strengthens this view by showing links between the CX and basal ganglia via an evolutionarily conserved genetic program, suggesting shared developmental and functional principles across species [29]. While human comparisons are challenging, they're not impossible. Solms, Panksepp, and Northoff propose that the periaqueductal grey (PAG) within the human basal ganglia is a crucial hub for affective/emotional-motor integration [27, 30].

Our study shows that loosely coupled oscillators could be a mechanism in the CX to regulate previously identified basic affective behaviors [23]. If similar motifs and circuits are conserved across species, simulations testable against experimental data could illuminate traditionally challenging subjects. While simulations and phase plane analysis offer valuable insights, they also identify areas for further exploration and experimental verification.

## 5 Data and Code Availability

Code associated with this research is available on the GitHub repository at, <https://github.com/sgarnell/ICONIP24>.

The code relies on several Python packages and includes: math, re, numpy, scipy, IPython, matplotlib, graphviz, pydot, networkx, neuroarch, nengo, tensorflow, nengo\_dl, nengo\_extras.

The code also utilizes the FlybrainLab software package available at, <https://github.com/FlyBrainLab>.

The research is based on the Hemibrain 1.1 and Flywire 1.2 datasets, including raw, processed, and intermediate data. Contact the corresponding author for access to the code and data. Consistent package versions are recommended for reproducibility. Feel free to use our code and data in your research, and don't hesitate to reach out with any questions or feedback. If you use our work, please cite our paper.

**Acknowledgments.** We thank the FFBO team at Columbia University for helpful advice using the FlybrainLab software platform.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

- Young, J., Armstrong, J.: Structure of the adult central complex in Drosophila: organization of distinct neuronal subsets. *J. Comp. Neurol.* **518**, 1500–1524 (2010)
- Sullivan, L.F.: Rewiring the drosophila brain with genetic manipulations in neural lineages. *Front. Mol. Neurosci.* **12**, 82 (2019)
- Frye, M., Dickinson, M.: Motor output reflects the linear superposition of visual and olfactory inputs in Drosophila. *J. Exp. Biol.* **207**, 123–131 (2004)
- Gray, C.M.: Synchronous oscillations in neuronal systems: mechanisms and functions. *J. Comput. Neurosci.* **1**(1), 11–38 (1994)
- Yap, M.: Oscillatory brain activity in spontaneous and induced sleep stages in flies. *Nat. Commun.* **8**(1), 1815 (2017)
- He, L.: Rapid adaptation to elevated extracellular potassium in the pyloric circuit of the crab, cancer borealis. *J. Neurophysiol.* **123**, 2075–2089 (2020)
- Loverso, P., Wachter, C., Cui, F.: Cross-species transcriptomic comparison of In Vitro and In Vivo mammalian neural cells. *Bioinform. Biol. Insights* **9**, S33124 (2015)
- Raji, J., Potter, C.: The number of neurons in drosophila and mosquito brains. *PLoS ONE* **16**, e0250381 (2021)
- Hulse, B.: A connectome of the drosophila central complex reveals network motifs suitable for flexible navigation and context-dependent action selection. *elife.* **10**, e66039 (2021)
- Scheffer, L., Meinertzhagen, I.: A connectome is not enough -what is still needed to understand the brain of Drosophila? *J. Exp. Biol.* **224** (2021)
- Chen, K.F.: Neurocalcin regulates nighttime sleep and arousal in drosophila. *elife.* **8**, e38114 (2019)

12. Jaimes, N.: Using biologically hierarchical modular architecture for explainable (2023)
13. Lazar, A.: Accelerating with flybrainlab the discovery of the functional logic of the drosophila brain in the connectomic and synaptomic era
14. Lazar A, Türkcan, M., Zhou, Y.: A programmable ontology encompassing the functional logic of the Drosophila brain (2021)
15. Bates, A.: Complete connectomic reconstruction of olfactory projection neurons in the fly brain. *Curr. Biol.* **30**, 3183–3199 (2020)
16. Scheffer, L.: A connectome and analysis of the adult drosophila central brain
17. Fornito, A., Zalesky, A., Breakspear, M.: Graph analysis of the human connectome: promise, progress, and pitfalls. *Neuroimage* **80**, 426–444 (2013)
18. Burkitt, N.: A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biol. Cybern.* **95**, 1–19 (2006)
19. Mazzoni, A.: Computing the local field potential (LFP) from integrate-and-fire network models. *PLoS Comput. Biol.* **11**, e1004584 (2015)
20. Xie, H.: Comparison among dimensionality reduction techniques based on random projection for cancer classification. *Comput. Biol. Chem.* **65**, 165–172 (2016)
21. Ashwin, P., Coombes, S., Nicks, R.: Mathematical frameworks for oscillatory network dynamics in neuroscience. *J. Math. Neurosci.* (2016)
22. Wang, X.: Neurophysiological and computational principles of cortical rhythms in cognition. *Physiol. Rev.* **90**, 1195–1268 (2010)
23. Huoviala, P.: Neural circuit basis of aversive Odour processing in Drosophila from sensory input to descending output (2018)
24. Friston, K., Costa, L., Parr, T.: Some interesting observations on the free energy principle. *Entropy (Basel)* **23** (2021)
25. Buckley, C.: The free energy principle for action and perception: a mathematical review. *J. Math. Psychol.* **81**, 55–79 (2017)
26. Bruineberg, J., Kiverstein, J., Rietveld, E.: The anticipating brain is not a scientist: the free-energy principle from an ecological-enactive perspective. *synthese* **195**, 2417–2444 (2016)
27. Solms, M.: The Hidden Spring: A Journey to the Source of Consciousness. W. W. Norton Company (2021)
28. Fabbro, F.: Evolutionary aspects of self-and world consciousness in vertebrates. *Front. Human Neurosci.* (2015)
29. Strausfeld, N., Hirth, F.: Deep homology of arthropod central complex and vertebrate basal ganglia. *Science* **340**, 157–161 (2013)
30. Solms, M.: What are affects? *50*, 485–522 (1996)



# EL-LSTM: A Multivariate Time Series Forecasting Model Combining Spiking Neurons and Long Short-Term Memory Networks

Lei Yang<sup>(✉)</sup> , Yuhan Jiang , Kaixin Wang , Pinjie Zhao, and Kangshun Li

South China Agricultural University, Guangzhou 510642, China  
yanglei\_s@scau.edu.cn

**Abstract.** This research aims to address critical challenges in multivariate time series forecasting: capturing complex temporal patterns and overcoming the exponential decay issue in the long-term memory of Long Short-Term Memory (LSTM) models. To address these, we propose a new deep learning model called EL-LSTM. The model combines a simplified Leaky Integrate-and-Fire (LIF) neuron model with LSTM and significantly enhances the model's ability to capture complex temporal dependencies through local and global point attention mechanisms. The EL-LSTM model can effectively retain key information. We have validated the effectiveness of the model in multivariate time series prediction tasks, especially in the fields of finance and traffic flow forecasting. Experimental results show that the EL-LSTM model has achieved significant improvements in forecasting accuracy, particularly when dealing with data in concept-drift environments. This study offers a new perspective for the field of time series prediction and demonstrates the potential for handling complex time series data in practical applications.

**Keywords:** Long Short-Term Memory · Multivariate Time Series Forecasting · Attention Mechanism · Leaky Integrate-and-Fire neuron

## 1 Introduction

Prediction [1,2] is a method that utilizes temporal data, analyzing historical data to predict future trends, thereby assisting decision-making. For example, forecasting urban traffic flow [3,4] can help transportation departments improve operational efficiency and is an essential part of urban intelligence. Weather forecasting [5,6], such as rainfall and wind speed predictions, can notify people in advance to make necessary preparations and allow relevant departments to respond proactively to extreme weather events like floods and droughts. Forecasting urban electricity consumption [7] allows power grid companies to adjust energy distribution in advance, and power plants can determine the storage and usage of coal, thus improving resource utilization and reducing energy loss. Stock

forecasting [8,9] results can serve as a reference for investors, assisting them in making informed investment decisions. For fresh products, accurate predictions of inventory and sales volumes are directly related to a company's profitability. Factors that may affect the accuracy of predictions include non-stationarity of data, the impact of random noise, the influence of random events [2], or simply insufficient data [3].

The application of deep learning models to time series forecasting has gradually emerged [8,9]. The earliest was based on the basic Multilayer Perceptron model [10]. It maps sequence data to the prediction of target values by introducing multiple input features and multiple output nodes. However, this simple model ignores the temporal dependencies in sequence data. Then came the traditional Recurrent Neural Networks (RNN) [11–16], which played a significant role in multivariate time series forecasting.

In recent years, several good models have emerged that combine attention mechanisms with neural networks [17–19]. These models can better handle the relationship between long-term dependencies and important time steps, thereby improving the performance of time series forecasting. Examples include MS-Attn [20], DSTP-RNN [4], and HSN-LSTM [21] models.

The contributions of this paper can be summarized as follows:

(1) There is an issue of exponential decay in long-term memory within traditional LSTM models. To address this problem, the EL-LSTM model employs a Leaky Integrate-and-Fire (LIF) neuron model and embeds it into the LSTM's candidate memory gate (g gate). The LIF model simulates the firing behavior of neurons through the accumulation and decay of membrane potential, relying not on complex gating mechanisms but on a simple threshold to determine whether to emit a spike. This mechanism allows the LIF model to update, rather than simply reset, the membrane potential at each time step. As a result, the model is capable of retaining a greater amount of historical information within a localized range, thereby enhancing its capacity for long-term memory.

(2) The study further enhances the model's performance by implementing a dual attention mechanism that includes local-point attention and global-point attention. The local-point attention mechanism focuses on capturing local features and short-term dependencies within each time step of the sequence. The global-point attention mechanism is responsible for capturing the global dependencies between all time steps in the sequence. By combining the use of these dual attention mechanisms, the EL-LSTM model can learn time series data from both local and global perspectives simultaneously.

(3) Through theoretical analysis, we have introduced ReLU activation functions and normalization to optimize the model's learning of key time series features. In the environment of concept drift, the distribution of data changes over time. The EL-LSTM model's setup for dynamic adjustment of features, as well as the selection of activation functions and normalization, are referred to as Dynamic Adjustment of Features.

## 2 Related Work

### 2.1 Spiking Neural Networks

Spiking Neural Networks (SNNs) [22–26], taking cues from the brain’s circuitry, represent a class of neuromorph computing models. In SNN models, each neuron adjusts its membrane potential in response to its stored state and incoming signals, emitting an action potential once the potential crosses a predefined threshold. SNNs carry both spatial and temporal information [27–30]. Theoretical [31] and empirical [26] studies have demonstrated that SNNs have better computational potential in comparison with Artificial Neural Networks.

Gerum et al. [32] explored how to adjust the parameters of LSTM units to exhibit dynamic behaviors similar to those of LIF neurons. Wu et al. [33] proposed a novel LIAF neuron model and constructed a deep network, LIAF-Net, based on this model. The LIAF model combines the temporal processing characteristics of LIF with the spatial processing capabilities of artificial neural networks, enabling it to more effectively handle spatiotemporal information. Gerum et al. focused on achieving biological plausibility within the existing LSTM framework, without explicitly proposing the integration of spiking neurons and LSTM to enhance the performance of time series prediction. The design focus of LIAF-Net is on improving computational efficiency and simplifying network architecture, rather than specifically addressing long-term dependencies and concept drift issues in time series prediction. Overall, the EL-LSTM model is specifically designed to meet the needs of time series prediction, by leveraging the strengths of both spiking neurons and LSTM, as well as introducing attention mechanisms and concept drift adaptation strategies, to enhance the model’s performance on complex time series data.

### 2.2 Self-Attention Module

The working principle of the self-attention mechanism involves calculating the similarity scores between each position in the sequence and all other positions. These scores, which are based on the relationships between positions, are transformed into weights that are used to aggregate information within the sequence. This process enables the model to focus on the most critical parts of the sequence, capturing key temporal dependency features. In this study, the self-attention mechanism has been modified to play a corresponding role at local points, thereby better adjusting the attention scores on the global sequence. Both types of attention mechanisms are based on the self-attention mechanism.

**Local-Point Attention:** Local-Point Attention focuses on the feature relationships within a single time step. By limiting the scope of the attention mechanism to a local area, the model can process local patterns in detail.

**Global-Point Attention:** Unlike Local-Point Attention, Global-Point Attention looks at the entire sequence to capture long-term dependencies that span multiple time steps. This mechanism allows the model to integrate information across the entire dataset, identify long-term trends and patterns, and provide a broad perspective for forecasting.

By combining both attention mechanisms, our model is able to learn time series data from both local and global perspectives simultaneously.

### 2.3 Concept Drift Environment

Concept drift [34] is a type of in non-stationary data, focusing more on the statistical properties of the target variable changing over time. In practical applications, data distributions may shift due to changes in time, environment, or user behavior. Such shifts can cause well-trained models to perform poorly on new data because they have not adapted to the changes in data distribution or relationships.

There are different types of concept drift environments. One is sudden drift, where changes in the data distribution occur at a specific point in time, and models need to quickly adapt to the new data distribution. Another is gradual drift, where the relationships among data change slowly over time, and models need to continually adapt to new relationships. There are also progressive drift, cyclical drift, and composite drift environments.

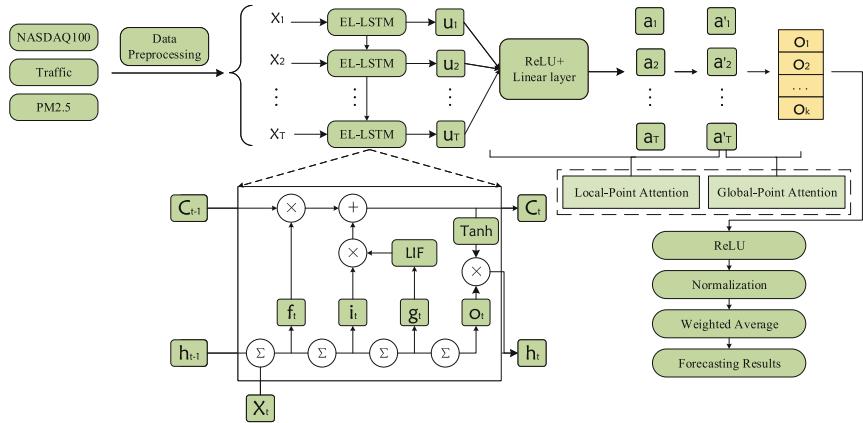
Dealing with concept drift environments is a challenging issue because models need to adapt to new data distributions or relationships in a timely manner to maintain accuracy and performance. Methods for handling concept drift typically include Drift Detection, Drift Understanding, and Drift Adaptation. This study mitigates the adverse effects of concept drift through drift adaptation methods.

## 3 EL-LSTM Model

### 3.1 Overall Architecture

The paper combines spiking neurons with LSTM (see Fig. 1), leveraging the advantage of spiking neurons in long-term memory while retaining the effectiveness of LSTM in sequence modeling. This helps to enhance the model’s ability to capture long-term dependencies. Specifically, the LIF (Leaky Integrate-and-Fire) model maintains memory of past information by updating the membrane potential at each time step. This update is achieved through a decay factor controlled by  $\tau_m$ , allowing the membrane potential to decay gradually rather than being reset abruptly. This decay mechanism enables the LIF model to retain historical information for a longer period because the rate of information loss is slower. A longer  $\tau_m$  means that the decay of the membrane potential is slower, which helps the model maintain memory of historical information over a longer time range.

Moreover, in this experiment, the threshold judgment has been decoupled, allowing researchers to flexibly modify the threshold determination and action potential generation mechanisms according to the specific needs of the task. The separated threshold judgment mechanism can adaptively adjust the firing behavior of neurons based on the dynamic changes of the input signal. This adaptability enables the LIF model to better simulate the activity of neurons



**Fig. 1.** EL-LSTM Model

in biological nervous systems, enhancing the model's responsiveness to environmental changes.

Additionally, this paper comprehensively considers aspects such as model architecture, attention mechanisms, integration of external information, data preprocessing, loss function selection, parameter tuning and training strategies, model evaluation and comparison, as well as hyperparameter optimization. Through rational design and experimental validation, the performance and effectiveness of the model in multivariate time series forecasting tasks can be enhanced.

During the training and optimization process of the model, we have employed the following techniques: Adam Optimization, Einstein Summation, ReLU and Linear Layer, Normalization, and Weighted Average. By combining the use of these techniques, our model has demonstrated excellent performance in handling complex sequence data, effectively enhancing its ability to capture long-term dependencies and overall forecasting accuracy.

### 3.2 Spiking Neural Network

The time constants, denoted as  $\tau_m$  and  $\tau_a$ , serve as the basis for determining the dynamic attenuation coefficients, which are represented by  $\alpha$  and  $\beta$  as follows.

$$\alpha = \exp(-k * dt * u_{t-1} / \tau_m) \quad (1)$$

$$\beta = \exp(-k * dt * s_{t-1} / \tau_a) \quad (2)$$

In this context, the parameter  $k$  denotes a constant blend ratio of the neuronal membrane potential and the spiking activity from preceding time steps. The variables  $u_{t-1}$  and  $s_{t-1}$  are indicative of the potential difference across the neuronal membrane and the neural activity signals one step prior, correspondingly. Additionally, the time constants are specified as  $\tau_m$  equals 4 and  $\tau_a$  equals 32.

The process for calculating the membrane potential and updating the spike information peak value is as follows:

$$\gamma_t = \beta \cdot \gamma_{t-1} + s_{t-1} \odot (1 - \beta) \quad (3)$$

$$\theta = c_0 + d_0 \gamma_t \quad (4)$$

$$u_t = \alpha \odot u_{t-1} + R_m(1 - \alpha) \odot I_t - dt \cdot s_{t-1} \odot \theta \quad (5)$$

$$K_t = u_t - \theta \quad (6)$$

$$s_t = T(K_t \geq 0.5), \quad (7)$$

Here,  $\theta$  represents a dynamic threshold (ranging from [0.22; 0.56]). The membrane potential  $u_t$  is updated and subtracted by the dynamic threshold  $\theta$ . When the value is positive, the neuron generates a firing signal.  $\gamma_t$  and  $\theta$  control the variation of the dynamic threshold, making the neuron's firing behavior more complex and adaptive. When updating spike information, if the condition within the parentheses holds true, it is 1.0, otherwise, it is zero. Ultimately, the SNN module's ultimate output value is derived from the formula presented in Equation (7).

### 3.3 Spiking Neuron Embedding in LSTM

Specifically, The paper designs specialized structures and mechanisms [35] that allow for the embedding of spiking neurons into the LSTM model, enhancing its ability to retain a longer memory. In this way, the model can better utilize past input information and pass it on to subsequent time steps, enhancing the modeling capability of long-term dependencies.

To achieve multi-level forecasting, this paper requires a cyclic repetition of the following computational process to obtain multiple outputs.

$$\begin{bmatrix} i_t \\ f_t \\ o_t \end{bmatrix} = \sigma(W \otimes h_{t-1} + \mathbf{U} \otimes \mathbf{x}_t + \mathbf{b}) \quad (8)$$

$$\tilde{\mathbf{g}}_t = \tanh(W \otimes h_{t-1} + \mathbf{U} \otimes \mathbf{x}_t + \mathbf{b}) \quad (9)$$

$$\tilde{\mathbf{g}}'_t = LIF(\mathbf{h}_{t-1}, \tilde{\mathbf{g}}_t, \mathbf{i}_t) \quad (10)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot c_{t-1} + i_t \odot \tilde{\mathbf{g}}'_t \quad (11)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(c_t) \quad (12)$$

Here,  $i_t$ ,  $f_t$ , and  $o_t$  denote the input, forget, and output gates, respectively. The cell state and the hidden state at timestep t are denoted by  $c_t$  and  $h_t$ , respectively. The functions  $\sigma$  and  $\tanh$  correspond to the sigmoid and hyperbolic tangent activations, respectively.

Related studies [4, 35] have discussed the role of the  $g_t$  gate as the activation of crucial candidate information.  $g_t$  calculates a new candidate memory content based on the current input and the memory content from the previous moment. This candidate's memory content is then used to update the memory content.

### 3.4 Local-Point Attention and Global-Point Attention Mechanisms

This paper has explored dual attention mechanisms [9], which have achieved certain effects under specific circumstances. However, when integrated with the EL-LSTM framework, the self-attention mechanism has demonstrated superior performance. The input sequence X has dimensions [L, B, D, N], where L is the sequence length, B is the batch size, D is the number of features at each position, and N is the dimension of each feature.

In Local-Point Attention, for each time step t and each batch b in the sequence, the local self-attention mechanism is calculated as follows: Generation of Query, Key, and Value:

$$Q_t = W^Q X_t, \quad K_t = W^K X_t, \quad V_t = W^V X_t \quad (13)$$

The terms  $W^Q$ ,  $W^K$  and  $W^V$  represent the learnable weight matrices.

$$S_t = \frac{Q_t K_t^T}{\sqrt{N}} \quad (14)$$

$$W_t = \text{softmax}(S_t) \quad (15)$$

$$b_{t,b} = W_t V_t \quad (16)$$

Residual Connection is used to combine the output  $b_{t,b}$  of the local self-attention with the original input  $Q_{t,b}$ :

$$Q'_{t,b} = Q_{t,b} + W b_{t,b} + b \quad (17)$$

In Global-Point Attention, the calculations are performed using the updated  $Q'$ , while K (Key) and V (Value) continue to use their original values:

$$S' = \frac{Q' K^T}{\sqrt{N}} \quad (18)$$

$$W' = \text{softmax}(S') \quad (19)$$

$$Z = W' V \quad (20)$$

In the process of calculating the final output, two different types of attention weights,  $\alpha$  and  $\beta$ , are involved.

Calculating attention weights  $\alpha_{t,b}$ : This step involves applying a linear transformation to the input tensor  $O$  (outputs), along with an additional bias term  $F_{\alpha_n}^b$ . The Einstein summation convention is used to simplify the notation of tensor multiplication. A *ReLU* (Rectified Linear Unit) function is then applied to the result to introduce nonlinearity and ensure that the attention weights are positive.

$$\alpha_{t,b} = \text{ReLU} \left( \sum_{i,j} O_{b,t,i,j} F_{\alpha_n,i,k} + F_{\alpha_n}^b \right) \quad (21)$$

Here,  $\alpha_{t,b}$  represents the attention weights at time step  $t$  and batch  $b$ .  $b$  stands for the batch size, indicating the number of samples in a batch.  $t$  represents the time step, referring to the point in time within the sequence data.  $i$  denotes the number of features at each position and  $j$  is the dimension of each feature.  $F_{\alpha_n, i, k}$  represents the attention weight matrix and  $F_{\alpha_n}^b$  is the bias term.

The attention weights need to be normalized to ensure that the sum of the weights across all time steps and batches equals 1.

$$\alpha_{t,b}^{\text{norm}} = \frac{\alpha_{t,b}}{\sum_{t=1}^T \sum_{b=1}^B \alpha_{t,b}} \quad (22)$$

Here,  $g_n$  is obtained by multiplying the normalized attention weights with their corresponding  $O$  (outputs) and then summing the results to get the weighted output.

$$g_n = \sum_{t=1}^T \sum_{b=1}^B \alpha_{t,b}^{\text{norm}} \cdot O_{t,b} \quad (23)$$

Similar to the computation of  $\alpha$ ,  $\beta$  is also calculated through a linear transformation followed by a ReLU activation function. It is then subjected to an exponentiation and normalization process to generate a valid probability distribution.

$$\mu = \Phi(h_g) \quad (24)$$

$$\beta = \text{ReLU}(F_\beta(h_g)) \quad (25)$$

$$\beta = \exp(\beta) \quad (26)$$

$$\beta = \frac{\beta}{\sum_{t=1}^T \sum_{b=1}^B \beta} \quad (27)$$

$$\text{mean} = \sum_{t=1}^T \sum_{b=1}^B \beta \cdot \mu \quad (28)$$

Here:  $\beta$  represents the normalized attention weights, which have been processed through exponentiation and normalization.  $\mu$  is the linear representation obtained through the  $\Phi$  transformation, which maps the concatenated features  $h_g$  to the output dimension. In this way, the mean represents the model's final output, which is obtained by computing a weighted sum of  $\beta$  and  $\mu$  across time steps and batches.

## 4 Experiments

### 4.1 Experimental Setup

We conducted experiments on the NASDAQ100, Traffic, and PM2.5 datasets, with specific settings as shown in Table 1.

The experiment utilized the EL-LSTM model for forecasting and set some key hyperparameters, including the dimensions of the hidden state, the number

**Table 1.** Datasets Statistics Information

Datasets	Samples	Columns	Interval	Time Window
NASDAQ100 [9]	40560	82	1 min	20
Traffic [36]	48204	9	1 h	6
PM2.5 [37]	43824	12	1 h	10

of EL-LSTM units, and the choice of activation function. The selection of these hyperparameters needed to be tuned according to the specific dataset and task to achieve optimal performance. Additionally, this paper set some hyperparameters for the training process, including the learning rate, number of iterations, and batch size. The choice of these hyperparameters was also determined through trials and tuning to achieve better convergence and generalization during the model training process.

In the experiment, the number of EL-LSTM units was set to 128. An increased number of units can enhance the model's learning capability. However, it also raises the model's complexity and training time. The standard deviation for parameter initialization was set to 0.02. A smaller standard deviation helps prevent issues of vanishing or exploding gradients during the initial training phase, facilitating faster model convergence.

In the NASDAQ100 dataset, alpha was set to 0.5, and beta to 0.4. In the PM2.5 dataset, alpha was set to 0.4, with beta at 0.5. In the Traffic dataset, alpha was set to 0.3, and beta to 0.6. These values for alpha and beta have been identified as optimal through rigorous tuning processes.

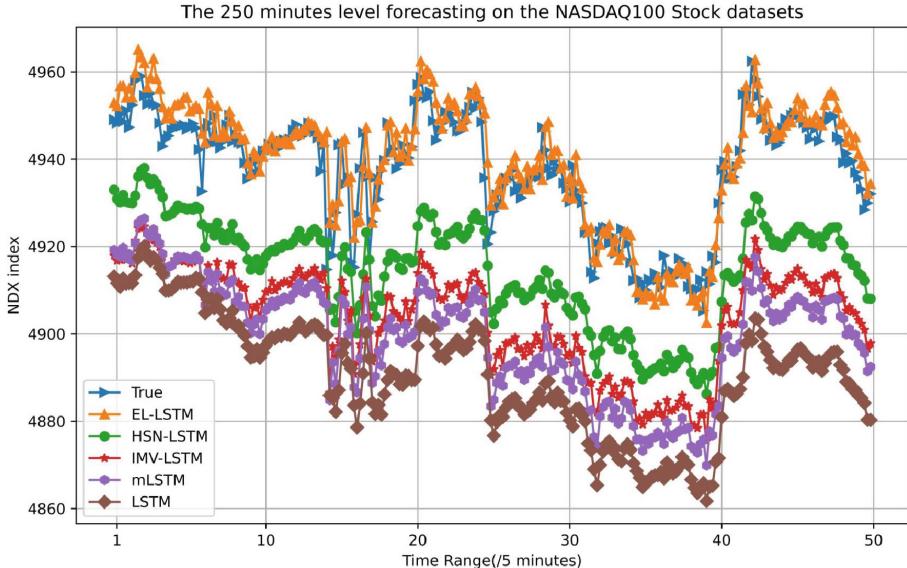
Furthermore, due to the sparse activation characteristic of the ReLU activation function in the Dynamic Adjustment of Features module, it aligns well with the EL-LSTM model. This adjustment reduces the training period of the model, allowing the parameter ‘patience’ of the early stopping strategy to be lowered from 140 to a range of 35 to 50, thereby significantly reducing the time cost of the model.

Finally, we introduced two metrics, Mean Squared Error and Mean Absolute Error [9, 20], to more comprehensively evaluate the model's predictive performance.

## 4.2 Experimental Results

To illustrate the influence of time windows and prediction horizons on the proposed EL-LSTM method and baseline, we conducted specific experiments. As shown in Fig. 2, with a time window set to 20 and a prediction horizon of 24, a comparison of the predicted values and actual values of five models on the test set within 250 min shows that the EL-LSTM model closely fits the actual values.

On the NASDAQ100 dataset, we conducted a series of ablation studies with the aim of assessing the contribution of key components of the EL-LSTM model to predictive performance by removing them. As shown in Table 2 and Fig. 3,



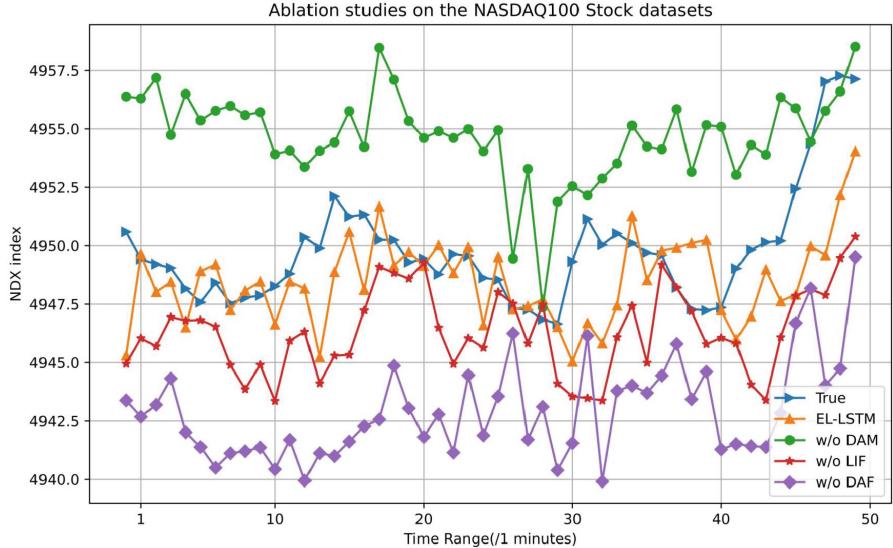
**Fig. 2.** Experimental Comparison of Baseline Models

we individually removed the Dual-Attention Mechanism, the Leaky Integrate-and-Fire module, and the Dynamic Adjustment of Features module.

**Table 2.** Ablation Studies on the NASDAQ100 Datasets

NASDAQ	EL-LSTM	w/o LIF	w/o DAM	w/o DAF
MAE	5.627	9.354	10.855	12.239
RMSE	6.947	11.416	12.946	15.253

As shown in Tables 3 and 4, on the NASDAQ100, PM2.5, and Traffic datasets, the EL-LSTM model was compared with eight other baseline models. These experiments indicate that the EL-LSTM model of this study outperforms other baselines in most cases. As the forecast time horizon expands, the performance of the EL-LSTM model shows a trend of decline to varying degrees. In contrast, it is noteworthy to highlight that the EL-LSTM demonstrates superior performance on the NASDAQ100 financial dataset. This finding can be attributed to the characteristics of financial data itself. Compared to general time series data, financial data contains more complex nonlinear and unpredictable patterns, which provide richer temporal dynamic information for the model's learning process. Therefore, although long-term forecasting is a challenge for any time series forecasting model, the EL-LSTM model, with its strong learning capability for time series, seems to show better adaptability and robustness when dealing with such



**Fig. 3.** Ablation Studies on the NASDAQ100 Datasets

**Table 3.** Performance of Baseline Models Across Various Time Windows

Models	Metrics	NASDAQ100				Traffic				PM2.5			
		tw=20	tw=24	tw=36	tw=48	tw=6	tw=24	tw=36	tw=48	tw=10	tw=24	tw=36	tw=48
EL-LSTM	MAE	5.627	5.435	5.910	5.623	255.916	246.438	240.956	246.883	11.377	11.418	11.527	11.395
	RMSE	6.947	6.987	7.458	7.010	373.458	363.521	362.257	368.050	20.127	20.677	20.681	20.659
IMV-LSTM[8]	MAE	12.600	8.398	11.340	16.847	258.011	253.350	261.368	253.670	11.707	12.815	13.305	12.921
	RMSE	13.519	10.876	11.894	17.492	375.177	375.492	385.777	379.616	21.293	21.657	21.856	21.815
HSN-LSTM[21]	MAE	10.162	8.593	18.409	21.543	258.955	267.749	257.800	264.373	11.419	11.691	11.577	12.222
	RMSE	11.209	11.053	22.147	25.661	380.207	387.983	374.559	386.989	20.771	20.912	20.651	21.458
MS-Attn[20]	MAE	17.018	13.064	19.303	22.157	267.454	449.148	495.044	471.392	12.964	24.281	25.283	26.342
	RMSE	22.921	14.124	20.178	34.268	394.239	712.742	723.558	690.068	24.164	39.025	39.605	40.249
LSTM[38]	MAE	20.059	20.457	24.586	22.905	256.195	258.244	247.490	255.814	11.533	11.775	11.782	11.481
	RMSE	20.381	20.855	25.075	23.340	376.191	378.873	366.744	371.215	20.806	20.740	20.817	20.746
GRU[39]	MAE	21.925	23.904	22.604	26.283	262.792	254.590	246.442	254.486	11.599	12.314	11.935	11.660
	RMSE	22.359	24.249	22.897	26.786	381.099	371.646	358.624	366.927	20.947	21.175	20.806	20.787
DA-RNN[9]	MAE	27.973	21.343	19.674	16.034	268.604	259.206	253.964	259.679	11.873	11.678	12.003	11.839
	RMSE	28.297	21.605	20.010	16.397	378.476	381.312	365.926	371.646	20.908	20.800	21.039	20.698
DeepAR[40]	MAE	22.747	23.085	29.014	30.805	257.345	263.026	247.101	248.875	11.400	11.454	11.672	11.485
	RMSE	23.184	23.500	29.363	31.119	371.150	381.367	367.878	359.012	20.756	20.497	20.818	20.692
mLSTM[41]	MAE	23.746	23.945	24.414	24.654	265.472	257.096	256.904	247.167	11.889	11.669	12.173	11.747
	RMSE	24.245	24.362	24.652	25.055	383.049	375.497	374.675	374.458	21.037	20.809	21.181	20.851

highly complex data. These findings highlight the potential application value of the EL-LSTM model in the field of financial time series forecasting and also provide a new perspective for future research directions.

**Table 4.** Performance of baseline models across different forecast horizons

Modols	Metrics	NASDAQ100				Traffic				PM2.5			
		h=3	h=6	h=12	h=24	h=1	h=6	h=12	h=24	h=1	h=6	h=12	h=24
EL-LSTM	MAE	5.627	5.505	6.261	9.904	255.916	578.069	883.737	1272.948	11.377	33.158	47.294	58.928
	RMSE	6.947	7.352	8.231	12.025	373.458	850.896	1221.173	1605.239	20.127	50.450	67.723	81.301
IMV-LSTM[8]	MAE	12.600	10.794	11.090	32.442	258.011	657.288	891.312	1325.982	11.707	34.371	50.308	62.407
	RMSE	13.519	11.877	12.474	33.566	375.177	940.593	1233.240	1669.562	21.293	52.201	70.352	84.589
HSN-LSTM[21]	MAE	10.162	8.706	9.122	16.980	258.955	657.001	905.515	1327.393	11.419	35.747	49.328	60.155
	RMSE	11.209	9.609	11.922	20.556	380.207	926.200	1242.096	1646.305	20.771	53.508	71.594	83.360
MS-Attn[20]	MAE	17.018	17.198	21.259	29.425	267.454	780.216	1037.188	1422.219	12.964	35.727	49.318	62.366
	RMSE	22.921	20.103	28.451	34.065	394.239	1089.962	1378.880	1746.372	24.164	53.973	71.119	83.357
LSTM[38]	MAE	20.059	35.203	44.502	52.898	256.195	602.556	925.491	1296.636	11.533	33.763	48.192	60.568
	RMSE	20.381	35.527	45.043	53.798	376.191	918.430	1235.459	1624.550	20.806	52.800	69.450	82.696
GRU[39]	MAE	21.925	29.412	37.467	47.983	262.792	646.305	908.677	1277.111	11.599	35.640	47.623	60.367
	RMSE	22.359	29.792	37.926	48.582	381.099	913.422	1247.988	1624.331	20.947	53.257	68.045	83.246
DA-RNN[9]	MAE	27.973	18.546	36.638	55.453	268.604	584.480	929.343	1282.255	11.873	34.137	47.199	60.342
	RMSE	28.297	19.051	37.284	55.969	378.476	889.209	1238.523	1627.795	20.908	52.969	68.705	82.563
DeepAR[40]	MAE	22.747	33.910	40.101	56.969	257.345	590.382	903.286	1308.477	11.400	36.213	47.297	59.817
	RMSE	23.184	34.235	40.501	57.626	371.150	887.573	1237.022	1646.919	20.756	53.780	68.833	82.610
mLSTM[41]	MAE	23.746	27.011	31.143	39.831	265.472	625.043	911.040	1285.566	11.889	35.095	48.479	60.372
	RMSE	24.245	27.622	31.625	40.560	383.049	889.827	1257.243	1624.198	21.037	53.179	69.587	81.819

The experimental results indicate that this study has achieved significant success on three real-world datasets. The EL-LSTM model, by integrating the characteristics of spiking neural networks and the strengths of long short-term memory networks, has demonstrated exceptional performance in multivariate time series forecasting tasks. Particularly when dealing with time series data that have long-term dependencies and are subject to concept drift, the EL-LSTM model is capable of effectively retaining key information and mitigating the adverse effects of concept drift. These experiments have confirmed the potential application of the EL-LSTM model in areas such as finance and traffic flow prediction, providing new directions and ideas for future time series forecasting research.

## 5 Conclusion

The study aim at tackling the difficulties of long-term dependency and concept drift in time series forecasting. By incorporating a simplified Leaky Integrate-and-Fire (LIF) model, dual attention mechanisms, and dynamic adjustment of feature, the EL-LSTM significantly enhances its learning capability and predictive accuracy for complex temporal patterns. Experimental results further confirm the effectiveness of the EL-LSTM model in handling time series data with long-term dependencies. The application of the model on multiple real-world datasets demonstrates its ability to adapt to environments with concept drift and maintain stable and accurate performance in long-term forecasting tasks.

Overall, the introduction of the EL-LSTM model contributes a new perspective and methodology to the field of time series forecasting. It exhibits distinctive

strengths, particularly in processing multivariate time series data that have long-term dependencies and concept drift, and its potential in practical applications is worth further exploration and development. Future research can continue to optimize the model structure, expand the range of applications, and explore integration with other advanced machine learning technologies to achieve more accurate and robust time series forecasting.

**Acknowledgements.** This work was partially supported by Natural Science Foundation of Guangdong Province of China (Grant No.2020A1515010691), Science and Technology Innovation Special Project of Guangdong Province (Grant No.SDZX 2023032), Agricultural Science and Technology Commissioner Project of Guangzhou City (Grant No.20212100036), National Key R&D Program Projects (Grant Nos.2023 YFF1105101 and 2023YFF1104201), and National Natural Science Foundation of China (Grant Nos.61573157 and 61703170).

## References

1. Deng, J., Chen, X., Jiang, R., Song, X., Tsang, I.W.: A multi-view multi-task learning framework for multi-variate time series forecasting. *IEEE Trans. Knowl. Data Eng.* **35**(8), 7665–7680 (2023). <https://doi.org/10.1109/TKDE.2022.3218803>
2. Zheng, W., Hu, J.: Multivariate time series prediction based on temporal change information learning method. *IEEE Trans. Neural Netw. Learn. Syst.* **34**(10), 7034–7048 (2023). <https://doi.org/10.1109/TNNLS.2021.3137178>
3. Wu, Z., et al.: Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD Conference Knowledge Discovery Data Mining, pp. 753–763 (2020). <https://doi.org/10.1145/3394486>
4. Trinh, N.P.A., Tran, K.N., Do, T.H.: Traffic flow forecasting using multivariate time-series deep learning and distributed computing. In: 2022 RIVF International Conference on Computing and Communication Technologies (RIVF), pp. 665–670. Ho Chi Minh City (2022). <https://doi.org/10.1109/RIVF55975.2022.10013796>
5. Verma, S.K., Gupta, A., Jyoti, A.: Stack layer and bidirectional layer long short - term memory (LSTM) time series model with intermediate variable for weather prediction. In: International Conference on Computational Performance Evaluation (ComPE), pp. 065–070. Shillong (2021). <https://doi.org/10.1109/ComPE53109.2021.9752357>
6. Ahmed, M., Abdelrazek, S., Kamalasadan, S., Enslin, J., Fenimore, T.: Weather forecasting based intelligent distribution feeder load prediction. In: IEEE Power and Energy Society General Meeting (PESGM), pp. 1–5. Boston (2016). <https://doi.org/10.1109/PESGM.2016.7741878>
7. Zlatkova, A., Velkovski, B., Kokolanski, Z., Taskovski, D.: Short-term energy forecasting for public educational institution. In: IEEE International Conference on Environment and Electrical Engineering and IEEE Industrial and Commercial Power Systems Europe (EEEIC / I and CPS Europe), pp. 1–6. Madrid (2023)
8. Mahmud, M., et al.: Deep learning in mining biological data. *Cogn. Comput.* **13**(1), 1–33 (2021)
9. Qin, Y., et al.: A dual-stage attention-based recurrent neural network for time series prediction. In: Proceedings 26th International Joint Conference Artificial Intelligence, pp. 2627–2633 (2017). <https://doi.org/10.48550/arXiv.1704.02971>

10. Widiasari, I.R., Nugroho, L.E., Widyawan: deep learning multilayer perceptron (MLP) for flood prediction model using wireless sensor network based hydrology time series data mining. In: International Conference on Innovative and Creative Information Technology (ICITech), pp. 1–5. Salatiga (2017). <https://doi.org/10.1109/INNOCIT.2017.8319150>
11. Satu, M.S., Rahman, S., Khan, M.I., Abedin, M.Z., Kaiser, M.S., Mahmud, M.: Towards improved detection of cognitive performance using bidirectional multilayer long-short term memory neural network. In: Mahmud, M., Vassanelli, S., Kaiser, M.S., Zhong, N. (eds.) BI 2020. LNCS (LNAI), vol. 12241, pp. 297–306. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-59277-6\\_27](https://doi.org/10.1007/978-3-030-59277-6_27)
12. Madan, R., Mangipudi, P.S.: Predicting computer network traffic: a time series forecasting approach using DWT, ARIMA and RNN. In: Eleventh International Conference on Contemporary Computing (IC3), pp. 1–5. Noida (2018)
13. Sbrana, A., Debiaso Rossi, A.L., Coelho Naldi, M.: N-BEATS-RNN: deep learning for time series forecasting. In: IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 765–768. Miami(2020). <https://doi.org/10.1109/ICMLA51294.2020.00125>
14. Rokui, J.: Historical time series prediction framework based on recurrent neural network using multivariate time series. In: International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 486–489. Niigata (2021). <https://doi.org/10.1109/IIAI-AAI53430.2021.00084>
15. Ludwig, S.A.: Comparison of time series approaches applied to greenhouse gas analysis: ANFIS, RNN, and LSTM. In: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 1–6. New Orleans (2019)
16. Saini, K., Sharma, S.: Gated recurrent unit (GRU) in RNN for traffic forecasting based on time-series data. In: International Conference on Innovative Sustainable Computational Technologies (CISCT), pp. 1–4. Dehradun(2022)
17. Hu, Y., Wang, N., Lyu, L., Zhou, X., Fang, M.: Application of Seq2Seq model based on TCN-GRU to multivariate water quality time series prediction. In: International Conference of Information and Communication Technology (ICTech), pp. 201–205. China (2023). <https://doi.org/10.1109/ICTech58362.2023.00058>
18. Mohapatra, S.K., Mishra, S., Tripathy, H.K.: Energy consumption prediction in electrical appliances of commercial buildings using LSTM-GRU Model. In: International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), pp. 1–5. Bhubaneswar (2022). <https://doi.org/10.1109/ASSIC55218.2022.10088334>
19. Riemer, M., et al.: Correcting forecasts with multifactor neural attention. In: International Conference on Machine Learning. PMLR, pp. 3010–3019 (2016)
20. Hu, J., Zheng, W.: Multistage attention network for multivariate time series prediction. Neurocomputing, 122–137 (2020)
21. Zheng, W., Zhao, P., Chen, G., Zhou, H., Tian, Y.: A hybrid spiking neurons embedded LSTM network for multivariate time series learning under concept-drift environment. IEEE Trans. Knowl. Data Eng., 6561–6574 (2023). <https://doi.org/10.1109/TKDE.2022.3178176>
22. Bellec, G., et al.: Long short-term memory and learning-to-learn in networks of spiking neurons. In: Proceedings Advance Neural Information Processing Systems, pp. 795–805 (2018)
23. Yin, B., et al.: Effective and efficient computation with multiple timescale spiking recurrent neural networks. In: Proceedings International Conference Neuromorphic Systems, pp. 1:1–1:8 (2020)

24. Fang, H., et al.: Exploiting neuron and synapse filter dynamics in spatial temporal learning of deep spiking neural network. In: Proceedings of the 29th International Joint Conference Artificial Intelligence, pp. 2799–2806 (2020). <https://doi.org/10.48550/arXiv.2003.02944>
25. Zhang, W., Li, P.: Temporal spike sequence learning via backpropagation for deep spiking neural networks. In: Advance Neural Information Processing Systems, Art (2020)
26. Wu, Y., et al.: Direct training for spiking neural networks: faster, larger, better. In: Proceedings of the 33rd AAAI Conference Artificial Intelligence, pp. 1311–1318 (2019). <https://doi.org/10.1609/aaai.v33i01.33011311>
27. He, W., et al.: Comparing SNNs and RNNs on neuromorphic vision datasets: similarities and differences. *Neural Netw.* **132**, 108–120 (2020)
28. Zheng, H., et al.: Going deeper with directly-trained larger spiking neural networks. In: Proceedings of the 35th AAAI Conference Artificial Intelligence, pp. 11062–11070 (2021)
29. Zhang, M., Qu, H., Belatreche, A., Chen, Y., Yi, Z.: A highly effective and robust membrane potential-driven supervised learning method for spiking neurons. *IEEE Trans. Neural Netw. Learn. Syst.* **30**(1), 123–137 (2019)
30. Jeyasothy, A., Sundaram, S., Sundararajan, N.: SEFRON: a new spiking neuron model with time-varying synaptic efficacy function for pattern classification. *IEEE Trans. Neural Netw. Learn. Syst.* **30**(4), 1231–1240 (2019)
31. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, Cambridge, U.K. (2002)
32. Gerum, R., et al.: Leaky-integrate-and-fire neuron-like long-short-term-memory units as model system in computational biology. In: 2023 International Joint Conference on Neural Networks (IJCNN). IEEE (2023)
33. Wu, Z., et al.: LIAF-Net: leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Trans. Neural Netw. Learn. Syst.* (2021)
34. Lu, J., et al.: Learning under concept drift: a review. *IEEE Trans. Knowl. Data Eng.*, 2346–2363(2018)
35. Ponghiran, W., Roy, K.: Hybrid analog-spiking long short-term memory for energy efficient computing on edge devices. In: Proceedings Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 581–586 (2021). <https://doi.org/10.23919/DATEx51398.2021.9473953>
36. John, H.: Metro interstate traffic volume data set. UCI (2019). [https://archive-beta.ics.uci.edu/ml/datasets/metrointerstatetraffic\\_volume](https://archive-beta.ics.uci.edu/ml/datasets/metrointerstatetraffic_volume)
37. Liang, X. et al.: Assessing Beijing's PM2.5 pollution: severity, weather impact, APEC and winter heating. In: Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 471, no. 20150257 (2015). <https://doi.org/10.1098/rspa.2015.0257>
38. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.*, 1735–1780 (1997). <https://doi.org/10.1162/neco.1997.9.8.1735>
39. Chung, J., et al.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014). <https://doi.org/10.48550/arXiv.1412.3555>
40. Salinas, D., et al.: DeepAR: probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecast.*, 1181–1191 (2020)
41. Zhao, J., et al.: Do RNN and LSTM have long memory. In: Proceedings of the 36th International Conference Machine Learning, pp. 11365–11375 (2020)



# A Two-Stage Network for Enhanced Intracranial Artery 3D Segmentation in TOF-MRA Volume

Bin Hu<sup>1,2</sup>, Shi-Qi Liu<sup>1(✉)</sup>, Xiao-Liang Xie<sup>1</sup>, Xiao-Hu Zhou<sup>1</sup>, Tao Wang<sup>3</sup>, Ji-Chang Luo<sup>3</sup>, De-Lin Liu<sup>3</sup>, Zeng-Guang Hou<sup>1,2(✉)</sup>, and Jia-Xing Wang<sup>1</sup>

<sup>1</sup> The State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China

{shiqi.liu,zengguang.hou}@ia.ac.cn

<sup>2</sup> School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

<sup>3</sup> Department of Neurosurgery, Xuanwu Hospital, Capital Medical University, Beijing, China

**Abstract.** Accurate segmentation of three-dimensional (3D) vascular structures from TOF-MRA images is crucial for the diagnosis and interventional treatment of cerebrovascular diseases. However, current 3D segmentation algorithms often result in numerous false positives, where non-vascular or non-target vascular regions are incorrectly segmented. Additionally, the segmented vessels frequently exhibit poor continuity. To address these issues, we propose a simple yet effective two-stage 3D vessel segmentation network. In the first stage, a hybrid CNN-Transformer network is designed to achieve 3D vessel segmentation. In the second stage, another standalone network is employed to refine the segmentation results from the first stage, yielding more accurate outcomes. This second-stage network is trained independently using a corrupt-reconstruct approach, which endows it with the ability to enhance segmentation results. A TOF-MRA dataset has been constructed and extensive experiments have been carried out. The experimental results demonstrate that the proposed method significantly enhances 3D vessel segmentation performance, offering potential benefits for the diagnosis and interventional treatment of intracranial diseases. Moreover, the proposed method is flexible and can be extended to new models by altering or improving specific components within the two-stage framework, providing new insights for future research on 3D vessel segmentation.

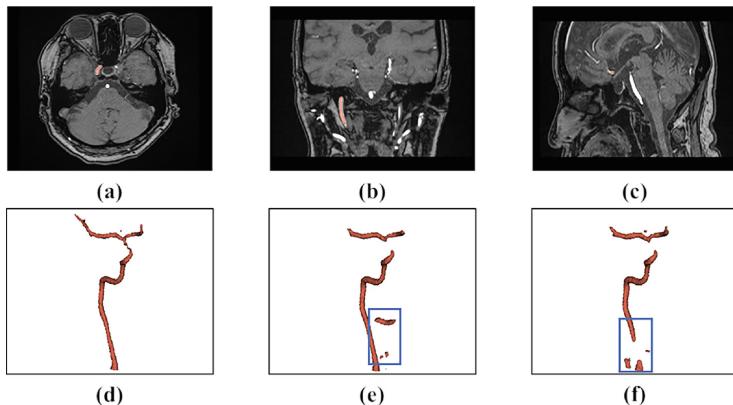
**Keywords:** 3D Segmentation · Intracranial Artery · TOF-MRA

## 1 Introduction

Cerebrovascular diseases, such as intracranial atherosclerotic stenosis and acute ischemic stroke, pose severe threats to human health [1,2]. Time-of-Flight Magnetic Resonance Angiography (TOF-MRA) is a non-invasive imaging technique

widely used for three-dimensional (3D) imaging of head and neck arteries [3]. Therefore, accurate segmentation of vascular structures from TOF volumes is crucial for the diagnosis and interventional treatment of cerebrovascular diseases [4,5].

However, existing 3D segmentation algorithms have several limitations. The brain contains numerous vessels with complex branching, all appearing as bright regions in TOF volumes with similar morphology. Moreover, other non-vascular structures might also appear as bright regions in TOF volumes. Consequently, extracting the target vessel segments from TOF volumes is challenging, resulting in some non-target vessel segments being incorrectly segmented as false positives. Furthermore, due to the thin and tortuous morphology of intracranial vessels, the segmented vessels often lack continuity and exhibit breakages, as illustrated in Fig. 1.



**Fig. 1.** Example of a TOF volume and its vessel segmentation results. (a)-(c) show central slices of the TOF volume from three different perspectives, with the red regions indicating the target vessel areas for segmentation. (d) is the 3D structure of the segmentation label, i.e. the target vessel. (e) and (f) are the segmentation results obtained by existing methods, with the blue boxes highlighting the unsatisfactory areas. Notably, the results contain some false positives and exhibit poor continuity. (Color figure online)

To address these challenges, we propose a two-stage method to achieve enhanced 3D segmentation of intracranial vessels from TOF volumes. In the first stage, a hybrid CNN-Transformer network is designed for 3D vessel segmentation. CNNs excel at extracting local features, while Transformers are adept at capturing global information [6]. The proposed hybrid network combines the strengths of both, leading to improved segmentation performance. In the second stage, another standalone network is employed to refine the initial segmentation results from the first stage. To tackle the issues of numerous false positives and poor vascular continuity present in segmentation results, this network is trained

independently using a corrupt-reconstruct approach. Label corruption strategies are designed to specifically improve these segmentation issues. By utilizing this coarse-to-fine two-stage segmentation framework, the number of false positives in the segmentation results is significantly reduced, vascular continuity is enhanced, and the overall segmentation quality is substantially improved. Our contributions are as follows:

- A simple yet effective two-stage 3D vessel segmentation method is proposed. Two networks are trained independently with different aims and are combined in series during inference to achieve enhanced segmentation results.
- In the first stage, a hybrid CNN-Transformer network is designed to perform 3D vessel segmentation, leveraging both local features and global information to obtain better preliminary segmentation results.
- For the second-stage enhancement network, three label corruption strategies and a corrupt-reconstruct training strategy are introduced, aiming to eliminate false positives and improve vessel continuity.
- A TOF dataset comprising 165 collected and annotated brain TOF volumes is created. Extensive experiments conducted on this dataset demonstrate the effectiveness and superiority of the proposed method.

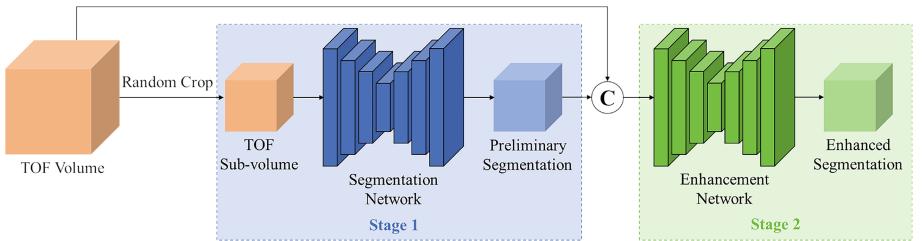
## 2 Related Work

In recent years, deep learning has made significant advances in the field of medical image segmentation. Numerous CNN-based segmentation networks have been proposed. U-Net [7] was introduced for 2D segmentation tasks and achieved excellent performance, with its U-shaped encoder-decoder architecture combined with skip connections becoming the mainstream and extending to 3D segmentation [8]. Although CNN is powerful in extracting local features, it inherently struggles with capturing global information. Conversely, Transformer-based networks excel at modeling long-range dependencies [9–11]. Many Transformer-based networks have been developed for 2D or 3D medical image segmentation [12, 13]. CNN and Transformer each have their strengths and weaknesses, and some studies have attempted to combine them in various ways to achieve better results [14–17].

Vessels have unique morphological characteristics, leading to specialized studies focused on this particular segmentation target. Mou et al. proposed a segmentation network that pays more attention to the curved structure of vessels [18]. Qi et al. designed dynamic snake convolutions tailored for tubular structures like vessels [19]. 3D volumes are quite large, and due to storage limitations, they are cropped into sub-volumes before being fed into networks. However, this method results in the loss of global information, reducing segmentation performance. Zhao et al. designed a region-global fusion network to reintroduce global information [20]. Sun et al. avoided the adverse effects of missing global information by tracking the vessels while segmenting sub-volumes, but sometimes obtained unexpected results [21].

### 3 Method

The proposed method consists of two sequential networks to achieve a two-stage 3D vessel segmentation, with the overall framework illustrated in Fig. 2. Following the common 3D segmentation workflow, the TOF volumes are randomly cropped before being fed into the network. In the first stage, a hybrid CNN-Transformer network is designed to perform the preliminary 3D segmentation. In the second stage, the randomly cropped sub-volumes and the corresponding segmentation results from the first stage are concatenated along the channel dimension and input into another simple 3D segmentation network, resulting in enhanced and more accurate segmentation outcomes with fewer false positives and better continuity.



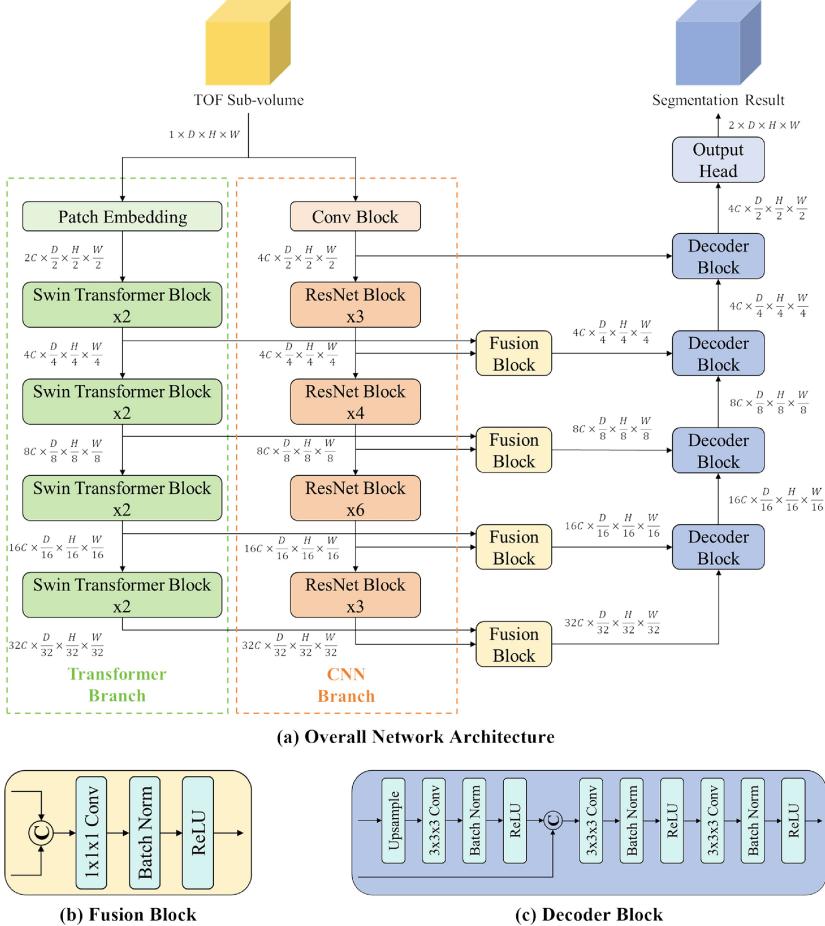
**Fig. 2.** Overview of the proposed two-stage method. A hybrid CNN-Transformer network is designed to perform the preliminary 3D segmentation in the first stage. In the second stage, another network, trained independently with special settings (see Sect. 3.2), is employed to enhance the segmentation performance.

#### 3.1 Hybrid CNN-Transformer Network

Recent studies have shown that convolutional neural networks (CNNs) excel at extracting local features from images, while Transformer networks effectively model long-range dependencies and capture global information [6, 22]. For 3D vessel segmentation tasks, both local features and global information are crucial. Therefore, we propose a hybrid CNN-Transformer network in the first stage of our method.

Specifically, the network is designed as an encoder-decoder U-shaped architecture, as shown in Fig. 3. In the encoder part, a dual-branch design is used for multi-scale feature extraction from the input sub-volumes. One branch consists of CNN blocks and the other of Transformer blocks. ResNet [23] and Swin Transformer [11] are employed as the backbone of the dual-branch encoder. Features from the same scale in both branches are fused to enhance both local and global information. Specifically, feature maps are concatenated along the channel dimension and then fused through convolution, retaining the most comprehensive and rich information. This fusion block allows the network to reinforce features

beneficial for the segmentation task during training. In the decoder part, CNN blocks and upsampling are used to gradually increase the feature map size, ultimately yielding the segmentation results. Skip connections are employed between encoder and decoder features of the same scale through channel concatenation to supplement the details lost during upsampling in the encoder and refine the segmentation outcomes.



**Fig. 3.** Architecture of the proposed hybrid CNN-Transformer Network. (a) shows the overall U-shaped architecture, including the dual-branch encoder, fusion block, skip connection, and decoder. (b) and (c) show the details of the fusion block and decoder block, respectively.

The network input is a single-channel TOF sub-volume with dimensions  $D \times H \times W$ , and the deepest feature maps are reduced to a size of  $\frac{D}{32} \times \frac{H}{32} \times \frac{W}{32}$ ,

with an increased number of channels  $32C$ . The network output is a two-channel segmentation result, where each channel represents a target class, i.e. background or vessel. The network is trained with the supervision of the ground truth segmentation labels.

### 3.2 Segmentation Enhancement

False positives refer to non-target regions that are incorrectly segmented as target vessels in the 3D segmentation results. These regions typically appear as bright areas in TOF volumes, similar to target vessels, making them difficult to distinguish and leading to segmentation errors. Discontinuity in segmented vessels mainly occurs in narrow parts. To address these issues, the network needs to focus more on extreme details. However, it is challenging for a single network, like the one in the first stage, to simultaneously segment the overall target and attend to such details. Therefore, the second stage in our method uses a separate network to enhance the segmentation results.

The second-stage network is trained in a corrupt-reconstruction manner. Corrupted labels and their corresponding TOF sub-volumes are fed into the model to reconstruct the original segmentation labels. Since 3D segmentation is performed on sub-volumes, their segmentation labels lack complete structural information. Therefore, incorporating the corresponding original TOF sub-volumes is necessary. Specific corruption operations on the input labels during training are designed to achieve the intended goals, including false positives simulation, random morphological operation, and random masking. The entire process of generating corrupted labels is shown in Algorithm 1. Through this well-designed training process, the second-stage network effectively identifies and corrects errors in the preliminary segmentation during the inference phase, thus improving overall segmentation performance.

**False Positive Simulation.** As analyzed previously, the false positives in the segmentation results correspond to other bright regions in the TOF volumes outside the target vessels. To enable the network to eliminate these false positives, simulated false positives are introduced into the corrupted labels during the training process of the second-stage network. Specifically, for a given TOF volume  $V \in \mathbb{R}^{D \times H \times W}$ , bright regions are first marked using threshold segmentation to obtain  $L_t \in \mathbb{R}^{D \times H \times W}$ . These bright regions include both the target vessels and non-target vessels, the latter of which are prone to being segmented as false positives. In our method, a fixed ratio  $r \in [0, 1]$  is used for threshold segmentation, where the ratio  $r$  is multiplied by the maximum intensity value  $m$  in the TOF volume to obtain the segmentation threshold  $t$ , as shown in Eqs. (1)–(3). Next, random cropping is performed to obtain TOF sub-volume  $V_s \in \mathbb{R}^{d \times h \times w}$  and its corresponding corrupted labels  $L_c \in \mathbb{R}^{d \times h \times w}$ . This approach ensures that the corrupted labels realistically simulate the presence of false positives in the network’s segmentation results.

$$m = \max_{i,j,k} V(i, j, k), (i, j, k) \in \{1, 2, \dots, D\} \times \{1, 2, \dots, H\} \times \{1, 2, \dots, W\} \quad (1)$$

**Algorithm 1.** Corrupt Label Generation

---

**Input:** TOF volume  $V$ , threshold ratio  $r$ , mask ratio  $k$ , and morphological operation structure element  $e$ .

**Output:** TOF sub-volume  $V_s$  and its corrupt label  $L_c$ .

- 1: get the maximum intensity value  $m$  in volume  $V$
  - 2: calculate threshold  $t \leftarrow r \times m$
  - 3: get the threshold segmentation result  $L_t$  of volume  $V$  according to threshold  $t$
  - 4: random crop to get sub-volume  $V_s$  and its corrupt label  $L_c$  from  $V$  and  $L_t$
  - 5: generate random morphological operation flag  $f \in \{0, 1, 2, 3, 4\}$
  - 6: **if**  $f == 1$  **then**
  - 7:    $L_c \leftarrow dilation(L_c, e)$
  - 8: **else if**  $f == 2$  **then**
  - 9:    $L_c \leftarrow erosion(L_c, e)$
  - 10: **else if**  $f == 3$  **then**
  - 11:    $L_c \leftarrow opening(L_c, e)$
  - 12: **else if**  $f == 4$  **then**
  - 13:    $L_c \leftarrow closing(L_c, e)$
  - 14: **end if**
  - 15: generate random mask  $M$  according to mask ratio  $k$
  - 16: get corrupt label  $L_c \leftarrow M \odot L_c$
- 

$$t = r \times m \quad (2)$$

$$L_t(i, j, k) = \begin{cases} 0, & V(i, j, k) < t \\ 1, & V(i, j, k) \geq t \end{cases} \quad (3)$$

**Random Morphological Operation.** Morphological operations are used to further corrupt the labels, simulating imperfect vessel segmentation results and highlighting specific issues [24]. For instance, dilation expands the segmentation results, accentuating false positives, while opening can break the vessel segmentation at narrow regions, simulating discontinuities. To introduce diverse label corruption and improve the network’s performance, four morphological operations are employed: dilation, erosion, opening, and closing. A random flag  $f \in \{0, 1, 2, 3, 4\}$  is generated for each sample and each epoch during training, and one of the four morphological operations is applied. When  $f = 0$ , no morphological operation is performed, which helps enhance the model’s robustness.

**Random Masking.** Applying random masking to the segmentation labels can effectively simulate broken vessel segmentation results. Additionally, in the previous false positive simulation, all bright regions of non-target vessels in TOF volumes are included in the corrupted labels. However, in actual segmentation results, only some of these regions are missegmented as false positives. Random masking reduces simulated false positive regions and introduces irregular shapes, better aligning with the actual requirements for enhancing segmentation results. Specifically, for a previous sub-volume corrupt label  $L_p \in \mathbb{R}^{d \times h \times w}$  and a given masking ratio  $k \in [0, 1]$ , a mask  $M \in \mathbb{R}^{d \times h \times w}$  with the same shape as  $L_p$  is

generated, where each element is either 0 or 1 with the probability  $k$  of being 1. The new corrupted label  $L_n$  is then obtained by element-wise multiplication of  $L_p$  and  $M$ , as shown in Equation (4)-(6).

$$M(i, j, k) \in \{0, 1\}, \forall (i, j, k) \in \{1, 2, \dots, d\} \times \{1, 2, \dots, h\} \times \{1, 2, \dots, w\} \quad (4)$$

$$\frac{\sum_{i=1}^d \sum_{j=1}^h \sum_{k=1}^w M(i, j, k)}{d \times h \times w} \approx k \quad (5)$$

$$L_n = M \odot L_p \quad (6)$$

While this random masking operation appears similar to He et al.'s masked autoencoder (MAE) [25], its usage and purpose differ. MAE is used for self-supervised pre-training in computer vision, enabling the model to learn the semantic features of images. However, in our method, random masking is employed to make the model focus on morphological anomalies in vessel segmentation results, such as false positives and discontinuities, thereby improving performance.

## 4 Experiments

### 4.1 Datasets

An in-house dataset of TOF-MRA volumes was constructed, comprising 165 cerebral TOF-MRA scans collected from 165 patients at Xuanwu Hospital. While the image dimensions vary slightly, the spacing was uniformly set to  $0.6 \times 0.6 \times 0.6$ . Five clinicians annotated the vessels in these TOF images, and the annotations were verified by two experienced physicians. In accordance with clinical requirements for the diagnosis and interventional treatment of intracranial diseases, each TOF volume was annotated to include intracranial arteries from the carotid bifurcation to the A1 and M1 segments. The dataset was randomly divided into 145 cases for the training set and 20 cases for the test set, which is similar to the approach used in some related studies [20].

### 4.2 Implementation Details

In the experiments, the size of the randomly cropped sub-volumes  $D \times H \times W$  was set to  $64 \times 64 \times 64$ . In the first stage, the channel parameter  $C$  of the hybrid CNN-Transformer network (see Fig. 3) was set to 16. During the training of the second stage network, the threshold ratio  $r$  was set to  $\frac{1}{3}$ , and the random morphological operation flag  $f$  was sampled uniformly from  $\{0, 1, 2, 3, 4\}$ . The structuring element  $e$  for morphological operations was defined as a 3D cross-shaped structure with a square connectivity of 1. The random masking ratio  $k$  was set to 0.5 and the mask  $M$  was generated from a uniform distribution. AttnUNet3D [26] was adopted as the second-stage network, chosen for its lightweight design and fast training and inference speed. In the model inference phase, the sliding window method is used to segment the complete TOF volume.

The models were implemented using PyTorch 2.2.2. The MONAI library [27] was utilized for random sub-volume cropping and sliding window inference, while the SciPy library [28] was used for 3D morphological operations. The models were trained and inferred on an NVIDIA RTX A6000 GPU. Both the first-stage and second-stage models were trained using the cross-entropy loss function and the Adam optimizer, with a batch size of 4, a learning rate of 0.0001, and 400 epochs.

### 4.3 Segmentation Results

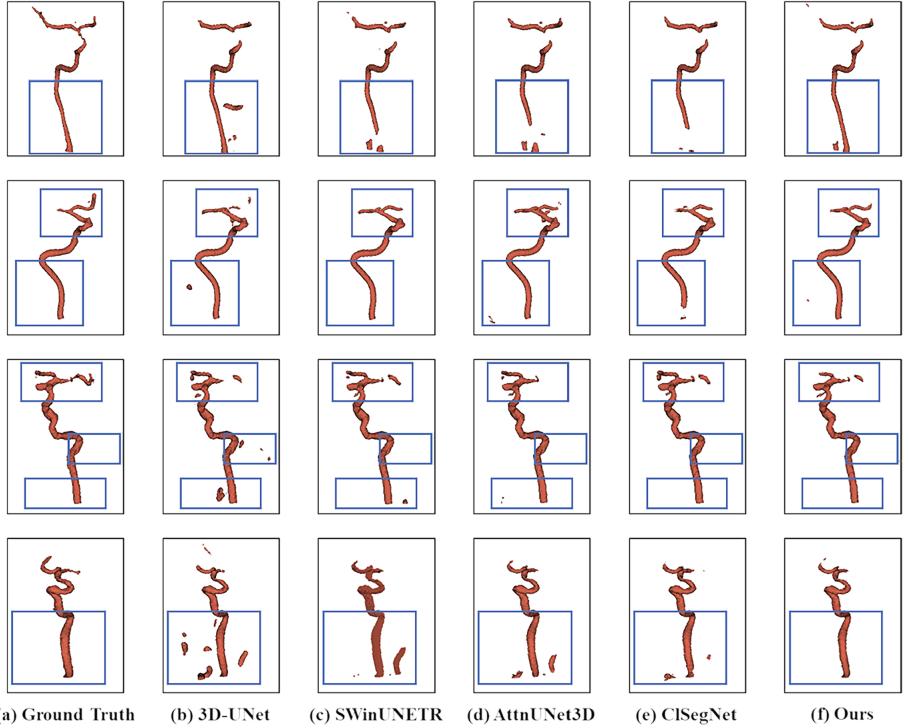
Comparison experiments were conducted on the cerebral TOF-MRA dataset. The proposed method was compared with 11 state-of-the-art 3D segmentation methods, including CNN-based methods, pure Transformer-based methods, and hybrid CNN-Transformer methods. The segmentation performance was evaluated on the test set with four widely used metrics, including precision, IoU, Dice, and HD95. The quantitative results are shown in Table 1, where the best results for each metric are highlighted in bold, the second-best results are underlined, and the third-best results are italicized. Our method achieves the best results on almost all metrics. For the HD95 metric, our method ranks second, but the difference from the best result is negligible. The quantitative comparison results demonstrate the superiority of our method.

**Table 1.** Quantitative comparison results on the cerebral TOF-MRA dataset.

Methods	Precision	IoU	Dice	HD95
3D-UNet [8]	0.8136	0.7444	0.8427	80.19
TransBTS [15]	0.8122	0.7396	0.8418	68.05
UNETR [16]	0.8238	0.7341	0.8372	50.12
SwinUNETR [17]	<i>0.8833</i>	<i>0.7993</i>	<i>0.8830</i>	60.35
VT-UNet [13]	0.7774	0.6765	0.7953	78.27
CS2Net [18]	0.8744	0.7240	0.8320	<i>45.68</i>
DSCNet [19]	0.7745	0.7209	0.8260	57.05
AttnUNet3D [26]	0.8568	0.7867	0.8729	66.97
nnFormer [29]	0.7834	0.7050	0.8174	77.47
CoTr [30]	0.7453	0.6749	0.7941	99.87
ClSegNet [22]	<u>0.8972</u>	<u>0.7995</u>	<u>0.8836</u>	<b>33.04</b>
<b>Ours</b>	<b>0.9348</b>	<b>0.8282</b>	<b>0.9037</b>	<u>34.70</u>

Additionally, segmentation results were visualized for qualitative evaluation. Figure 4 shows four representative cases, each row displaying the results of the same TOF volume segmented by different methods. Blue boxes highlight regions prone to unsatisfactory segmentation. Compared with the four top-performing

models listed in Table 1, our method significantly reduces the number of false positives (e.g., the fourth row of Fig. 4) and achieves better vessel continuity (e.g., the first row of Fig. 4). Although some cases still exhibit minor false positives or discontinuities, our method substantially alleviates these issues compared to others. The qualitative visual analysis indicates that our method yields superior vessel segmentation results.



**Fig. 4.** The visualization of segmentation results. Four representative cases are shown.

#### 4.4 Ablation Study

To validate the effectiveness of each component of the proposed method, we conducted ablation experiments on the cerebral TOF-MRA dataset. Table 2 presents the quantitative results of the ablation experiments on four segmentation metrics (precision, IoU, Dice, and HD95) for six different combination settings of the four optional modules.

First, we verified the effectiveness of the proposed hybrid CNN-Transformer network. According to Table 2, when using only the first-stage hybrid CNN-Transformer network without the second stage, the IoU on the test set reaches

80.48%, and the Dice coefficient reaches 88.66%, outperforming all other methods compared in Table 1. Compared to pure CNN-based methods like 3D-UNet [8] and pure Transformer-based methods like SwinUNETR [17], our hybrid network leverages the strengths of both CNN and Transformer, resulting in superior segmentation performance.

Second, we assessed the improvement brought by the second-stage network. By adding the second-stage enhancement network to refine the segmentation results from the first stage, the precision, IoU, and Dice coefficients improve by 7.12%, 2.34%, and 1.71%, respectively (comparing the first and last rows in Table 2), demonstrating the significant enhancement provided by the second stage. Additionally, the HD95 metric decreased substantially from 66.63 to 34.70, indicating more accurate vessel boundaries in enhanced segmentation results. Since false positives and discontinuities in the vessel segmentation can significantly change the background-foreground boundary, the notable reduction in the HD95 metric suggests an alleviation of these issues.

**Table 2.** Quantitative results of the ablation study.

Second Stage	False Positive Simulation	Random Morphological Operation	Random Masking	Precision	IoU	Dice	HD95
✗	✗	✗	✗	0.8636	0.8048	0.8866	66.63
✓	✗	✗	✗	0.8260	0.7567	0.8541	60.76
✓	✓	✗	✗	0.8951	0.8204	0.8998	70.51
✓	✗	✓	✗	0.8739	0.8099	0.8907	53.33
✓	✗	✗	✓	0.8537	0.7998	0.8830	68.87
✓	✓	✓	✓	<b>0.9348</b>	<b>0.8282</b>	<b>0.9037</b>	<b>34.70</b>

Furthermore, we validated the necessity of using the corrupt-reconstruct approach to train the second-stage network independently. Although the proposed method can be seen as a simple concatenation of two models during the inference phase, the two models are trained separately with a well-designed strategy. Intuitively, concatenating networks might enhance performance, but experimental results show otherwise. When the two stages are directly concatenated and jointly trained, as shown in the second row of Table 2, there is only a minor improvement in the HD95 metric, while other metrics actually degrade compared to using only the first stage (the first row of Table 2). This indicates that the overall segmentation performance does not effectively improve. This result demonstrates that the superior performance of our method is not solely due to the simple concatenation of networks or the increase in the number of parameters. The designed training strategy is crucial, enabling the first-stage network to focus on the conventional segmentation task and the second-stage network to enhance the preliminary segmentation results. Together, they achieve the best segmentation performance.

Lastly, we validated the effectiveness of the proposed label corruption strategies. As shown in Table 2, training the second-stage network with either false positive simulation or random morphological operation alone improves segmentation performance. Applying random masking alone is less effective, likely because the corrupted labels differ significantly from the suboptimal results produced by preceding segmentation networks, creating a large gap between the reconstruction task and the desired segmentation enhancement task. However, random masking introduces substantial morphological anomalies during label corruption, which helps improve the network’s robustness. Combining all three label corruption strategies yields the best results across all metrics. The ablation study results demonstrate the effectiveness of the designed label corruption methods and training strategies.

## 5 Conclusion

In this paper, we propose a two-stage network to address the challenges of numerous false positives and poor continuity in TOF-MRA intracranial artery 3D segmentation. In the first stage, a hybrid CNN-Transformer network is employed for conventional 3D vessel segmentation, leveraging the strengths of both CNN and Transformer networks to achieve superior performance. In the second stage, a network trained with a corrupt-reconstruct approach is used to enhance the segmentation results. Specific label corruption strategies are designed to guide this enhancement network to reduce false positives and improve vessel continuity effectively. We collect and annotate 165 cerebral TOF-MRA volumes to construct a dataset. Extensive experiments conducted on this dataset demonstrate that the proposed method achieves state-of-the-art results, offering significant potential benefits for the diagnosis and interventional treatment of cerebrovascular diseases. Furthermore, the proposed method serves as a flexible framework that can be extended to new approaches. By modifying or improving certain components within the two-stage framework, further performance enhancements may be achieved, providing valuable ideas for future research on 3D vessel segmentation.

**Acknowledgments.** This work was supported in part by the National Key Research and Development Program of China (No. 2022YFB4700902), the National Natural Science Foundation of China (No. 62303463 and No. 62073325), the Haidian Frontier Project of the Beijing Natural Science Foundation (No. L232137), the National High Level Hospital Clinical Research Funding (No. 2022-PUMCH-B-125), and the Development Project of National Major Scientific Research Instrument (No. 82327801).

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Fischer, U., Hsieh-Meister, K., Kellner-Weldon, F., et al.: Symptomatic and asymptomatic intracranial atherosclerotic stenosis: 3 years' prospective study. *J. Neurol.* **267**(6), 1687–1698 (2020)
2. GBD 2016 Lifetime Risk of Stroke Collaborators: Global, Regional, and Country-Specific Lifetime Risks of Stroke, 1990 and 2016. *New England J. Med.* **379**(25), 2429–2437 (2018)
3. Saloner, D.: An introduction to MR angiography. *Radiographics* **15**(2), 453–465 (1995)
4. Park, H.Y., Suh, C.H., Shim, W.H., et al.: Diagnostic yield of TOF-MRA for detecting incidental vascular lesions in patients with cognitive impairment: an observational cohort study. *Front. Neurol.* **13**, 958037 (2022)
5. Campbell, B., Mitchell, P.J., Kleinig, T.J., et al.: Endovascular therapy for ischemic stroke with perfusion-imaging selection. *N. Engl. J. Med.* **372**(11), 1009–1018 (2015)
6. Song, M., et al.: A novel fusion network for morphological analysis of common iliac artery. In: Wang, L., Dou, Q., Fletcher, P.T., Speidel, S., Li, S. (eds.) *Medical Image Computing and Computer Assisted Intervention – MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part VII*, pp. 54–63. Springer Nature Switzerland, Cham (2022). [https://doi.org/10.1007/978-3-031-16449-1\\_6](https://doi.org/10.1007/978-3-031-16449-1_6)
7. 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.) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III*, pp. 234–241. Springer International Publishing, Cham (2015). [https://doi.org/10.1007/978-3-319-24574-4\\_28](https://doi.org/10.1007/978-3-319-24574-4_28)
8. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3D U-Net: learning dense volumetric segmentation from sparse annotation. In: Ourselin, S., Joskowicz, L., Sabuncu, M.R., Unal, G., Wells, W. (eds.) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016: 19th International Conference, Athens, Greece, October 17–21, 2016, Proceedings, Part II*, pp. 424–432. Springer International Publishing, Cham (2016). [https://doi.org/10.1007/978-3-319-46723-8\\_49](https://doi.org/10.1007/978-3-319-46723-8_49)
9. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. *Adv. Neural. Inf. Process. Syst.* **30**, 1–11 (2017)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al.: An Image is Worth  $16 \times 16$  Words: transformers for Image Recognition at Scale. In: *9th International Conference on Learning Representations*, pp. 1–21. ICLR, Virtual, Online (2020)
11. Liu, Z., Lin, Y., Cao, Y., et al.: Swin Transformer: hierarchical vision transformer using shifted windows. In: *18th IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022. IEEE, Virtual, Online (2021)
12. Cao, H., et al.: Swin-Unet: unet-like pure transformer for medical image segmentation. In: Karlinsky, L., Michaeli, T., Nishino, K. (eds.) *Computer Vision – ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, pp. 205–218. Springer Nature Switzerland, Cham (2023). [https://doi.org/10.1007/978-3-031-25066-8\\_9](https://doi.org/10.1007/978-3-031-25066-8_9)

13. Peiris, H., Hayat, M., Chen, Z., Egan, G., Harandi, M.: A robust volumetric transformer for accurate 3D tumor segmentation. In: Wang, L., Dou, Q., Fletcher, P.T., Speidel, S., Li, S. (eds.) Medical Image Computing and Computer Assisted Intervention – MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part V, pp. 162–172. Springer Nature Switzerland, Cham (2022). [https://doi.org/10.1007/978-3-031-16443-9\\_16](https://doi.org/10.1007/978-3-031-16443-9_16)
14. Chen, J., Lu, Y., Yu, Q., et al.: TransUNet: transformers make strong encoders for medical image segmentation. arXiv preprint, [arXiv:2102.04306](https://arxiv.org/abs/2102.04306) (2021)
15. Wang, W., Chen, C., Ding, M., Yu, H., Zha, S., Li, J.: TransBTS: multimodal brain tumor segmentation using transformer. In: de Bruijne, M., et al. (eds.) Medical Image Computing and Computer Assisted Intervention – MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part I, pp. 109–119. Springer International Publishing, Cham (2021). [https://doi.org/10.1007/978-3-030-87193-2\\_11](https://doi.org/10.1007/978-3-030-87193-2_11)
16. Hatamizadeh, A., Tang, Y., Nath, V., et al.: UNETR: transformers for 3D medical image segmentation. In: 9th IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 574–584. IEEE, Waikoloa, Hawaii (2022)
17. Hatamizadeh, A., Nath, V., Tang, Y., Yang, D., Roth, H.R., Xu, D.: Swin UNETR: swin transformers for semantic segmentation of brain tumors in MRI images. In: Crimi, A., Bakas, S. (eds.) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 7th International Workshop, BrainLes 2021, Held in Conjunction with MICCAI 2021, Virtual Event, September 27, 2021, Revised Selected Papers, Part I, pp. 272–284. Springer International Publishing, Cham (2022). [https://doi.org/10.1007/978-3-031-08999-2\\_22](https://doi.org/10.1007/978-3-031-08999-2_22)
18. Mou, L., Zhao, Y., Fu, H., et al.: CS2-Net: deep learning segmentation of curvilinear structures in medical imaging. Med. Image Anal. **67**, 101874 (2021)
19. Qi, Y., He, Y., Qi, X., et al.: Dynamic snake convolution based on topological geometric constraints for tubular structure segmentation. In: 19th IEEE/CVF International Conference on Computer Vision, pp. 6070–6079. IEEE, Paris, France (2023)
20. Zhao, H., et al.: An effective morphological analysis framework of intracranial artery in 3D digital subtraction angiography. In: Luo, B., Cheng, L., Wu, Z.-G., Li, H., Li, C. (eds.) Neural Information Processing: 30th International Conference, ICONIP 2023, Changsha, China, November 20–23, 2023, Proceedings, Part X, pp. 50–61. Springer Nature Singapore, Singapore (2024). [https://doi.org/10.1007/978-981-99-8141-0\\_4](https://doi.org/10.1007/978-981-99-8141-0_4)
21. Sun, Q., Yang, J., Zhao, S., et al.: LIVE-Net: comprehensive 3D vessel extraction framework in CT angiography. Comput. Biol. Med. **159**, 106886 (2023)
22. Kuang, H., Wang, Y., Liu, J., et al.: Hybrid CNN-transformer network with circular feature interaction for acute ischemic stroke lesion segmentation on non-contrast CT scans. IEEE Trans. Med. Imaging **43**(6), 2303–2316 (2024)
23. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: 29th IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 770–778. IEEE, Las Vegas, Nevada, USA (2016)
24. Adiga, S., Dolz, J., Lombaert, H.: Anatomically-aware uncertainty for semi-supervised image segmentation. Med. Image Anal. **91**, 103011 (2024)
25. He, K., Chen, X., Xie, S., et al.: Masked autoencoders are scalable vision learners. In: 35th IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16000–16009. IEEE, New Orleans, Louisiana, USA (2022)

26. Islam, M., Vibashan, V.S., Jose, V., Wijethilake, N., Utkarsh, U., Ren, H.: Brain tumor segmentation and survival prediction using 3D attention UNet. In: Crimi, A., Bakas, S. (eds.) Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 5th International Workshop, BrainLes 2019, Held in Conjunction with MICCAI 2019, Shenzhen, China, October 17, 2019, Revised Selected Papers, Part I, pp. 262–272. Springer International Publishing, Cham (2020). [https://doi.org/10.1007/978-3-030-46640-4\\_25](https://doi.org/10.1007/978-3-030-46640-4_25)
27. Cardoso, M.J., Li, W., Brown, R., et al.: Monai: an open-source framework for deep learning in healthcare. arXiv preprint [arXiv:2211.02701](https://arxiv.org/abs/2211.02701) (2022)
28. Virtanen, P., Gommers, R., Oliphant, T.E., et al.: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods **17**(3), 261–272 (2020)
29. Zhou, H.Y., Guo, J., Zhang, Y., et al.: nnFormer: volumetric medical image segmentation via a 3D transformer. IEEE Trans. Image Process. **32**, 4036–4045 (2023)
30. Xie, Y., Zhang, J., Shen, C., Xia, Y.: CoTr: efficiently bridging CNN and transformer for 3D medical image segmentation. In: de Bruijne, M., et al. (eds.) Medical Image Computing and Computer Assisted Intervention – MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III, pp. 171–180. Springer International Publishing, Cham (2021). [https://doi.org/10.1007/978-3-030-87199-4\\_16](https://doi.org/10.1007/978-3-030-87199-4_16)



# Independence Constrained Disentangled Representation Learning from Epistemological Perspective

Ruoyu Wang<sup>1</sup>(✉) and Lina Yao<sup>1,2</sup>

<sup>1</sup> University of New South Wales, Sydney, Australia  
`{ruoyu.wang5,lina.yao}@unsw.edu.au`

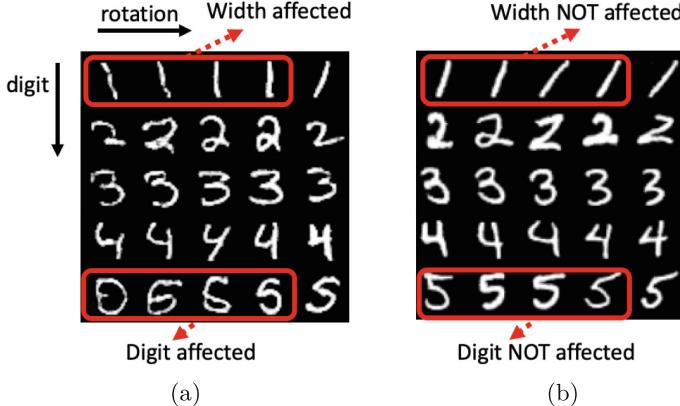
<sup>2</sup> Commonwealth Scientific and Industrial Research Organisation,  
Clayton South, Australia

**Abstract.** Disentangled Representation Learning aims to improve the explainability of deep learning methods by training a data encoder that identifies semantically meaningful latent variables in the data generation process. Nevertheless, there is no consensus regarding a universally accepted definition for the objective of disentangled representation learning. In particular, there is a considerable amount of discourse regarding whether should the latent variables be mutually independent or not. In this paper, we first investigate these arguments on the interrelationships between latent variables by establishing a conceptual bridge between Epistemology and Disentangled Representation Learning. Then, inspired by these interdisciplinary concepts, we introduce a two-level latent space framework to provide a general solution to the prior arguments on this issue. Finally, we propose a novel method for disentangled representation learning by employing an integration of mutual information constraint and independence constraint within the Generative Adversarial Network (GAN) framework. Experimental results demonstrate that our proposed method consistently outperforms baseline approaches in both quantitative and qualitative evaluations. The method exhibits strong performance across multiple commonly used metrics and demonstrates a great capability in disentangling various semantic factors, leading to an improved quality of controllable generation, which consequently benefits the explainability of the algorithm.

**Keywords:** Disentangled Representation Learning · Generative Adversarial Network · Explainability

## 1 Introduction

Representation learning is widely recognized as a fundamental task in the field of machine learning, as the efficacy of machine learning methods heavily relies on the quality of data representation. It is suggested that an ideal representation should be disentangled [1], which means it can identify the genuine generative



**Fig. 1.** Disentangled Representation Learning learns semantically meaningful latent variables such as *Rotation* and *Digit*. Our method outperforms existing methods by encouraging the factors to be further separated. For example, on MNIST: (a) Without the independence constraint in our method, the *digit* and *width* are affected when traversing on variable *rotation*; (b) With the independence constraint, *digit* and *width* are NOT affected when traversing on *rotation*.

factors hidden in the observed data, and the latent variables should be semantically meaningful and correspond to the ground truth generative factors.

However, there is no general agreement on a formal definition of disentangled representation [1, 13, 31]. Despite the lack of agreement on the formal definition of disentangled representation learning, existing methods suggested that two quantities are significant in disentangled representation learning: 1) Mutual Information between the latent variables and the data [8]; and 2) Independence between the latent variables [7, 17].

Nevertheless, regarding the second quantity *Independence between the latent variables*, it is worth noting that there is a lack of consensus about whether latent variables should be mutually independent in disentangled representation learning. While some [1, 7, 13, 14, 17] suggest that hidden factors should be strictly independent, some other works [27, 30, 31, 35] argued that causal relationships exist between generative factors, thus they are not necessarily independent. Therefore, these arguments lead us to ask:

*What should be considered as generative factors? What should be independent in latent space? And what should be causally connected?*

To answer these questions, it is crucial to understand how humans perceive and comprehend these factors and their relationships, because the fundamental objective of disentangled representation learning is to extract factors that are **interpretable to us human**. Therefore, in this paper, we first answer the above questions by borrowing the concepts from epistemology. Then, based on these interdisciplinary theories, we introduce a unified framework to consolidate

prior arguments regarding the relationships between latent variables. Finally, after clarifying these questions, we propose a novel method for disentangled representation learning that jointly optimizes the two objectives mentioned earlier: 1) the mutual information objective and 2) the independent objective. The contribution of this paper is threefold:

- We establish a conceptual bridge between epistemology and disentangled representation learning to facilitate the understanding of the data generation process and disentangled representation learning.
- We introduce a two-level latent space framework to unify the prior arguments regarding the relationships between generative factors and latent variables in disentangled representation learning.
- We propose a novel method for disentangled representation learning to jointly optimize the mutual information and the independent objectives, which outperforms the baseline methods consistently on multiple evaluation metrics.

## 2 Our Method

### 2.1 A Perspective of Epistemology

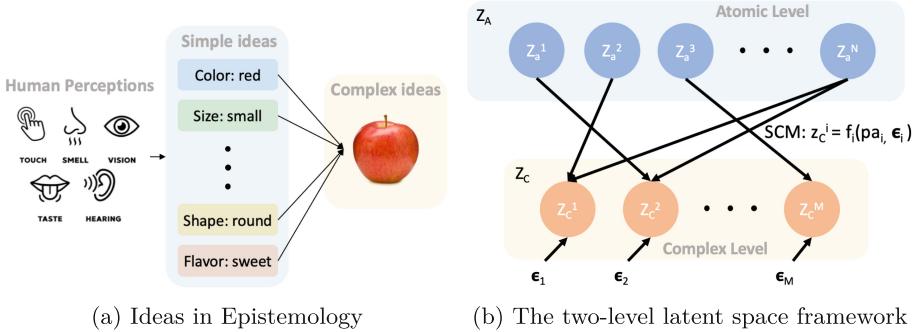
As discussed in Sect. 1, comprehending the relationships between latent variables in disentangled representation learning necessitates an understanding of how humans perceive these factors. Because the concept of *interpretability* inherently implies interpretability to humans; therefore, interdisciplinary insights from epistemology are indispensable in this context.

In epistemology, mental representations of perceptions are called *ideas*, which can be grouped into two categories, simple ideas and complex ideas [15]. Simple ideas are basic, indivisible concepts that form the foundation of our knowledge, and complex ideas are more advanced concepts that are built upon multiple simple ideas. For instance, when we imagine an apple (Fig. 2a), the sensory perceptions of an apple such as its colour, shape and taste are irreducible, and thus are regarded as simple ideas. These simple ideas can then be combined to form the complex idea of an apple as a whole.

Taking these theories as a reference, we argue that the disagreement regarding whether latent variables should be independent arises from the lack of understanding of this hierarchical structure of human concepts. Existing works instinctively consider ALL latent variables in a single latent space. However, building upon these interdisciplinary theories, we contend that interpretable latent variables in disentangled representation learning should also follow a hierarchical structure in a similar way as demonstrated in Fig. 2a.

### 2.2 Two-Level Latent Space Framework

Inspired by the interdisciplinary theories introduced in Sect. 2.1, we propose a two-level latent space framework to consolidate prior arguments on the interrelationships between the latent variables, as illustrated in Fig. 2b.



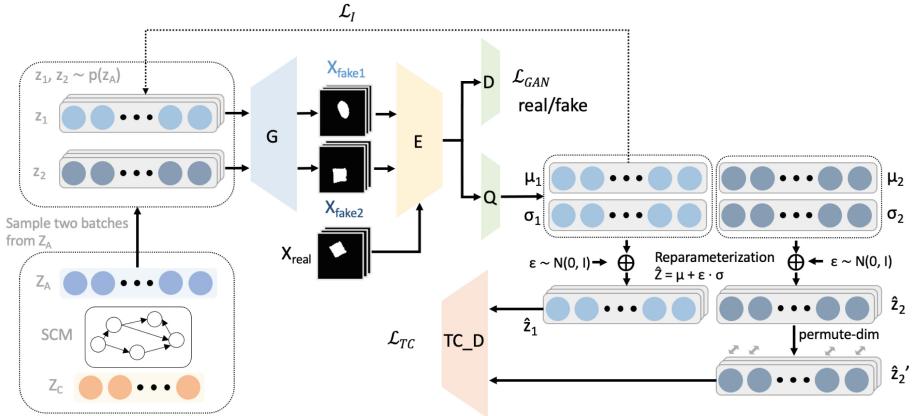
**Fig. 2.** (a) In epistemology, the complex idea *apple* is composed by simple ideas such as its *colour*, *shape* and *size*, which are irreducible and mutually independent; (b) Our proposed framework groups latent variables into two levels: 1) **Atomic Level**  $Z_A$ : comprises factors that are basic and irreducible such as colour and shape of an apple, where all factors are mutually independent; 2) **Complex Level**  $Z_C$ : comprises factors that are derived from the atomic level variables. The two levels are connected by causal relationships.

The framework groups all latent variables into two levels: 1) **Atomic Level**: which corresponds to the simple idea in epistemology, and comprises the factors that can be directly perceived from the observed data, and latent variables in this level should be **mutually independent**. 2) **Complex Level**: which comprises concepts derived from the atomic level variables. The two levels are connected by causal relationships and can be modelled by the Structural Causal Model (SCM). Within this framework, atomic-level latent variables should be **mutually independent**, and complex-level variables are **not necessarily independent** because the atomic-level variables may work as confounders due to the causal relationships between the two levels.

We argue that prior arguments regarding the relationships between latent variables are primarily due to their focus on a set of selected factors in certain datasets, thus having different views on this issue. In contrast, our framework, supported by the theories in epistemology [15] and also cognitive science [2, 25, 29], offers a general explanation that is adaptable to various scenarios, and consolidates prior arguments regarding the interrelationships between latent variables. In this paper, we concentrate on the datasets that consist only of atomic-level latent variables, thus all latent variables are independent. Therefore, we leverage the constraints of independent and mutual information to augment the efficacy of our proposed method for disentangled representation learning.

### 2.3 Method Formulation

After clarifying the legitimacy of applying independence constraints in this problem in Sect. 2.2, we introduce our proposed method in this section. We build our method in the paradigm of Generative Adversarial Networks (GAN). Formally,



**Fig. 3.** Framework of TC-GAN. Two batches of latent variables  $z_1, z_2$  are sampled from the atomic level latent space and passed to the generator  $G$  to generate the fake images. The discriminator  $D$  and the auxiliary network  $Q$  share the same data encoder  $E$ . The network  $Q$  outputs the predicted mean  $\mu_1, \mu_2$  and variance  $\sigma_1, \sigma_2$  of the latent factor. Then, we employ the re-parameterization trick and the permute-dim algorithm to aid the  $TCD$  in estimating Total Correlation.

GAN solves the minimax optimization problem (Eq. 1) by utilizing a generator  $G$  and a discriminator  $D$ , and the generator  $G$  could learn the real data distribution  $P_{real}(x)$  when this framework converges.

$$\min_G \max_D \mathcal{L}_{GAN}(D, G) = \mathbb{E}_{x \sim P_{real}} [\log D(x)] + \mathbb{E}_{z \sim noise} [\log(1 - D(G(z)))] \quad (1)$$

However, this framework imposes no restrictions on the semantic meaning of the latent variable  $z_i$ . To encourage the latent variables to possess semantic meanings, as introduced in Sect. 1, we apply the two constraints in this framework: 1) Mutual information between the latent variables and the data [8]; and 2) Independence between the latent variables [7, 17].

For the constraint on mutual information, we adapt the implementation of InfoGAN [8]. First, the latent variables are decomposed into latent code  $z$  which controls the semantics in the image, and noise  $\epsilon$  which is considered incompressible. Then, an auxiliary network  $Q$  is introduced into the framework to maximize the lower bound of mutual information between the latent code  $z$  and the image (Eq. 2). In practice, a regularization term  $\mathcal{L}_I(G, Q)$  (Eq. 5) is introduced into the framework to maximize the Mutual Information between latent variable  $z$  and the generated data.

$$I(z, G(z, \epsilon)) \geq \mathbb{E}_{x \sim G(z, \epsilon)} [\mathbb{E}_{z' \sim P(z|x)} \log Q(z'|x)] + H(z) \quad (2)$$

On the other hand, for the independence constraint, we aim to minimize the *Total Correlation* [34] between latent variables, and apply this constraint to the

learned latent code  $\hat{z}$  such that:

$$TC(\hat{z}) = KL(q(\hat{z}) || \prod_i q(\hat{z}_i)) \quad (3)$$

where  $\hat{z}$  is the samples drawn from the learned posterior distribution obtained by  $Q(G(z))$ . Therefore, by integrating these two constraints, the overall objective of our method becomes:

$$\min_{G, Q} \max_D \mathcal{L}_{TCGAN} = \mathcal{L}_{GAN}(D, G) - \lambda \mathcal{L}_I(G, Q) + \beta \mathcal{L}_{TC}(G, Q) \quad (4)$$

where

$$\mathcal{L}_I(G, Q) = \mathbb{E}_{z \sim P_{(z)}, x \sim G_{(z, \epsilon)}} [\log Q(z|x)] \quad (5)$$

$$\mathcal{L}_{TC}(G, Q) = KL(q(\hat{z}) || \prod_i q(\hat{z}_i)) \quad (6)$$

Thus far, we have formulated our method by integrating the two constraints in the paradigm of Generative Adversarial Network. However, the total correlation term (Eq. 6) is intractable. In Sect. 2.4, we introduce the method to estimate this term and how this estimation process is integrated into our framework, then give an end-to-end illustration of our method.

## 2.4 Total Correlation Estimation

To estimate the Total Correlation term (Eq. 6), we adapt the method from FactorVAE [17]. Specifically, we utilize the Density-Ratio trick, which trains a Total Correlation Discriminator  $TCD$  to predict the probability that the input vector is from  $q(\hat{z})$  rather than  $\prod_i q(\hat{z}_i)$ , and then the Total Correlation term can be estimated by Eq. 7.

$$TC(\hat{z}) = \mathbb{E}_{q(\hat{z})} \left[ \log \frac{q(\hat{z})}{\prod_i q(\hat{z}_i)} \right] \approx \mathbb{E}_{q(z)} \left[ \log \frac{TCD(\hat{z})}{1 - TCD(\hat{z})} \right] \quad (7)$$

The end-to-end framework of our method is illustrated in Fig. 3. On top of the vanilla GAN framework which comprises a generator  $G$ , a data encoder  $E$  and a Discriminator head  $D$ , we first utilize an auxiliary network  $Q$  to predict the mean and variance of the latent variables of the generated data, as implemented in InfoGAN [8]. Then, we sample  $\hat{z}$  by utilizing a reparametrization trick to ensure the differentiability of the framework. By employing this approach, we acquired samples from the distribution  $q(\hat{z})$ . On the other hand, samples from  $\prod_i q(\hat{z}_i)$  cannot be sampled directly, so we adapt the permute-dim algorithm proposed in [17] to do the sampling, where the samples are drawn by randomly permuting across the batch for each latent dimension. Therefore, the input of our framework should be two randomly selected batches of latent variables, one of which is used to calculate the mutual information loss  $\mathcal{L}_I$  and sample from  $q(\hat{z})$ , another batch is used to sample from  $\prod_i q(\hat{z}_i)$  in order to estimate the total correlation loss  $\mathcal{L}_{TC}$ .

**Algorithm 1:** Pseudocode of TC-GAN

---

**Input:** dataset  $\{x_k\}_{k=1}^N$ , batch size  $M$ , network structure Generator  $G$ , Discriminator  $D$ , Auxiliary Network  $Q$ , TC Discriminator  $TCD$

**while** *not converged* **do**

Sample latent code  $z_1, z_2 \sim p(z_a)$   
 Sample  $x \sim p_{real}(x)$   
 $x_{fake1} = G(z_1)$   
 Update  $D$  by ascending  $\mathcal{L}_{GAN}$   
 $\mu_1, \sigma_1 = Q(x_{fake1})$   
 $\hat{z}_1 = Reparameterization(\mu_1, \sigma_1)$   
 $\mathcal{L}_I = NLL\_Loss(z_1, \mu_1, \sigma_1)$   
 $\mathcal{L}_{TC} = \log(\bar{TCD}(\hat{z}_1)/(1 - TCD(\hat{z}_1)))$   
 Update  $G, Q$  by descending  $\mathcal{L}_{TCGAN}$   
 $x_{fake2} = G(z_2)$   
 $\mu_2, \sigma_2 = Q(x_{fake2})$   
 $\hat{z}_2 = Reparameterization(\mu_2, \sigma_2)$   
 $\hat{z}'_2 = permute\_dim(\hat{z}_2)$   
 $\mathcal{L}_{TCD} = CrossEntropy(\hat{z}_1, \hat{z}'_2)$   
 Update  $TCD$  by descending  $\mathcal{L}_{TCD}$

**Output:** Trained networks

---

We perform three steps iteratively to train the framework: 1) Train the Discriminator  $D$  in GAN to ensure the generated images look real with other modules fixed; 2) Train the Generator  $G$  and Network  $Q$  by the loss defined in Eq. 4 with other modules fixed, where  $\mathcal{L}_{TC}$  is estimated by Eq. 7; 3) Train the Total Correlation Discriminator  $TCD$  to distinguish  $q(\hat{z})$  and  $\prod_i q(\hat{z}_i)$  with other modules fixed. These steps are illustrated in Algorithm 1.

### 3 Experiments

#### 3.1 Experiment Setting

We evaluate the performance of our method from two perspectives: (1) Quantitative Evaluation (Sect. 3.2), where we compare our method with several baseline methods on multiple commonly used metrics for Disentangled Representation Learning; and (2) Qualitative Evaluation (Sect. 3.3), where we conduct Latent Space Traversal Test and compare our results with the case without the enforcement of independence constraint, to observe the direct impact of our method on the generated images.

We follow the settings in previous works on the model architectures of the Generator, Discriminator [22] and the Total Correlation Discriminator [17]. We use the Adam optimizer for training, with a learning rate equal to 0.001 for the generator, 0.002 for the discriminator and TC discriminator. We use  $\alpha = 0.1$  and  $\beta = 0.001$  in Eq. 4. The latent dimension equals 10. We used a batch size of 64 and trained the model for 30 epochs, then selected the checkpoint with the highest Explicitness score for evaluation on other metrics.

### 3.2 Quantitative Evaluation

Since the quantitative evaluation of disentanglement requires **ground-truth labels** of all generative factors, most datasets are not suitable for quantitative evaluation. Therefore, following previous works in this domain, we concentrate on dSprites [24] for quantitative evaluation, where all factors are well-defined.

**Table 1.** Quantitative Evaluation of Disentanglement on dSprites. We highlight the **highest** and second highest score for each metric in the table. We report the average and standard deviation over 10 runs for all results.

	EXP	JEMMIG	MOD	SAP	Z-diff
VAE [19]	$0.42 \pm .00$	$0.20 \pm .00$	$0.87 \pm .01$	$0.11 \pm .00$	$0.69 \pm .03$
$\beta$ -VAE [14]	$0.49 \pm .00$	$0.26 \pm .00$	$0.82 \pm .01$	$0.16 \pm .00$	$0.86 \pm .01$
AnnealedVAE [3]	$0.72 \pm .01$	$0.33 \pm .00$	$0.97 \pm .00$	$0.39 \pm .01$	$0.86 \pm .05$
Factor-VAE [17]	$0.41 \pm .00$	$0.19 \pm .00$	$0.92 \pm .01$	$0.21 \pm .00$	$0.80 \pm .02$
$\beta$ -TCVAE [7]	$0.68 \pm .01$	$0.12 \pm .00$	$0.90 \pm .00$	$0.22 \pm .00$	$0.87 \pm .03$
InfoGAN [8]	$0.54 \pm .00$	$0.08 \pm .00$	$0.56 \pm .02$	$0.05 \pm .00$	$0.76 \pm .04$
IB-GAN [16]	$0.78 \pm .02$	$0.02 \pm .01$	$0.86 \pm .03$	$0.19 \pm .01$	$0.84 \pm .04$
InfoGAN-CR [22]	$0.62 \pm .00$	$0.38 \pm .00$	$0.95 \pm .00$	$0.41 \pm .00$	$0.99 \pm .02$
TC-GAN (ours)	<b><math>0.85 \pm .01</math></b>	<b><math>0.45 \pm .00</math></b>	<b><math>0.98 \pm .00</math></b>	<b><math>0.48 \pm .00</math></b>	<b><math>0.99 \pm .01</math></b>

**Metrics.** We evaluate the quality of disentanglement by several metrics proposed in recent literature, including 1) **Explicitness Score** [28] which evaluates the quality of disentanglement by training a classifier on latent code to predict the ground-truth factor classes, a higher Explicitness Score indicates that all the generative factors are decoded in the representation; 2) **JEMMIG Score** [9] which evaluates the quality of disentanglement by estimating the mutual information (MI) between the ground-truth generative factors and the latent variables; 3) **Modularity Score** [28] which measures whether one latent variable encodes no more than one generative factor, it estimates the mutual information (MI) between a certain latent variable and the factor with maximum MI and compares it with all other factors; 4) **SAP Score** [20], which evaluates the quality of disentanglement by training a linear regression model for every pair of latent variables and ground-truth factor, and uses the  $R^2$  score of the regression model to denote the disentanglement score; and 5) **Z-diff** [14] which first selects pairs of data points with the same value on a fixed latent variable, and then evaluates the quality of disentanglement by training a classifier to predict which factor was fixed. We used the code provided by [4] for all evaluation metrics.

**Baselines.** We compare our method with several baseline methods in the domain of Disentangled Representation Learning, which include VAE [19],  $\beta$ -VAE [14], AnnealedVAE [3], Factor-VAE [17],  $\beta$ -TCVAE [7], InfoGAN [8], IB-GAN [16], and InfoGAN-CR [22]. For the VAE-based baseline methods [3, 7, 14, 17, 19], we reproduce the results by the code provided in [10] with the suggested optimized parameters. In particular, we use a batch size of 64 for all experiments and train the model for 30 epochs then select the checkpoint with the highest Explicitness score for evaluation. We use the model architecture suggested in [3] and use a latent dimension equal to 10 for all methods. For  $\beta$ -VAE [14], we train the model with  $\beta$  equal to 4. For Annealed VAE [3], we set the capacity equal to 25. For factor VAE and  $\beta$ -TCVAE, we use the weight of 6.4 for the total correlation term. On the other hand, for baseline methods based on GAN [8, 16, 22], we reproduce the result with the exact parameters and model architecture provided in the corresponding paper and code.

**Results.** The results are presented in Table 1, where we present the average and standard deviation over 10 runs for all metrics, and highlight the highest score in bold, and the second-highest score by an underscore. Our method outperforms the existing methods from two perspectives:

- 1) Our method achieves the best performance across all the evaluation metrics. Specifically, our method outperforms baseline methods by a considerable margin on Explicitness, Modularity and SAP scores. And on the other two metrics JEMMIG and Z-diff scores, even though existing methods already performed well, our method also outperforms baseline methods by a reasonable margin.
- 2) Our method exhibits consistency across the evaluation metrics. Other methods, in contrast, often lack this consistency. For instance, IB-GAN performs well on Explicitness and Modularity Score but performs poorly on JEMMIG and SAP. This inconsistency arises from the fact that different metrics perform their evaluation from different perspectives as introduced in Sect. 1 and the *Metrics* section above. Therefore, the consistency of our method shows the effectiveness of our method in enhancing the quality and generalizability of Disentangled Representation Learning algorithms.

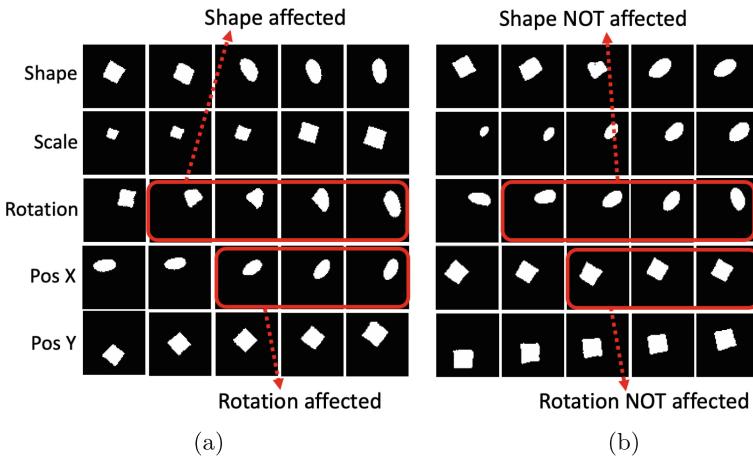
### 3.3 Qualitative Evaluation by Latent Space Traversal Test

While the datasets suitable for quantitative evaluation are limited, we further evaluate our method on other datasets by conducting Latent Space Traversal Tests. **Latent space traversal test** is a commonly used technique to investigate the semantic meaning of latent variables. It traverses one latent variable while keeping all the other variables invariant, and generates a sequence of images with these features. The semantic meaning of the traversed variable can be obtained by inspecting the changes in the images.

In our experiments, since we aim to improve the quality of disentanglement, we examine whether the changes in one variable affect more than one generative factor. If the traversed variable affects **only one** semantic meaningful factor, it

means the quality of disentanglement is desirable. In contrast, if the traversed variable affects **more than one** semantic meaningful factor, it means the factors are still entangled in the latent space.

To this end, we make comparisons between the cases with/without the enforcement of the proposed independence constraint and directly observe the impact of implementing our method. To ensure fair comparisons, all the settings remain the same except for the total correlation loss  $\mathcal{L}_{TC}$  (Eq. 6) in each pair of comparisons. These comparisons are conducted on three datasets: MNIST, FashionMNIST and dSprites. For all three datasets, the settings remain unchanged as introduced in Sect. 3.1, except for the dimension settings of the latent space, because the number of generative factors contained in each dataset is different. This will be further elaborated in each paragraph below.

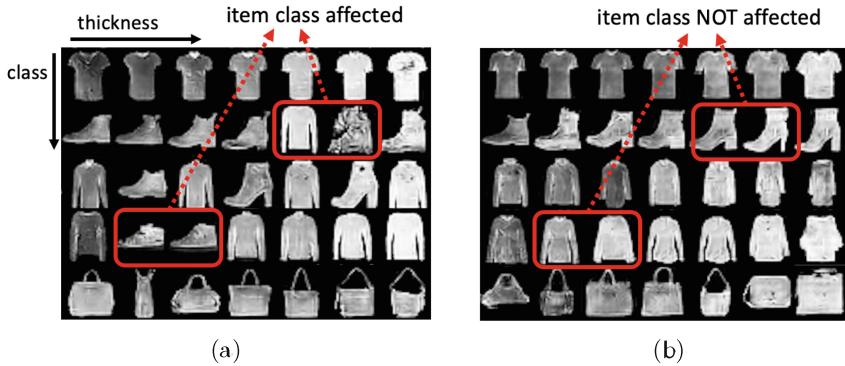


**Fig. 4.** dSprites Samples for comparison. Both models are trained with 5 continuous variables, and 5 noise variables. (a) Without independence constraint, the shape is affected by traversing on rotation and Rotation is affected by traversing on PosX; (b) With independence constraint, these factors are unaffected by traversing on another variable.

**MNIST.** We trained the models with 1 ten-dimensional discrete variable, 2 continuous variables, and 62 noise variables. The ideal outcome of Disentangled Representation Learning is that the data encoder will map the discrete variable to the *digit class*, and the two continuous variables will correspond to the *width* and *rotation* of the digit. The results of our experiment are provided in Fig. 1. In each row, we keep the discrete variable *digit* invariant and traverse on the continuous variable *rotation*. When the model is trained without the independence constraint (Fig. 1a), we observe that: 1) The variables *digit class* and *rotation* are not fully disentangled in the latent space. For example, while we control

the *digit* to be 5 on the fifth row, some 0 and 6 are generated when traversing on *rotation*; and 2) the variables *width* and *rotation* of the digit are not fully disentangled in the latent space. For example, as highlighted in the first row of Fig. 1a, while we only traverse the variable *rotation*, the *width* of the digit is also affected. We highlight this trend on the first row, and similar patterns can be observed on other rows. In contrast, with the enhancement of the independent constraint (Fig. 1b), both *digit* and *width* remain unchanged when traversing on *rotation*.

**dSprites.** We trained the models with 5 continuous variables, and 5 noise variables. Ideally, the five continuous variables will be mapped to the five generative factors of the dataset, which include *Shape*, *Scale*, *Rotation*, *Pos X* and *Pos Y*. The results of our experiment are provided in Fig. 4. On each row of the images, we traverse one factor while keeping all other factors invariant as noted on the left side of each row. When the model is trained without the independence constraint (Fig. 4a), we observe that the factors are entangled in the latent space. For example, on the third row of the image, while traversing the factor of *rotation*, the *shape* of the figures are affected. And on the fourth row, *rotation* is affected while traversing the *Position X*. In contrast, with the enhancement of the independent constraint (Fig. 4b), these factors are not affected, as highlighted by red boxes in Fig. 4.



**Fig. 5.** FashionMNIST Samples for comparison. Both models are trained with 1 ten-dimensional discrete variable, 1 continuous variable, and 62 noise variables. (a) Without constraining on the independence between variables, item type is affected by traversing on thickness. (b) With the constraint on the independence between variables, item type is unaffected while traversing on thickness.

**FashionMNIST.** We trained the models with 1 ten-dimensional discrete variable, 1 continuous variable, and 62 noise variables. Ideally, the discrete variable

and the continuous variable will correspond to the *item class* and the *thickness* of the image. The results of our experiment are provided in Fig. 5. A similar conclusion can be drawn as we did on other datasets above. On each row, we keep the discrete variable *item class* invariant and traverse on the continuous variable *thickness* of the image. When the model is trained without the constraint of the independence between latent variables (Fig. 5a), we observe that the item is affected by variable *thickness*, as highlighted on the second row and the fourth row. In contrast, the item type remains unaffected when the model is trained with the enhancement of the independent constraint (Fig. 5b).

**Summary.** Based on these comparisons, we conclude that our method could consistently enhance the quality of disentanglement. Note that this does not mean our method could always disentangle all the factors perfectly, however, while some factors may still exhibit slight entanglement in the given images, the differences between the cases with/without the independence constraint are nontrivial, which validates the effectiveness of our method.

## 4 Related Works

### 4.1 Disentangled Representation Learning

Disentangled Representation Learning aims to learn a data encoder that can identify true latent variables that are semantically meaningful. [14] suggested that increasing the weight of the KL regularizer in VAE [19] can benefit the quality of disentangled representation learning. [3] proposed that disentanglement quality can be improved by progressively increasing the bottleneck capacity. FactorVAE [17] and  $\beta$ -TCVAE [7] both penalize the total correlation [34] between latent variables, while FactorVAE uses a density-ratio trick for total correlation estimation,  $\beta$ -TCVAE proposed a biased Monte-Carlo estimator to approximate total correlation. Several other methods were also proposed in the paradigm of VAE [11, 18, 20]. However, [23] claimed that unsupervised disentangled representation learning is impossible without inductive biases.

On the other hand, some methods are proposed to learn disentangled representation in the paradigm of GAN [12, 40]. InfoGAN [8] proposed a method to learn disentangled representation by maximizing the mutual information between latent variables and the generated image. InfoGAN-CR [22] claimed that self-supervision techniques can be used to improve the quality of disentanglement. IB-GAN [16] utilized the Information Bottleneck framework for the optimization of GAN. Besides, some methods attempted to learn disentangled representations without utilizing generative models [32].

Recently, diffusion models have been applied to the domain of disentangled representation learning [5, 6, 36]. Additionally, disentangled representation learning has found broad applications in areas such as graph representation learning [21], graph neural architecture search [39], recommendation systems [33, 38], and out-of-distribution generalization [37].

## 4.2 Causal Representation Learning

Recent studies are aimed at connecting the field of causal inference and disentangled representation learning. [31] introduced a causal perspective of disentangled representation learning by modelling the data generation process as a Structural Causal Model (SCM) [26], where they introduced a set of confounders that causally influence the generative factors of observable data. [27] further developed this idea and studied the role of intervention and counterfactual effects. CausalVAE [35] introduced a fully supervised method that builds a Causal Layer to transform independent exogenous factors into causal endogenous factors that correspond to causally related concepts in the observed data. And DEAR [30] proposed a weakly supervised framework, which learns causally disentangled representation with SCM as prior.

## 5 Conclusion

In this paper, we investigated the prior disagreement on the interrelationships between latent variables in Disentangled Representation Learning and proposed a novel method to improve the quality of disentanglement. First, we build a conceptual bridge between epistemology and disentangled representation learning, thus clarifying what should and should not be independent in the latent space by introducing a two-level latent space framework based on interdisciplinary theories. Then, after clarifying the legitimacy of applying the independence constraint on the problem of Disentangled Representation Learning, we introduce a novel method that applies the mutual information constraint and independence constraint within the Generative Adversarial Network (GAN) framework. Experiments show that our method consistently achieves better disentanglement performance on multiple evaluation metrics, and Qualitative Evaluation results show that our method leads to an improved quality for controllable generation. Besides, our paper introduced a novel perspective to apply causal models to the field of representation learning, which facilitates the development of explainability of deep learning and holds potential for wide-ranging applications that value explainability, transparency and controllability.

## References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), 1798–1828 (2013)
2. Berlin, B.: Ethnobiological classification: Principles of categorization of plants and animals in traditional societies, vol. 185. Princeton University Press (2014)
3. Burgess, C.P., et al.: Understanding disentangling in *beta*-VAE. arXiv preprint [arXiv:1804.03599](https://arxiv.org/abs/1804.03599) (2018)
4. Carbonneau, M.A., Zaidi, J., Boilard, J., Gagnon, G.: Measuring disentanglement: a review of metrics. *IEEE Trans. Neural Netw. Learning Syst.* (2022)
5. Chen, H., Zhang, Y., Wang, X., Duan, X., Zhou, Y., Zhu, W.: DisenBooth: disentangled parameter-efficient tuning for subject-driven text-to-image generation. arXiv preprint [arXiv:2305.03374](https://arxiv.org/abs/2305.03374) **3** (2023)

6. Chen, H., Zhang, Y., Wang, X., Duan, X., Zhou, Y., Zhu, W.: DisenDreamer: subject-driven text-to-image generation with sample-aware disentangled tuning. *IEEE Trans. Circ. Syst. Video Technol.* (2024)
7. Chen, R.T., Li, X., Grosse, R.B., Duvenaud, D.K.: Isolating sources of disentanglement in variational autoencoders. In: *Advances in Neural Information Processing Systems*, vol. 31 (2018)
8. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In: *Advances in Neural Information Processing Systems*, vol. 29 (2016)
9. Do, K., Tran, T.: Theory and evaluation metrics for learning disentangled representations. arXiv preprint [arXiv:1908.09961](https://arxiv.org/abs/1908.09961) (2019)
10. Dubois, Y., Kastanov, A., Lines, D., Melman, B.: Disentangling VAE. <http://github.com/YannDubs/disentangling-vae/> (2019)
11. Dupont, E.: Learning disentangled joint continuous and discrete representations. In: *Advances in Neural Information Processing Systems*, vol. 31 (2018)
12. Goodfellow, I., et al.: Generative adversarial networks. *Commun. ACM* **63**(11), 139–144 (2020)
13. Higgins, I., et al.: Towards a definition of disentangled representations. arXiv preprint [arXiv:1812.02230](https://arxiv.org/abs/1812.02230) (2018)
14. Higgins, I., et al.: BETA-VAE: learning basic visual concepts with a constrained variational framework. In: *International Conference on Learning Representations* (2017)
15. Hume, D.: *A treatise of human nature*. Courier Corporation (2003)
16. Jeon, I., Lee, W., Pyeon, M., Kim, G.: IB-GAN: disentangled representation learning with information bottleneck generative adversarial networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 7926–7934 (2021)
17. Kim, H., Mnih, A.: Disentangling by factorising. In: *International Conference on Machine Learning*, pp. 2649–2658. PMLR (2018)
18. Kim, M., Wang, Y., Sahu, P., Pavlovic, V.: Relevance factor VAE: learning and identifying disentangled factors. arXiv preprint [arXiv:1902.01568](https://arxiv.org/abs/1902.01568) (2019)
19. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint [arXiv:1312.6114](https://arxiv.org/abs/1312.6114) (2013)
20. Kumar, A., Sattigeri, P., Balakrishnan, A.: Variational inference of disentangled latent concepts from unlabeled observations. arXiv preprint [arXiv:1711.00848](https://arxiv.org/abs/1711.00848) (2017)
21. Li, H., Wang, X., Zhang, Z., Chen, H., Zhang, Z., Zhu, W.: Disentangled graph self-supervised learning for out-of-distribution generalization. In: *Forty-first International Conference on Machine Learning* (2024)
22. Lin, Z., Thekumparampil, K., Fanti, G., Oh, S.: InfoGAN-CR and modelcentrality: self-supervised model training and selection for disentangling GANs. In: *International Conference on Machine Learning*, pp. 6127–6139. PMLR (2020)
23. Locatello, F., et al.: Challenging common assumptions in the unsupervised learning of disentangled representations. In: *International Conference on Machine Learning*, pp. 4114–4124. PMLR (2019)
24. Matthey, L., Higgins, I., Hassabis, D., Lerchner, A.: dSprites: disentanglement testing sprites dataset. <https://github.com/deepmind/dsprites-dataset/> (2017)
25. Murphy, G.: *The big book of concepts*. MIT Press (2004)
26. Pearl, J.: *Causality*. Cambridge University Press (2009)

27. Reddy, A.G., Balasubramanian, V.N., et al.: On causally disentangled representations. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 8089–8097 (2022)
28. Ridgeway, K., Mozer, M.C.: Learning deep disentangled embeddings with the f-statistic loss. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
29. Rips, L.J., Shoben, E.J., Smith, E.E.: Semantic distance and the verification of semantic relations. *J. Verbal Learn. Verbal Behav.* **12**(1), 1–20 (1973)
30. Shen, X., Liu, F., Dong, H., Lian, Q., Chen, Z., Zhang, T.: Weakly supervised disentangled generative causal representation learning. *J. Mach. Learn. Res.* **23**, 1–55 (2022)
31. Suter, R., Miladinovic, D., Schölkopf, B., Bauer, S.: Robustly disentangled causal mechanisms: validating deep representations for interventional robustness. In: International Conference on Machine Learning, pp. 6056–6065. PMLR (2019)
32. Wang, T., Yue, Z., Huang, J., Sun, Q., Zhang, H.: Self-supervised learning disentangled group representation as feature. *Adv. Neural. Inf. Process. Syst.* **34**, 18225–18240 (2021)
33. Wang, X., Pan, Z., Zhou, Y., Chen, H., Ge, C., Zhu, W.: Curriculum co-disentangled representation learning across multiple environments for social recommendation. In: International Conference on Machine Learning, pp. 36174–36192. PMLR (2023)
34. Watanabe, S.: Information theoretical analysis of multivariate correlation. *IBM J. Res. Dev.* **4**(1), 66–82 (1960)
35. Yang, M., Liu, F., Chen, Z., Shen, X., Hao, J., Wang, J.: CausalVAE: disentangled representation learning via neural structural causal models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9593–9602 (2021)
36. Yang, T., Wang, Y., Lv, Y., Zheng, N.: DisDiff: unsupervised disentanglement of diffusion probabilistic models. arXiv preprint [arXiv:2301.13721](https://arxiv.org/abs/2301.13721) (2023)
37. Yoo, H., Lee, Y.C., Shin, K., Kim, S.W.: Disentangling degree-related biases and interest for out-of-distribution generalized directed network embedding. In: Proceedings of the ACM Web Conference 2023, pp. 231–239 (2023)
38. Zhang, Y., Wang, X., Chen, H., Zhu, W.: Adaptive disentangled transformer for sequential recommendation. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 3434–3445 (2023)
39. Zhang, Z., et al.: Disentangled continual graph neural architecture search with invariant modular supernet. In: Forty-first International Conference on Machine Learning (2024)
40. Zhu, X., Xu, C., Tao, D.: Where and what? Examining interpretable disentangled representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5861–5870 (2021)



# Utilizing Small and Large Spectral Radii for Appropriate Reservoir Computing Design

Bungo Konishi<sup>(✉)</sup>, Akira Hirose, and Ryo Natsuaki

The University of Tokyo, Tokyo 113-8656, Japan  
`eis.ut.konishi5282@gmail.com`, `ahirose@ee.t.u-tokyo.ac.jp`,  
`natsuaki@eis.t.u-tokyo.ac.jp`

**Abstract.** Echo state network (ESN) is a family of reservoir computing (RC) realizing energy-efficient learning. In an ESN, a reservoir maps time-series input signals nonlinearly to high-dimensional space, which is utilized for performing various tasks. Many studies have analyzed reservoirs on stability, nonlinearity, and memory capacity. Among them, the memory capacity has been analyzed and evaluated as the integral of a determination coefficient, which is obtained as a function of time delay. In this paper, by focusing on the determination-coefficient function (DCF), we elucidate the relationship between memory length and respective hyperparameters, namely, the number of neurons, activation function, network connectivity, and the spectral radius in a reservoir. We conduct two experiments assuming tasks of multi-frequency signal separation and multi-variance noise grouping. The experiments show that the DCF profile is of significant importance in determined by the degree of concentration of the reservoir state to the spherical surface in the high dimension information space depending on the spectral radius. The results also demonstrate that ESNs organize proper clusters, having a high representation ability even with a spectral radius smaller or larger than unity. These findings help us to determine the spectral radius in reservoirs.

**Keywords:** Echo state network · reservoir computing · memory capacity

## 1 Introduction

Recurrent neural networks (RNNs) are useful for time series forecasting and spatial sequence estimation [4, 9]. Real-time recurrent learning (RTRL) and back-propagation through time (BPTT) are typically employed to train weights in a network. However, these methods have serious problems of the large computational cost and the vanishing gradients.

Reservoir computing (RC), a special type of RNNs, can overcome these issues. RC includes echo state networks (ESNs) proposed by Jaeger in 2001 [11], and

liquid state machine (LSM) proposed by Maass in 2002 [13]. ESNs basically consist of three parts, namely, an input terminal layer, a hidden layer called "reservoir", and a readout layer. A reservoir has numerous neurons having self- and mutual-connections, and thus input signals and internal states are mapped nonlinearly to a high-dimensional space. Like the principle of the support vector machine (SVM), we are able to find a suitable linear separating hyperplane in the reservoir space for each task without training recurrent weights. This means that we only calculate the weights connected to the readout part with very short time.

To provide ESNs with appropriate characteristics such as stability and nonlinearity, we need to determine a network structure and statistical properties of fixed weights. In particular, the echo state property (ESP) [11], the fading memory property (FMP) [14], and the pairwise separation property (SP) are crucial to be designed appropriately. When the network has the ESP, the internal states converge to the values corresponding to a steady input signal regardless of the initial reservoir states. The FMP, primarily discussed in the context of the LSM, is a property for fading input signals [6]. These properties play an important role in evaluating stability of dynamical systems. Finally, when the network satisfies the SP, the internal reservoir states converge to different states corresponding to different input sequences. This property can measure complexity and nonlinearity of the network.

Depending on a practical application task, these characteristics should be exploited, but such a method has not been established [15]. In addition, memory capacity (MC) of RC is also important for learning of long-term and short-term dependencies. Since ESNs were proposed for the first time, many studies have presented indicators for measuring MC and structures for improving the maximum memory length. Many studies have calculated MC as the integral of determination coefficients for delayed inputs and outputs [10]. For instance, Han et al., using this measure, investigated the relationships between MC and hyperparameters such as input scaling and network density, and then proposed an expanded clustered RC model [8]. While the majority of conventional memory analyses have employed only the integral of the coefficients, an examination of the shape of the function with the delay as the independent variable may prove more fruitful for understanding long- and short-term memory characteristics. Henceforth, this shape shall be referred to as the determination coefficient function (DCF). Moreover, the dependencies of hyperparameters on MC have not been completely investigated.

In this paper, we first compute DCFs for various hyperparameters, such as the number of neurons and the spectral radius in a reservoir, and also calculate their integrals, or MC. As a result, we reveal that the DCF profile is of significant importance in determined by the degree of concentration of the reservoir state to the spherical surface in the high dimension information space depending on the spectral radius. We then conduct two experiments assuming tasks of multi-frequency signal separation and multi-variance noise grouping. Experi-

ments demonstrate that ESNs organize proper clusters, having a high representation ability even with a spectral radius smaller or larger than unity.

The contribution of this paper is outlined below.

- The relationships between DCF/MC and the respective hyperparameters, namely the number of neurons, activation functions, connectivity, and spectral radius in a reservoir, were elucidated.
- DCFs with both small and large spectral radii exhibit disparate profiles, even when the integral value (MC) is nearly identical. These profiles are shaped by the concentration on the surface of a sphere in high dimension.
- A large spectral radius is useful for suppression of the effects of local fluctuations, while a small spectral radius is desirable for tasks requiring ultra-short-term memory.

## 2 Echo State Networks

### 2.1 Basic Network Dynamics

The dynamics in a reservoir represents the following equation.

$$\mathbf{x}(t) = (1 - \alpha)\mathbf{x}(t - 1) + \alpha \tanh(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}_{\text{res}}\mathbf{x}(t - 1) + \boldsymbol{\delta}) \quad (1)$$

where  $\mathbf{u}(t) \in \mathbb{R}^{N_{\text{in}}}$  and  $\mathbf{x}(t) \in \mathbb{R}^{N_{\text{res}}}$  are the input signals and the reservoir signals at step  $t$ , respectively. The input-reservoir weights  $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_{\text{res}} \times N_{\text{in}}}$ , the reservoir recurrent weights  $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N_{\text{res}} \times N_{\text{res}}}$ , and the reservoir bias  $\boldsymbol{\delta} \in \mathbb{R}^{N_{\text{res}}}$  are the fixed hyperparameters without learning. Note that  $N_{\text{in}}$  and  $N_{\text{res}}$  are the number of neurons in the input layer and the reservoir, respectively.

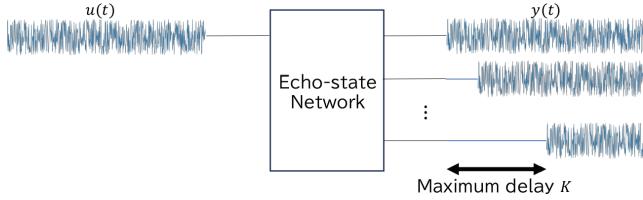
In (1),  $\alpha(0 \leq \alpha \leq 1)$ , known as “leaky parameter” is a hyperparameter in order to adjust a speed of the dynamics. When  $\alpha$  is close to one, the current reservoir states are strongly affected by the input data and the other reservoir neurons. In contrast, when  $\alpha$  is almost zero, the states are unchanging. In this paper, we set  $\alpha$  as one, investigating the performance of the memory capacity as follows.

$$\mathbf{x}(t) = \tanh(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}_{\text{res}}\mathbf{x}(t - 1) + \boldsymbol{\delta}) \quad (2)$$

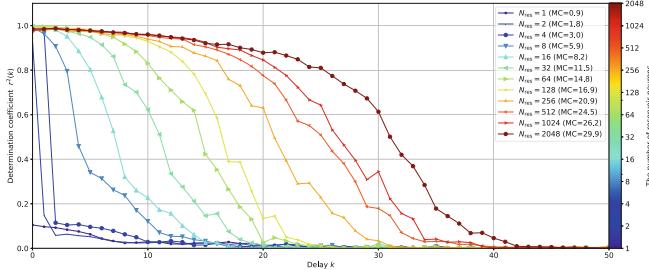
The output values  $\mathbf{y}(t) \in \mathbb{R}^{N_{\text{out}}}$  at step  $t$  is obtained by the following affine transformation.

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}}\mathbf{x}(t) + \mathbf{b}_{\text{out}} \quad (3)$$

where  $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_{\text{out}} \times N_{\text{res}}}$  is the reservoir-output weights,  $\mathbf{b}_{\text{out}} \in \mathbb{R}^{N_{\text{out}}}$  is the output bias, and  $N_{\text{out}}$  is the number of the output neurons. The parameters  $\mathbf{W}_{\text{out}}$  and  $\mathbf{b}_{\text{out}}$  are optimized by the Tikhonov regularization or gradient descent methods for online learning.



**Fig. 1.** Task for measuring memory capacity of ESNs.



**Fig. 2.** Determination coefficients and memory capacity in the number of neurons in a reservoir ( $N_{\text{res}}$  is 1 to 2048) at each delay  $k$ .

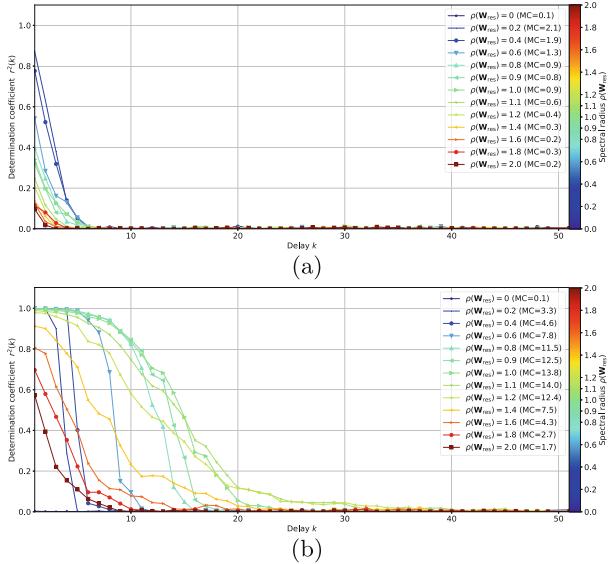
## 2.2 Echo State Property

Echo state property (ESP) is a property that guarantees stability of internal dynamics of a reservoir [16]. Ensuring this property, the network is guaranteed to have asymptotic stability. Some researches have provided the necessary conditions and the sufficient conditions by adjusting the absolute maximum eigenvalue of a reservoir weight matrix  $\rho(\mathbf{W}_{\text{res}})$ , called the spectral radius [3, 16]. However, most of conditions with full mathematical rigor are too strict on practical use. Thus, a spectral radius is generally determined as slightly smaller than unity on the assumption of  $\rho(\mathbf{W}_{\text{res}}) < 1$  [12]. Investigating the relationship between spectral radius and memory, we may be able to design a network suitable for a specific task and input signals.

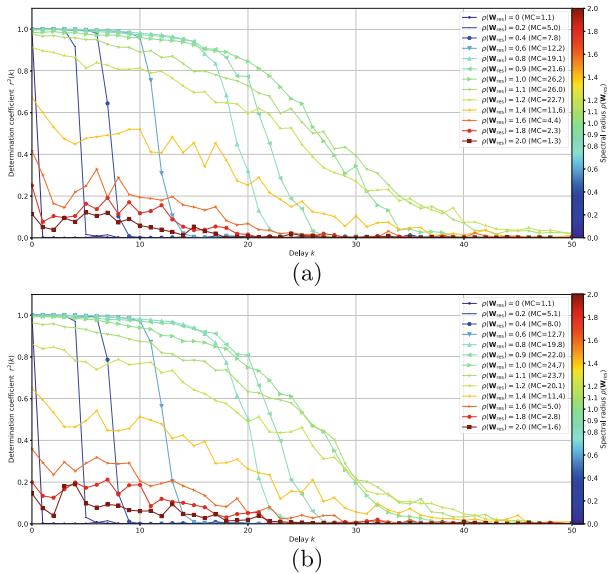
## 3 Analyses of Memory Capacity in Echo State Networks

### 3.1 Experimental Setups

We conduct an experiment in order to measure the maximum memory capacity (MC) of ESNs. In this experiment, we estimate  $K$  delayed output signals  $\mathbf{y}_k(t) = \{u(t), \dots, u(t-k), \dots, u(t-K)\}$  at the same timing, once giving a network an input signal  $u(t)$  as shown in Fig. 1. This input signal  $u(t)$  follows a uniform distribution in the range of [-0.5, 0.5]. We also elucidate the relations between the memory performance and the network parameters, such as the number of neurons and the spectral radius in a reservoir. A method using the determination coefficients



**Fig. 3.** Determination coefficients and memory capacity in the spectral radius ( $\rho(\mathbf{W}_{\text{res}})$  is 0 to 2.0) at each delay  $k$  using the activation functions: (a) Heaviside and (b) hyperbolic tangent functions.



**Fig. 4.** Determination coefficients and memory capacity in the spectral radius ( $\rho(\mathbf{W}_{\text{res}})$  is 0 to 2.0) at each delay  $k$  with the network connectivities of (a) 100% and (b) 5%.

is being the way to compute the MC [10]. The determination coefficient function  $r^2(k)$  (DCF) is defined as below.

$$r^2(k) = \frac{\text{Cov}^2(u(t - k), \mathbf{y}_k(t))}{\text{Var}(u(t - k))\text{Var}(\mathbf{y}_k(t))}. \quad (4)$$

where  $\mathbf{u}(t)$  is the input signals,  $\mathbf{y}_k(t)$  is the delayed output signals, and  $k$  denotes the amount of delay. For integrating the DCF, the MC is calculated as follows.

$$\text{MC} = \sum_k^K r^2(k) \quad (5)$$

It has been known that MC is theoretically maximized by a reservoir with the ring topology and the identity activation function [10] and then the maximum MC equals the number of reservoir neurons:  $\max(\text{MC}) = N_{\text{res}}$ . In addition, methods using a directed acyclic network [8] and an orthogonal matrix [5] have also been proposed. However, from a practical perspective, the strong structural constraints may degrade nonlinearity and design flexibility. In this section, we analyze the dependencies on some hyperparameters, using an unconstrained fully-connected network with hyperbolic tangent activation function.

### 3.2 Analyses of Relationships Between Memory Capacity and ESN Hyperparameters

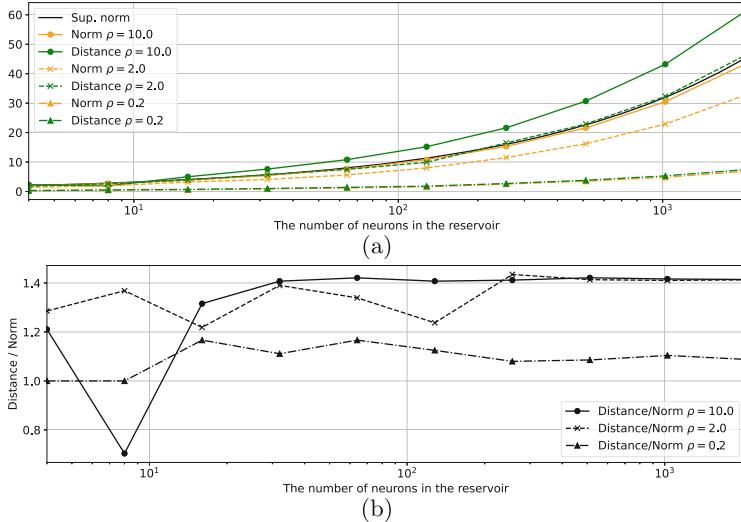
First, we analyze the relation between memory and the number of neurons in a reservoir. Figure 2 shows the determination coefficients at each delay  $k$  in each of the number of reservoir neurons ( $N_{\text{res}}$  is 1 to 2048). When the delay  $k$  is large, its determination coefficient, namely local memory capacity, becomes small. As shown in the figure, increment of the number of the neurons  $N_{\text{res}}$  brings the improvement of the memory performance monotonically.

Next, we visualize the memory dependency on the spectral radius and the activation function. Figure 3 depicts the determination coefficients, using the activation functions: (a) Heaviside and (b) hyperbolic tangent functions, at each delay  $k$  in each of the spectral radius ( $\rho(\mathbf{W}_{\text{res}})$  is 0 to 2.0). We show that the discrete-type activation works insufficiently, compared with the continuous-type function. Also, Fig. 3(b) shows that the memory performance is maximized by the spectral radius  $\rho(\mathbf{W}_{\text{res}})$  of around 1.0.

Finally, we show the memory dependence on the spectral radius and the network connectivity. Figure 4 represents the determination coefficients at each delay  $k$  in the spectral radius ( $\rho(\mathbf{W}_{\text{res}})$  is 0 to 2.0) with the network connectivities of (a) 100% and (b) 5%. These figures present that the connectivity hardly affects the performance. Also, they reveal that the MC is maximized in the spectral radius of 1.0 and the performance becomes to be degraded in both cases of the spectral radius closer to 0 and larger than 1.0. However, these cases differ in how to degrade the performance, that is, the graph shapes. We next focus on the qualitative transition and elucidate the reason in terms of the field of high-dimensional statistical analysis.

### 3.3 High Dimensional Statistical Analyses of Reservoir Dynamics

Here, we discuss the qualitative transition when the spectral radius changes. we first calculate reservoir vector norms  $\|\mathbf{x}(t)\|_2$ , or distances between the reservoir states and the origin, and distances between the last and the current reservoir states  $\|\mathbf{x}(t) - \mathbf{x}(t-1)\|_2$ . Additionally, we defined the temporally averaged norm and the temporally averaged distance.



**Fig. 5.** The transition of (a) the averaged vector norms and the distances between two states and (b) their proportions in the spectral radius ( $\rho(\mathbf{W}_{\text{res}}) = 0.2, 2.0$ , and  $10.0$ ).

Figure 5 shows the transition of (a) the temporally averaged norms and the temporally averaged distances and (b) their proportions (Distance/Norm) in the spectral radii  $\rho(\mathbf{W}_{\text{res}})$  of 0.2, 2.0, and 10.0. This figure demonstrates that, in the case of the large spectral radius, its proportion converges to  $\sqrt{2}$ . This fact has been also mathematically proved by the concentration on the surface of a sphere in the field of high-dimensional statistical analysis [7]. Some of the high dimensional geometric representations with regard to the distance and the angle are described as follows.

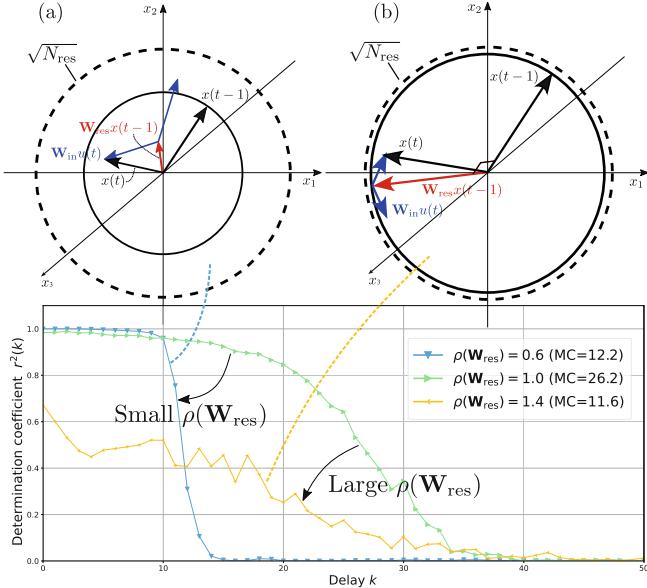
$$\|\mathbf{x}_1 - \mathbf{x}_2\| = \sqrt{2d} + O_p(1) \quad \text{as } d \rightarrow \infty \quad (6)$$

$$\text{Angle}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\pi}{2} + O_p(d^{-1/2}) \quad \text{as } d \rightarrow \infty \quad (7)$$

where  $\mathbf{x}_i$  is high-dimensional data and  $d$  indicates the number of dimensions. Assuming that the number of reservoir neurons is immense, we can derive the proportion using (6) by replacing  $d$  with  $N_{\text{res}}$ , as shown in (8). This analysis

demonstrates that the convergence value in Fig. 5(b) agrees with that of the theory.

$$\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{\|\bar{\mathbf{x}}\|} = \frac{\sqrt{2N_{\text{res}}}}{\sqrt{N_{\text{res}}}} + O_p\left(\frac{1}{\sqrt{N_{\text{res}}}}\right) = \sqrt{2} \quad \text{as} \quad N_{\text{res}} \rightarrow \infty \quad (8)$$



**Fig. 6.** Schematic illustration of the processes of qualitative memory deterioration with (a) smaller and (b) larger spectral radii than unity.

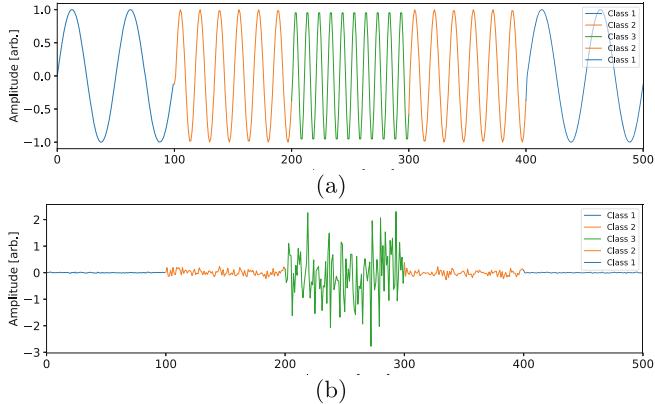
Moreover, as shown in Fig. 4(a), we found that the DCF profiles with a small and a large spectral radii are different each other. Figure 6 shows the schematic illustrations in the case of spectral radii (a) smaller ( $\rho(\mathbf{W}_{\text{res}}) = 0.6$ ) and (b) larger ( $\rho(\mathbf{W}_{\text{res}}) = 1.4$ ) than unity.

Noteworthy is the fact that the integrals of the DCFs with small and large spectral radius are almost the same, although these DCF profiles are completely different.

As shown in Fig. 6(a), the very past signals vanish immediately because the input term  $\mathbf{W}_{\text{in}} \mathbf{u}(t)$  gives greater influence than the feedback term  $\mathbf{W}_{\text{res}} \mathbf{x}(t)$ . This leads to small memory capacity. On the other hand, as shown in Fig. 6(b), the effects of the input signals  $\mathbf{u}(t)$  becomes relatively small because the surface of the hypersphere is completely wider than the internal space. This causes the overall performance degradation.

The spectral radius  $\rho(\mathbf{W}_{\text{res}})$  has been being generally determined as around unity by a criterion only of the stability. However, the memory performance also

has the possibility of being related to the spectral radius. In the next section, we provide the way to decide the spectral radius corresponding to a specific task on the basis of these findings.



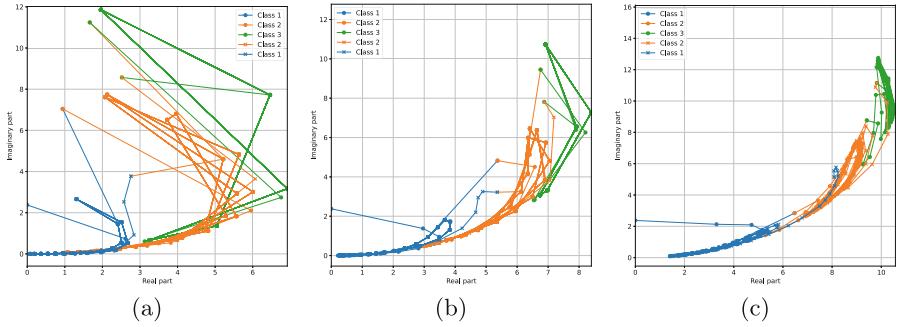
**Fig. 7.** Input signals on the tasks of (a) separating signals with multiple frequencies of 0.02, 0.06, and 0.1 [Hz] and (b) grouping random data with different variances of  $0.01^2$ ,  $0.1^2$ , and  $1^2$ .

**Table 1.** Hyperparameters of the ESNs for Tasks I and II.

Parameter	Value
The number of input terminals	$N_{\text{in}}$ 1
The number of reservoir neurons	$N_{\text{res}}$ 1024
Activation function in reservoir	$f$ <b>tanh</b>
Leaking rate	$\alpha$ 1
Input bias	$\delta$ <b>0</b>
Connectivity in reservoir	100%
Total sequential length	$T$ 500

#### 4 Simulations of Separating Multi-Frequency and Multi-Variance Signals

In this section, we conduct two experiments using multi-frequency signals and multi-variance noise data and visualize internal states in a high-dimensional reservoir. We aim to compare and elucidate the network behavior in cases of two types of profiles with a small and a large spectral radii. The list of hyperparameters used in these tasks is shown in Table 1.



**Fig. 8.** Two-dimensional mappings on Task I for visualizing the internal reservoir states by the dimension reduction. The indicators are complex-values that consist of the distances between the reservoir states  $\|\mathbf{x}(t) - \mathbf{x}(t-1)\|$  and the angles  $\text{Angle}(\mathbf{x}(t), \mathbf{x}(t-1))$  with the specific spectral radii of (a) 0.6, (b) 1.0, and (c) 1.4. Each color corresponds to that of the input signals in Fig. 7(a).

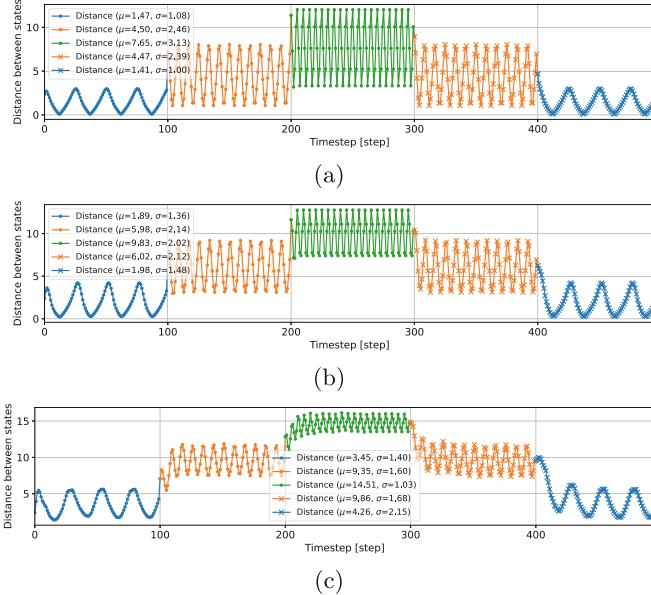
#### 4.1 Task I: Multi-frequency Signal Separation

Figure 7(a) is an input signal  $u(t)$ , used in this experiment, with multiple frequencies of 0.02 Hz, 0.06 Hz, and 0.1 Hz. Each color corresponds to the frequencies. We analyze the transition of reservoir states with high dimension using some measures, giving this input signals. Except for the initial transient, the reservoir states basically have a specific norm by the concentration on the surface of a sphere, as shown in the previous section. Thus, in order to accentuate the behavior of input signals, we calculate the distances and angular displacements between reservoir states at steps  $t-1$  and  $t$  by removing the influence of the state vector norms.

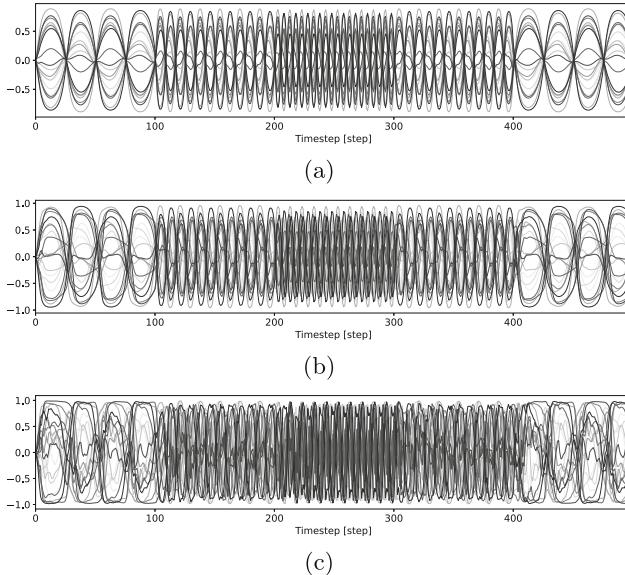
Figure 8 shows two-dimensional mappings on Task I for visualizing the internal reservoir states by the dimension reduction. The indicators are complex-values that consist of the distances between the reservoir states  $\|\mathbf{x}(t) - \mathbf{x}(t-1)\|$  and the angles  $\text{Angle}(\mathbf{x}(t), \mathbf{x}(t-1))$  with the specific spectral radii of (a) 0.6, (b) 1.0, and (c) 1.4. The colored lines correspond to the frequencies of the input signals in Fig. 7(a).

As shown in Fig. 8(a), the small spectral radius makes these indicators, which represents reservoir states, overlap between classes, or clusters. Therefore, with a small spectral radius, it is seemingly difficult for the input signals to be classified on the basis of differences in frequency. On the other hand, in Figs. 8(b) and (c), the larger spectral radii make indicators separate the data featured by frequency more clearly. Moreover, in this task I, the large scale of 1.4, which has been usually avoided in many applications, has rather clearly separated signals than that of unity. Also each class of Figs. 8(c) has obviously independent periodic orbits.

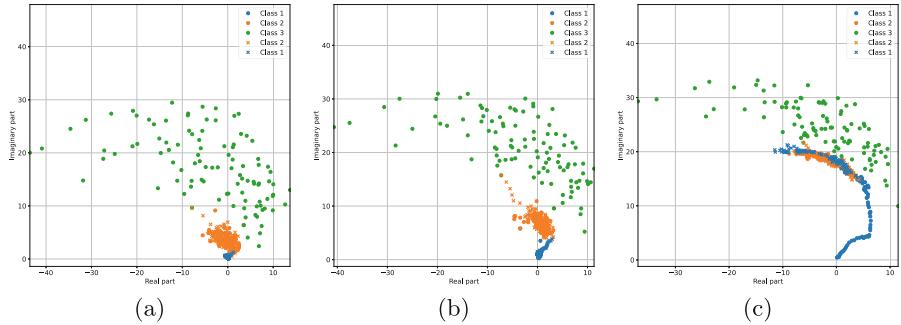
Figure 9 presents the distances between reservoir states with the spectral radii of (a) 0.6, (b) 1.0, and (c) 1.4. Each color corresponds to frequencies of the input signals in Fig. 7(a). This figure also shows that some data points overlap



**Fig. 9.** The distances between the reservoir states with the spectral radii of (a) 0.6, (b) 1.0, and (c) 1.4 on Task I. Each color corresponds to those of the input signals in Fig. 7(a).



**Fig. 10.** The reservoir state values with the specific spectral radii of (a) 0.6, (b) 1.0, and (c) 1.4 on Task I. Each curve corresponds to the reservoir neurons.



**Fig. 11.** Two-dimensional mappings on Task II for visualizing the internal reservoir states by the dimension reduction. The indicators are complex-values that consist of the distances between the reservoir states  $\|\mathbf{x}(t) - \mathbf{x}(t-1)\|$  and the angles  $\text{Angle}(\mathbf{x}(t), \mathbf{x}(t-1))$  with the specific spectral radii of (a) 0.8, (b) 1.0, and (c) 1.2. Colored points correspond to the standard deviations of the inputs in Fig. 7(b).

other clusters. Therefore, it is difficult to separate frequencies with low spectral radii. Also, as in Fig. 9(c), the input signals are separable only from distances when the spectral radius is large.

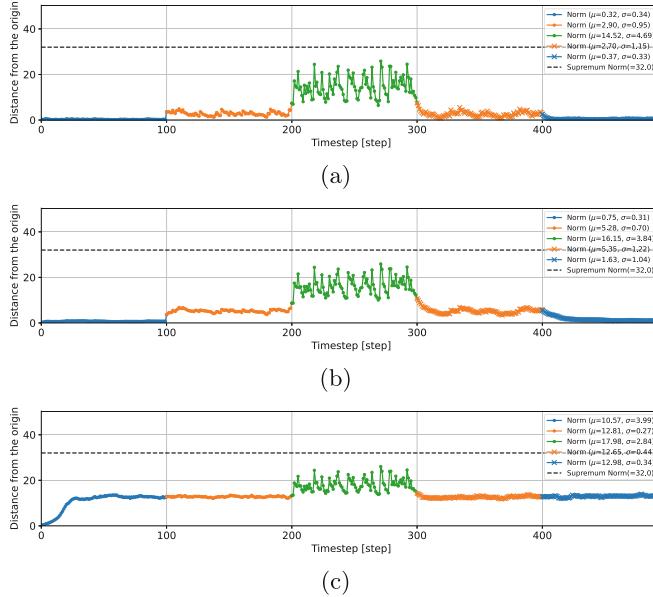
Additionally, we examine the reservoir state values for each spectral radius. Figure 10 shows the reservoir states with the spectral radii of (a) 0.6, (b) 1.0, and (c) 1.4. A line indicates the state values of a corresponding reservoir neuron. The reservoir signals with a small spectral radius in Fig. 10(a) are driven by strong external inputs, and most of these reservoir states are in phase and affected only by the current inputs. Contrarily, those with a large spectral radius in Fig. 10(c) are driven by strong recurrent dynamics, and these states seem to have richer nonlinearities containing various cycles and waveforms.

Finally, we discuss the principle of better frequency separation with large spectral radius. The frequency classification task requires us to suppress local fluctuations and memorize global, or long cycle, history. In other words, the value of ultra-short-term memory is low, and that of wide memory is very high. These findings are consistent with the discussion of the DCF profiles in Sect. 3.3. Next, we introduce a case of obtaining more preferable results with a small spectral radius.

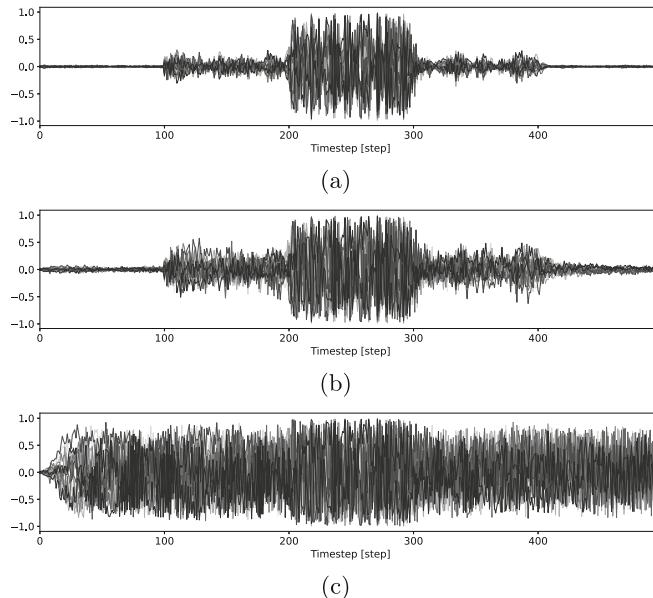
## 4.2 Task II: Multi-variance Gaussian Noise Separation

In the Task II, we conduct an experiment to separate multiple Gaussian noise inputs with the same mean but different standard deviations of 0.01, 0.1, and 1, as shown in Fig. 7(b).

Similarly with Task I, we first analyze the distances and angles between reservoir states at steps  $t - 1$  and  $t$ . Figure 11 shows that the two-dimensional mappings for visualizing the internal reservoir states by the dimension reduction. The indicators are complex-values that consist of the distances between the



**Fig. 12.** The vector norms of the reservoir states with the specific spectral radii of (a) 0.8, (b) 1.0, and (c) 1.2 on Task II. Each color corresponds to the variances of input signals in Fig. 7(b).



**Fig. 13.** The reservoir state values with the specific spectral radii of (a) 0.8, (b) 1.0, and (c) 1.2 on Task II. Each curve corresponds to the reservoir neurons.

reservoir states  $\|\mathbf{x}(t) - \mathbf{x}(t-1)\|$  and the angles  $\text{Angle}(\mathbf{x}(t), \mathbf{x}(t-1))$  with the specific spectral radii of (a) 0.8, (b) 1.0, and (c) 1.2. Colored points correspond to the standard deviations of the inputs in Fig. 7(b).

Figures 11(a) and (b) present that the small spectral radii make the different groups separate each other. On the other hand, when the spectral radius is large, Fig. 11(c) demonstrates that the data points with different variances overlap each other. These experimental results show that the radius of a hypersphere changes because of the difference in variance. Actually, also in the field of high-dimensional statistical analysis, the geometrical quadratic discriminant analysis utilizes this principle [1, 2]. Thus, we next focus on the vector norms corresponding to the radius of a hypersphere.

Figure 12 shows the norms of the reservoir states when the spectral radii are (a) 0.8, (b) 1.0, and (c) 1.2. Each color corresponds to the variances of the inputs in Fig. 7(b). Figure 12(c) shows that two types of data points generated by a large spectral radius are inseparable. In contrast, as shown in Figs. 12(a) and (b), we can separate these data into the three classes while the differences are slight.

Furthermore, we visualize the states in the reservoir directly. Figure 13 shows the reservoir state values with the specific spectral radii of (a) 0.8, (b) 1.0, and (c) 1.2. Each curve corresponds to neurons in the reservoir. When the spectral radius is large, the amplitude of reservoir signals is expanded inevitably. Therefore, even if the input variance is small, we hardly separate signals into a group.

Finally, we discuss the reason why the small spectral radius is appropriate for task II. This task, for estimating the scale of variance, requires a network to preserve the last states clearly because the changes between the last states and the current states are crucial. Since the network needs to keep ultra-short-term memory accurately, the DCF profile with a small spectral radius is ideal, as described in Sect. 3.3 and Fig. 6.

## 5 Conclusion

In this paper, we elucidated the relationships between DCF/MC and respective hyperparameters, namely, the number of neurons, activation functions, connectivity, and spectral radius in a reservoir. The results show that DCFs with small and large spectral radii have different profiles even for nearly the same integral value (MC). We also found that these profiles are shaped by the concentration on the surface of a sphere in high dimension. Finally, the importance of the DCF profiles was explained through two experiments assuming multi-frequency signals separation task and multi-variance noise grouping task. Experiments demonstrate that a large spectral radius is useful for the former task, which should suppress the effects of local fluctuations, while a small spectral radius is desirable for the latter task, which requires ultra-short-term memory. Future works will be directed toward analysing a leaking rate and extending these analyses to deep ESNs.

## References

1. Aoshima, M., Yata, K.: Two-stage procedures for high-dimensional data. *Seq. Anal.* **30**(4), 356–399 (2011)
2. Aoshima, M., Yata, K.: Geometric classifier for multiclass, high-dimensional data. *Seq. Anal.* **34**(3), 279–294 (2015)
3. Buehner, M., Young, P.: A tighter bound for the echo state property. *IEEE Trans. Neural Networks* **17**(3), 820–824 (2006)
4. Cho, K., van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pp. 103–111. Association for Computational Linguistics, Doha, Qatar (2014). <https://doi.org/10.3115/v1/W14-4012>, <https://aclanthology.org/W14-4012>
5. Farkaš, I., Bosák, R., Gergel', P.: Computational analysis of memory capacity in echo state networks. *Neural Networks* **83**, 109–120 (2016). <https://doi.org/10.1016/j.neunet.2016.07.012>, <https://www.sciencedirect.com/science/article/pii/S0893608016300946>
6. Grigoryeva, L., Ortega, J.P.: Differentiable reservoir computing. *J. Mach. Learn. Res.* **20**(179), 1–62 (2019)
7. Hall, P., Marron, J.S., Neeman, A.: Geometric representation of high dimension, low sample size data. *J. Royal Statist. Soc. Ser. B (Statist. Methodol.)* **67**(3), 427–444 (2005)
8. Han, X., Zhao, Y., Small, M.: Revisiting the memory capacity in reservoir computing of directed acyclic network. *Chaos: Interdisc. J. Nonlinear Sci.* **31**(3) (2021)
9. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. *Neural Computation* **9**(8), 1735–1780 (1997). <https://doi.org/10.1162/neco.1997.9.8.1735>
10. Jaeger, H.: Short term memory in echo state networks. Technical Report GMD Report **152** (2001)
11. Jaeger, H.: The “echo state” approach to analysing and training recurrent neural networks—with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148 (2001)
12. Lukoševičius, M.: A practical guide to applying echo state networks. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) *Neural Networks: Tricks of the Trade*: Second Edition, pp. 659–686. Springer, Berlin, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-35289-8\\_36](https://doi.org/10.1007/978-3-642-35289-8_36)
13. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* **14**(11), 2531–2560 (2002). <https://doi.org/10.1162/089976602760407955>
14. Maass, W., Natschläger, T., Markram, H.: Fading memory and kernel properties of generic cortical microcircuit models. *J. Physiol.-Paris* **98**(4–6), 315–330 (2004)
15. Yan, M., Huang, C., Bienstman, P., Tino, P., Lin, W., Sun, J.: Emerging opportunities and challenges for the future of reservoir computing. *Nat. Commun.* **15**(1), 2056 (2024)
16. Yildiz, I.B., Jaeger, H., Kiebel, S.J.: Re-visiting the echo state property. *Neural Netw.* **35**, 1–9 (2012)



# Noisy Deep Ensemble: Accelerating Deep Ensemble Learning via Noise Injection

Shunsuke Sakai<sup>(✉)</sup> ID, Shunsuke Tsuge ID, and Tatsuhito Hasegawa ID

Graduate School of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui City,  
Fukui 910-8507, Japan  
[mf240599@g.u-fukui.ac.jp](mailto:mf240599@g.u-fukui.ac.jp)

**Abstract.** Neural network ensembles is a simple yet effective approach for enhancing generalization capabilities. The most common method involves independently training multiple neural networks initialized with different weights and then averaging their predictions during inference. However, this approach increases training time linearly with the number of ensemble members. To address this issue, we propose the novel “Noisy Deep Ensemble” method, significantly reducing the training time required for neural network ensembles. In this method, a *parent model* is trained until convergence, and then the weights of the *parent model* are perturbed in various ways to construct multiple *child models*. This perturbation of the *parent model* weights facilitates the exploration of different local minima while significantly reducing the training time for each ensemble member. We evaluated our method using diverse CNN architectures on CIFAR-10 and CIFAR-100 datasets, surpassing conventional efficient ensemble methods and achieving test accuracy comparable to standard ensembles. Code is available at <https://github.com/TSTB-dev/NoisyDeepEnsemble>

**Keywords:** Ensemble Learning · Noise Injection · Weight Perturbation

## 1 Introduction

Deep neural networks have achieved remarkable results in various fields, such as image recognition, natural language processing, and speech recognition. Their success can be attributed to their exceptionally high representation capacity. However, deep neural networks often lead to overfitting, particularly when the training data size is small. Consequently, the assessment of deep neural networks primarily focuses on their capability to effectively predict outcomes on new, unseen data.

When multiple neural networks are trained independently, the randomness in the initialization of weights and the selection of mini-batches in SGD leads each network to learn different feature representations. Ensemble learning involves combining the predictions of multiple models to produce a more accurate prediction during inference. By integrating the predictions of multiple models, ensemble

learning improves accuracy and uncertainty estimation performance compared to using a single model [5, 17, 19]. Ensemble learning is known to be more effective when each ensemble member possesses comparably high accuracy and makes mistakes on different samples [12].

Ensemble learning is a simple yet effective approach for improving generalization performance, but it faces the issue of training time increasing linearly with the number of ensemble members. This issue is especially significant for deep neural networks, as training just one model can require weeks to months, which restricts the practical use of ensemble learning. Additionally, there are reported scaling laws indicating that the performance of neural networks improves with increased data size, model size, and computation [14]. Furthermore, a phenomenon known as “Grokking”, where continued training beyond the point of sufficient loss reduction can enhance generalization performance, has been observed in specific problems [13, 25]. These factors contribute to the increased training time for a single model, making realizing their ensembles more challenging.

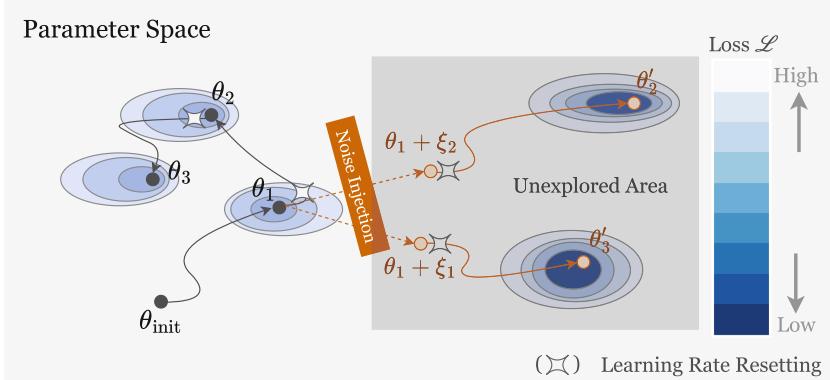
Several methods have been proposed to reduce the training time of ensemble learning [2, 3, 12, 28, 31, 32]. These methods improve the efficiency of ensemble learning by sharing parameters among ensemble members [3, 28, 31], constructing ensembles from the training process of a single model [12], and speeding up training through the pre-training of a base model [20, 32]. However, these methods have lower test accuracy than standard ensembles, resulting in significant performance differences.

In typical neural network training, randomness is generally limited to weight initialization and mini-batch selection in Stochastic Gradient Descent (SGD). However, additional perturbations can also be introduced. A notable example includes the addition of noise to inputs, which can enhance model regularization and robustness on out-of-distribution data [1, 6, 26]. There is also extensive research on perturbing the weights of neural networks [1, 2, 23, 29, 34–36]. Weight perturbation can lead to effects such as regularization, exploration of parameter space, and robustness against adversarial perturbations. These approaches of introducing perturbations during neural network training are called Noise Injection. In this study, we specifically focus on perturbations to the weights.

We aim to reduce ensemble members’ training time while achieving performance comparable to that of standard ensembles. In this study, we propose the Noisy Deep Ensemble, which utilizes noise injection to achieve this goal. In this method, a single model (*parent model*) is trained until it converges, then perturbs its weights to construct multiple ensemble members (*child models*). Since each ensemble member starts with reasonably good weights, they converge to nearby local minima after a short training period. Each ensemble member converges to different local minima compared to the original model, and their predictions exhibit diversity. Therefore, an ensemble of these models can achieve performance comparable to those trained independently at inference time.

Figure 1 shows the differences in the learning process between the proposed method and the existing method (Snapshot Ensemble [12]). The Snapshot

Ensemble converges a single model to multiple local solutions by resetting to a high learning rate after convergence. On the other hand, the proposed method adds perturbations to the initial convergence point and explores a wider range of the parameter space, not limited to optimization by SGD.



**Fig. 1.** Difference in the learning process between Noisy Deep Ensemble and the existing method (Snapshot Ensemble [12]). Noisy Deep Ensemble promotes the exploration of wider parameter space by not being limited to the optimization path of SGD through Noise Injection. Snapshot Ensemble consists of  $\mathcal{M} = \{\theta_2, \theta_3\}$ , while Noisy Deep Ensemble consists of  $\mathcal{M} = \{\theta'_2, \theta'_3\}$ .

The contributions of this study are outlined as follows:

1. We developed a novel ensemble method called Noisy Deep Ensemble, which significantly reduces the training time of ensembles by perturbing the weights of neural networks.
2. We evaluated the proposed method with various CNN architectures on CIFAR-10 and CIFAR-100 datasets. Noisy Deep Ensemble achieved higher test accuracies than the traditional efficient ensemble method, Snapshot Ensemble, across various CNN architectures.
3. We conducted comprehensive experiments to investigate the diversity of predictions of each ensemble member and the impact of hyperparameters in the proposed method.

## 2 Background

### 2.1 Ensemble Learning

Here, we describe ensemble learning for neural networks in a standard class classification setting. Consider a given training set  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ , where  $(\mathbf{x}_i, y_i)$  represents the pair of the  $i$ -th data point and its corresponding label.  $N$  denotes the size of the training data, and  $y$  belongs to  $\{1, 2, \dots, C\}$ , with  $C$

being the number of classes. Let  $f_{\theta}$  represent a neural network with parameters  $\theta$ . The optimization problem for a single neural network is formulated as follows:

$$\theta^* = \operatorname{argmin}_{\theta} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\mathcal{L}(f_{\theta}(x), y)] \quad (1)$$

Here,  $\mathcal{L}$  is the loss function given the neural network's predicted probability distribution  $f_{\theta}(x) \in \mathbb{R}^C$  and the correct label  $y$ , generally employing the cross-entropy loss function. In many cases, optimization, as depicted in Eq. (1), utilizes SGD.

In ensemble learning,  $M$  neural networks which have different initial weights,  $f_{\theta^{(1)}}, f_{\theta^{(2)}}, \dots, f_{\theta^{(M)}}$ , are independently optimized based on Eq. (1). At inference time, the ensemble prediction for a given test sample  $x$  is determined by the average of the predictions from each neural network, denoted as  $P_{\text{ens}}$ :

$$P_{\text{ens}} = \frac{1}{M} \sum_{j=1}^M f_{\theta^{(j)}}(x) \quad (2)$$

When we mention “standard ensemble” in this paper, we refer to this particular form of ensemble learning.

## 2.2 Efficient Ensemble Training

A key challenge in ensemble learning for neural networks is reducing training time. Standard ensemble learning increases training time in proportion to the number of ensemble members. Several methods have been proposed to reduce the training time of such ensemble learning [2, 12, 32]. The Snapshot Ensemble [12] saves checkpoints from different learning stages of a single model and uses these models at each checkpoint for ensembling during inference. This allows ensemble learning to be accomplished within the training time of a single model. With regular learning rate scheduling, the predictions from these models at different checkpoints tend to be similar, limiting the effectiveness of ensemble learning. To address this issue, Snapshot Ensemble uses Cyclic Learning Rate Scheduling [21] to encourage the model to converge different local minima, thus ensuring prediction diversity. However, its accuracy is lower compared to standard ensemble learning.

MotherNet [32] involves pre-clustering multiple candidate model architectures and selecting a model called MotherNet within each cluster. MotherNet is trained using the entire dataset until convergence. After training, the MotherNet is expanded in depth and width using the technique proposed in Net2Net [4], creating multiple ensemble members known as ChildNets. The training time required for each ChildNet is significantly reduced compared to that of MotherNet. In MotherNet, noise from a Gaussian distribution is added to the weights of the trained MotherNet before constructing the ChildNets.

Group Ensemble [3] reduces the training time of each ensemble member by sharing the parameters of the lower layers among all ensemble members.

Ensemble learning can be performed similarly to single-model training through Grouped Convolution [16]. In addition, in ensemble learning with Knowledge Distillation [11, 20], the base model is trained until convergence, and then the ensemble members, randomly initialized, are trained to mimic the output of the base model. Because the teacher signals from the base model provide richer label information, each ensemble member converges faster than regular training.

Methods such as Dropout [28] and DropConnect [31] implicitly perform ensemble learning with multiple subnetworks. These techniques provide regularization effects and achieve higher test accuracy than a single model. Implicit ensemble learning requires approximately the same training time as a single model, but its test accuracy is lower than standard ensemble learning. The Pseudo Ensemble [2] is a framework that generates multiple *child models* by perturbing a *parent model*. DropConnect [31] and Dropout [28] are special cases of [2].

Our proposed Noisy Deep Ensemble, similar to the Snapshot Ensemble [12], attempts to escape from local minima by resetting the learning rate. Unlike the Snapshot Ensemble, the Noisy Deep Ensemble additionally encourages the exploration of the parameter space through perturbations to the weights, enhancing the diversity of predictions in ensemble learning. Additionally, unlike MotherNet [32], the Noisy Deep Ensemble maintains the same model architecture across all ensemble members, making it simpler and easier to implement. And also, MotherNet makes little mention of weight perturbation. The Noisy Deep Ensemble can be considered a form of pseudo-ensemble [2] learning. However, it is not an implicit form of ensemble learning that uses a single model but an explicit form that utilizes multiple models for inference, achieving higher test accuracy.

### 2.3 Noise Injection

Noise Injection is a technique for introducing some form of perturbation during neural network training to enhance generalization performance and robustness against perturbations. Various forms of perturbations can be considered, including perturbations to the inputs, the model’s outputs, or the weights. In this paper, we focus on perturbations to the weights.

In NoisyTune [35], noise from a uniform distribution is added once to the weights of a pre-trained language model before fine-tuning for downstream tasks. Adaptively varying the strength of the noise according to the standard deviation of the weights improves performance in downstream tasks. This approach perturbs the weights to mitigate overfitting of the language model to the pre-training tasks. In WeightAugmentation [36], random rigid transformations are applied to the weight matrices of neural networks during training. This acts as a form of regularization and has been shown to improve test accuracy across various CNN architectures. DropConnect [31] and other forms of pseudo-ensemble learning [2] also involve perturbing weights, contributing to its regularization effects.

An [1] demonstrates, under certain assumptions, the theoretical impact of perturbations to model weights on learning. When weight noise is sampled from a

Gaussian or uniform distribution and is of a sufficiently small scale, this additive noise can make neural networks more sparse and suppress overfitting. On the other hand, these results are based on an assumption continuously perturbing the weights during training.

Bayesian neural networks learn the posterior distribution of weights over a given training set rather than providing point estimates of optimal weights. Variational Bayesian neural networks approximate this posterior distribution of weights using variational inference to minimize the Evidence Lower BOund (ELBO) [7]. Here, perturbations of weights can also be considered as sampling from some posterior distribution of weights. While Bayesian neural networks tend to explore a single mode in the function space, ensemble learning explores multiple modes [5]. Therefore, generally, ensemble learning exhibits superior generalization performance and uncertainty estimation compared to Bayesian neural networks.

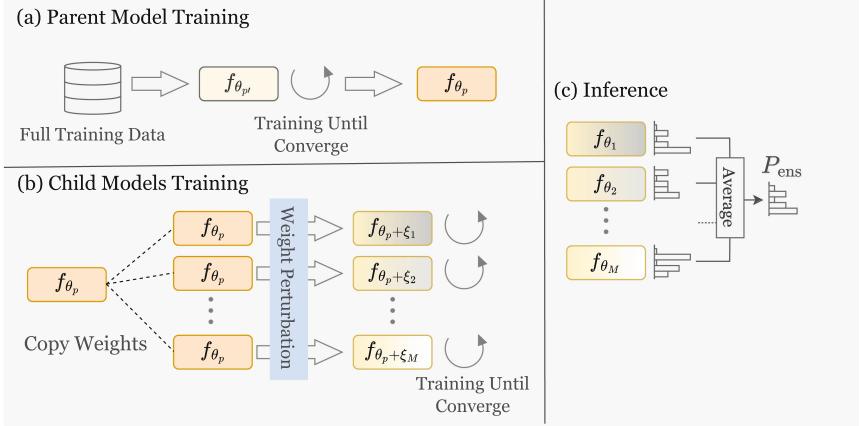
The model weights are perturbed only once after *parent model* training in the Noisy Deep Ensemble to explore parameter space effectively. Then, multiple *child models* are instantiated with different perturbed weights. This differentiates it from approaches [1, 2, 36] that continuously perturb the weights during training. The purpose of perturbation in the Noisy Deep Ensemble is similar to that of NoisyTune [35], aiming to explore the parameter space and encourage convergence to different local minima. Additionally, the Noisy Deep Ensemble incorporates the advantages of both Bayesian neural networks and ensembles, exploring diverse modes within the function space while accounting for the uncertainty of the weights.

### 3 Noisy Deep Ensemble

Figure 2 provides an overview of the proposed method: Noisy Deep Ensemble. The training process of the Noisy Deep Ensemble consists of two stages: training the *parent model* and training the *child models*. First, the *parent model* is trained until it converges on all available training data (Fig. 2(a)). Afterward, the weights of the trained *parent model* are duplicated to construct each ensemble member (*child models*). The *child models* have the same network architecture as the *parent model*, and their initial weights are identical to those of the trained *parent model*. Therefore, simply retraining the *child models* with standard SGD does not produce sufficient diversity in the ensemble predictions. To address this, we perturb the weights of each *child model* to encourage exploration of different local minima. After perturbing the weights of the *child models*, they are independently trained until convergence (Fig. 2(b)). During inference, the ensemble’s prediction is obtained by averaging the predicted probability distributions of all *child models* (Fig. 2(c)).

We present the Noisy Deep Ensemble details, considering a standard classification setting. Let us assume the training data set is given as follows:

$$\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad (3)$$



**Fig. 2.** Overview of the Noisy Deep Ensemble

where  $(\mathbf{x}_i, y_i)$  represents the data point and corresponding class label of the  $i$ -th sample, and  $i \in \{1, 2, \dots, N\}$ , with  $C$  being the number of classes. We define a neural network  $f_{\theta^{(p)'}}$  with initial weight  $\theta^{(p)'}$  as the *parent model*, and train it using SGD according to Eq. (1) until convergence, where  $\mathcal{D} = \mathcal{D}_{\text{train}}$ . The weight of the trained *parent model* is denoted as  $\theta^{(p)} \in \mathbb{R}^D$  where  $D$  is the number of dimensions of weights.

We independently sample  $M$  times from a noise distribution  $p(\xi)$  to obtain  $M$  noise vectors  $\{\xi^{(1)}, \xi^{(2)}, \dots, \xi^{(M)}\}$ . For any  $i$ ,  $\xi^{(i)} \in \mathbb{R}^D$  can be added to the weights of the trained *parent model*. Thus, we initialize the weights of multiple ensemble members (*child models*) by applying different perturbations to the pre-trained *parent model* as shown in Eq. (4):

$$\theta^{(i)'} = \theta^{(p)} + \xi^{(i)}, \quad i = 1, 2, \dots, M \quad (4)$$

Although Eq. (4) perturbs all weights, in this study, we selectively perturb weights by using a random mask vector  $\mathbf{m}^{(i)} \in \{0, 1\}^D$  as follows:

$$\theta^{(i)'} = \theta^{(p)} + \xi^{(i)} \odot \mathbf{m}^{(i)}, \quad i = 1, 2, \dots, M \quad (5)$$

where  $\odot$  denotes the hadamard product. Equation (4) is a special case in Eq. 5 where the weight perturbation by  $\mathbf{m}^{(i)} \in [1, 1, \dots, 1]^\top$ .

Subsequently, each *child model* is trained using SGD on  $\mathcal{D}_{\text{train}}$  until convergence, following Eq. (1). As each *child model* is initialized using the weights of the pre-trained *parent model*, they converge more quickly. Increasing the perturbation scale leads to a more significant deviation from the pre-trained weights of the *parent model*, resulting in a longer training time for the *child models*. Conversely, decreasing the perturbation scale tends to result in convergence to the same local minima as the pre-trained *parent model*, reducing the diversity of predictions in ensemble learning. Thus, the perturbation scale balances a trade-off between the diversity of predictions in ensemble learning and training time.

During inference, the average prediction probability distribution of the trained *child models*  $f_{\theta^{(1)}}, f_{\theta^{(2)}}, \dots, f_{\theta^{(M)}}$  is computed as Eq.(2), and this average is used as the ensemble's overall prediction. Each *child model* converges to different local minima through their weights perturbations and short-time training, providing diversity in their predictions. As a result, the ensemble can achieve superior test accuracy compared to individual *child models* or the *parent model* alone.

The key hyperparameters in a Noisy Deep Ensemble are the proportion of weights subject to perturbation and the scale of perturbation. The proportion of weights to be perturbed, denoted as  $\alpha$ , is related to the mask vector  $\mathbf{m}$ , and is given by  $\alpha = \frac{1}{D} \sum_{i=1}^D m_i$ . More precisely, each element of  $\mathbf{m}$  is sampled from Bernoulli distribution, i.e.,  $m_i \sim \text{Bernoulli}(p)$ , and  $p$  is equivalent to  $\alpha$ . In this study, we consider using a Gaussian distribution  $\xi_i \sim \mathcal{N}(0, \beta)$  or a uniform distribution  $\xi_i \sim U(-\beta, \beta)$  as the perturbation distribution, where  $\beta$  represents the scale of the perturbation.

## 4 Experiments

### 4.1 Experiment Setting

In this study, we validate the effectiveness of our proposed method using CIFAR-10 (C10) and CIFAR-100 (C100) [15]. As evaluation metrics, we employ test accuracy on CIFAR-10/100. To demonstrate the effectiveness of our method across various CNN architectures, we utilize popular CNN architectures such as ResNet18 [9], VGG16 [27], and EfficientNetB0 [30].

During training, the mini-batch size is set to 64, and Momentum SGD [24] is used as the optimizer, with a momentum value of 0.9 and weight decay set to 0.0005. Learning rate scheduling employs Cosine Scheduling [21], with maximum and minimum learning rates set to 0.1 and 0.0, respectively. Throughout all experiments, the *parent model* is trained for 200 epochs, while *child models* are trained for 50 epochs. Similarly, single models and conventional ensemble learning are trained for 200 epochs unless specified. The number of ensemble members is set to 10 unless specified. When perturbing the weights, two hyperparameters are considered: the proportion of weights perturbed,  $\alpha$ , and the scale of perturbation,  $\beta$ . This study determined their optimal values through grid search, presented in Table 1. Unless specified, these values are used. The impact of these hyperparameters on test accuracy is evaluated in Sect. 5.1.

### 4.2 Main Results

To verify the effectiveness of the proposed method, we compared the test accuracies on CIFAR-10 and CIFAR-100 under the following settings:

- **Single:** Single model.
- **Ensemble:** Standard ensemble learning, training  $M$  models independently from different initial weights.

**Table 1.** Optimal weight perturbation values found by grid search.  $\alpha$  denotes the ratio of weights which are perturbed.  $\beta$  denotes the strength of the noise.

Noise Parameter	ResNet18			VGG16			EfficientNetB0			
	C10		C100	C10		C100	C10		C100	
	$\alpha$	$\beta$	$\alpha$	$\beta$	$\alpha$	$\beta$	$\alpha$	$\beta$	$\alpha$	
$\xi_i \sim U(-\beta, \beta)$	0.8	1.6	0.8	1.6	0.1	0.1	0.05	0.1	0.7	0.1
$\xi_i \sim \mathcal{N}(0, \beta)$	0.8	1.6	0.8	1.6	0.1	0.01	0.05	0.05	0.7	0.1
									0.3	0.3

- **Noisy Single:** Single model in which weights were perturbed after first convergence and trained again until convergence. The perturbation distribution  $p(\xi)$  includes both uniform (uni) and Gaussian (norm) distributions.
- **Noisy Ensemble:** Trained single model until convergence, then weights were perturbed, and multiple models were independently trained. Each model’s weight noises are sampled independently from uniform (uni) or Gaussian (norm) distributions.

**Table 2.** Comparison of model performances on CIFAR10(C10) and CIFAR100 (C100) datasets. The highest accuracy is shown in **bold**, while the second highest is underlined.

Method	ResNet18		VGG16		EfficientNetB0	
	C10	C100	C10	C100	C10	C100
Single	0.8965	0.7350	0.8823	0.6023	<u>0.9186</u>	0.7197
Ensemble	<b>0.9158</b>	<b>0.7739</b>	<u>0.9036</u>	0.6532	<b>0.939</b>	<b>0.7782</b>
Noisy Single (uni)	0.8863	0.7075	0.8936	0.6163	0.903	0.6929
Noisy Single (norm)	0.8871	0.7640	0.8893	0.6112	0.8999	0.6852
Noisy Ensemble (uni)	<u>0.9147</u>	0.7151	<b>0.9072</b>	<b>0.662</b>	0.9115	<u>0.7284</u>
Noisy Ensemble (norm)	0.9132	<u>0.7660</u>	0.9008	<u>0.6577</u>	0.9129	0.7178

Table 3 presents the test accuracy in different ensemble configurations for CIFAR-10 and CIFAR-100. Standard ensemble learning (Ensemble) significantly improves test accuracy in all cases compared to a single model (Single). On the other hand, the Noisy Single approach, which involves training a single model with perturbed weights, does not consistently show superiority over the single model, and in some cases, test accuracy decreases depending on the model and dataset. This is true even when the type of perturbation distribution is varied. However, our Noisy Ensemble approach, which involves retraining multiple models with perturbed weights, surpasses the single model test accuracy in almost all configurations and demonstrates performance comparable to that of standard ensemble learning.

Table 3 compares the performance with existing ensemble learning methods. The CIFAR-10 and CIFAR-100 datasets and the ResNet18 model architecture are used. The number of ensemble members is set to 10. The proposed method significantly outperforms all existing ensemble learning methods [3, 11, 12] and narrows the performance gap with standard ensembles. Knowledge Distillation Ensemble [11], similar to Noisy Deep Ensemble, pre-trains a *parent model*, but the weights of the *child models* are randomly initialized. Therefore, it is possible that the models may not fully converge in a short training time.

**Table 3.** Comparison with other ensemble methods. KDE stands for Knowledge Distillation Ensemble [11], GE stands for Group Ensemble [3], SE stands for Snapshot Ensemble [12], and BE stands for BatchEnsemble [33].

Method	KDE [11]	GE [3]	SE [12]	BE [33]	Ours	Standard
C10	0.8725	0.9043	0.9088	0.8259	<u>0.9132</u>	<b>0.9158</b>
C100	0.6771	0.6524	0.7041	0.5869	<u>0.7660</u>	<b>0.7739</b>

**Table 4.** Test accuracy of each ensemble member.

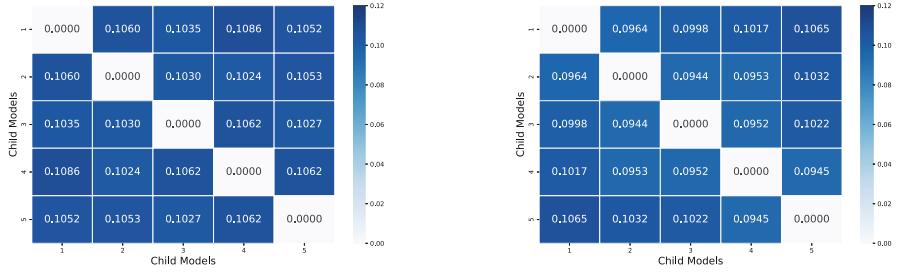
Accuracy	Child1	Child2	Child3	Child4	Child5	Ensemble
$\xi_i \sim U(-\beta, \beta)$	0.8836	0.8849	0.8911	0.8843	0.8876	<b>0.9116</b>
$\xi_i \sim \mathcal{N}(0, \beta)$	0.8897	0.8884	0.8894	0.8861	0.8829	0.9102

### 4.3 Evaluating Ensemble Effectiveness

Ensemble learning is more effective when these two conditions are met: (i) each ensemble member exhibits high test accuracy, and (ii) ensemble members don't share miss-classified samples with each other [12]. This section discusses about these conditions, (i) and (ii), in detail. We use the CIFAR-10 dataset and the ResNet18 model architecture in the subsequent experiments. The number of ensemble members is set to 5. The weights perturbation scale is the same as described in Sect. 4.1.

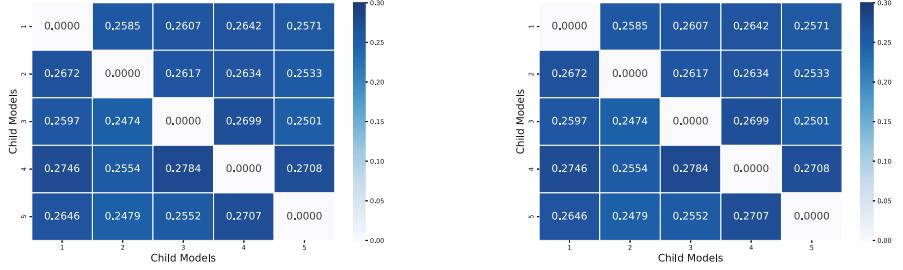
Table 4 shows the test accuracy of individual ensemble members in the Noisy Deep Ensemble. The test accuracy is sufficiently high for both uniform and Gaussian perturbation distributions, although they are slightly lower than those of the single models. For reference, the accuracy of a standalone ResNet18 on CIFAR-10 is 0.8965, as shown in Table 2. Furthermore, the performance is improved by their ensemble.

In ensemble learning, more significant benefits are obtained when the predictions of each ensemble member are diverse. One straightforward metric for



(a) Noisy Deep Ensemble (uni)

(b) Snapshot Ensemble

**Fig. 3.** The disagreement rate among ensemble members' predictions

(a) Noisy Deep Ensemble (uni)

(b) Snapshot Ensemble

**Fig. 4.** Average KL divergence between the prediction probability distributions of ensemble members on the test data

evaluating such prediction diversity is the disagreement rate. Figure 3 shows the disagreement rates in predictions on the CIFAR-10 test data for both the Noisy Deep Ensemble and the Snapshot Ensemble. A uniform distribution (uni) is used as the perturbation distribution, and the number of ensemble members is set to 5. The disagreement rate is calculated for every pair of ensemble members. Compared to the Snapshot Ensemble, the Noisy Deep Ensemble exhibits a slightly higher rate of prediction disagreement, i.e., the Noisy Deep Ensemble has higher prediction diversity than the Snapshot Ensemble.

The KL divergence between the prediction probability distributions of each ensemble member also represents the diversity of predictions well. Figure 4 shows the average KL divergence of the prediction probability distributions on the CIFAR-10 test data for both the Noisy Deep Ensemble and the Snapshot Ensemble. Clearly, the Noisy Deep Ensemble exhibits a higher average KL divergence among ensemble members than the Snapshot Ensemble. The Noisy Deep Ensemble enhances the diversity of predictions and improves the test accuracy of the ensemble through perturbations to the weights and independent training of each ensemble member.

#### 4.4 Training Efficiency of Noisy Deep Ensemble

In this section, we examine the training time of the Noisy Deep Ensemble in comparison to standard ensemble learning. In standard ensemble learning, if the training time for a single model is  $T_{\text{single}}$  and the number of ensemble members is  $M$ , the total training time is  $T_{\text{standard}} = M \cdot T_{\text{single}}$ . On the other hand, the total training time for the Noisy Deep Ensemble is  $T_{\text{parent}} + T_{\text{child}} \cdot M$ . Assuming that the training time for the *child model* can be significantly reduced compared to the *parent model*, with  $T_{\text{parent}} \gg T_{\text{child}}$ , the Noisy Deep Ensemble shows a linear reduction in training time relative to the number of ensemble members compared to standard ensembles. Therefore, the Noisy Deep Ensemble is more suitable for scaling the number of ensemble members than traditional ensemble learning. The impact of scaling the number of ensemble members on accuracy is examined in Sect. 5.2. We compared the training times of standard ensemble learning and Noisy Deep Ensemble with different numbers of ensemble members using wall-clock time<sup>1</sup>. The results are shown in Table 5. Compared to the training time of a standard ensemble, the training time reduction of Noisy Deep Ensemble ( $T_{\text{noisy}}$ ) decreases as the number of ensemble members increases. Notably, when the number of ensemble members is 10, the training time for the Noisy Deep Ensemble is 35% of that for the standard ensemble. Additionally, compared to a standard ensemble, the degradation in test accuracy is minimized<sup>2</sup>.

**Table 5.** Comparison of training time between standard and noisy deep ensemble models in wall-clock hours

# Ensemble members	2	4	6	8	10
$T_{\text{standard}}$ [h]	2.1	4.2	6.3	8.3	10.4
$T_{\text{noisy}}$ [h]	1.6	2.1	2.6	3.1	3.6
$T_{\text{noisy}}/T_{\text{standard}}$ [%]	73%	50%	41%	37%	35%
Accuracy Drop $[\Delta]$	0.0058	0.0029	0.0013	0.0019	0.0025

#### 4.5 Performance of Robustness and Calibration

We evaluate the robustness of data corruption on CIFAR-10-C [10]. The results are shown in Table 6. Noisy Deep Ensemble outperforms the single model across all corruption severity and performs similarly to the standard ensemble. This indicates the model’s robustness can be improved by the noisy, deep ensemble.

Table 7 shows the calibration performance on CIFAR-10 and CIFAR-100. Noisy Deep Ensemble is better overall than the single model regarding calibration

<sup>1</sup> The experiments were conducted using an NVIDIA GeForce RTX3090 (24 GB) and an Intel Core i9-10850K @ 3.60 GHz (32 GB).

<sup>2</sup> We used uniform noise for weight perturbation as Sect. 4.3.

**Table 6.** Comparison of model performance on CIFAR-10 across different severity levels of degradation. The values in the table represent test accuracy.

Method	Severity					
	0	1	2	3	4	5
Standard	<b>0.9158</b>	<b>0.8406</b>	<b>0.7846</b>	<b>0.7313</b>	<b>0.6648</b>	<b>0.5605</b>
Single	0.8965	0.8135	0.7549	0.7013	0.6379	0.5351
NDE (uni)	<b>0.9147</b>	0.8262	0.7712	0.7172	0.6483	0.5456
NDE (norm)	0.9132	<b>0.8281</b>	<b>0.7743</b>	0.7218	0.6531	0.5509

performance. However, the difference in calibration performance between the standard ensemble and the noisy deep ensemble is significant compared to the difference in accuracy. The improvement of Noisy Deep Ensemble performance in calibration remains as future work.

**Table 7.** Comparison of model performance on CIFAR-10 and CIFAR-100 datasets. Acc. represents test accuracy ( $\uparrow$ ), ECE is expected calibration error ( $\downarrow$ ), and NLL is negative log-likelihood ( $\downarrow$ ).

Method	CIFAR-10			CIFAR-100		
	Acc. ( $\uparrow$ )	ECE ( $\downarrow$ )	NLL ( $\downarrow$ )	Acc. ( $\uparrow$ )	ECE ( $\downarrow$ )	NLL ( $\downarrow$ )
Standard	<b>0.9158</b>	<b>0.0205</b>	<b>0.2961</b>	<b>0.7739</b>	<b>0.0575</b>	<b>0.9787</b>
Single	0.8965	0.0383	0.3791	0.7350	<b>0.0525</b>	1.1667
NDE (uni)	<b>0.9147</b>	<b>0.0329</b>	<b>0.3408</b>	0.7151	0.0762	0.9975
NDE (norm)	0.9132	0.0343	0.3453	<b>0.7660</b>	0.0740	<b>0.9856</b>

## 5 Ablation

### 5.1 Effect of Different Weight Perturbation Strategies

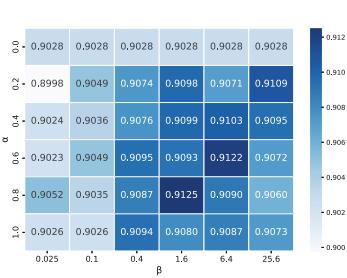
Here, we investigate the effects of varying the perturbation strategies applied to the weights. Specifically, we focus on two aspects: (i) the proportion of weights perturbed, denoted as  $\alpha$ , and (ii) the scale of noise applied to the weights, denoted as  $\beta$ . Details are discussed in Sect. 3. Figure 5 shows the test accuracy on CIFAR-10 when training the Noisy Deep Ensemble with different values of  $\alpha$  and  $\beta$ . A uniform distribution is considered for the perturbation distribution.

As shown in Fig. 5, the proportion of weights affected by perturbations has a minor impact on test accuracy. In contrast, the scale of the noise significantly influences test accuracy. Specifically, test accuracy declines when the noise scale is reduced to  $\beta \leq 0.1$ . This decrease occurs because a smaller noise scale tends to converge towards the same local minima as the original *parent model*, resulting in

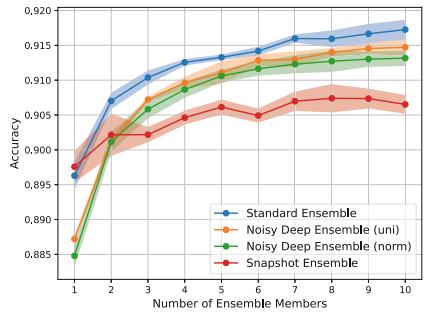
a loss of prediction diversity in ensemble learning. On the other hand, even when the noise scale is increased to  $\beta \geq 5.0$ , the decrease in test accuracy is minimal. This is due to the retraining of each model after perturbing the weights. When  $\alpha$  is set to 0, i.e., no weight perturbation, the test accuracy is lower than in most perturbed settings. This suggests that our proposed weight perturbation is essential.

## 5.2 Scaling Number of Ensemble Members

This section examines the impact of increasing the number of members in an ensemble. In standard ensemble learning, performance is typically improved as the number of ensemble members increases [19]. We investigate whether a similar trend is observed in the Noisy Deep Ensemble. The experimental results are shown in Fig. 6. Here, the Noisy Deep Ensemble was trained with varying numbers of ensemble members, and the mean and standard deviation of 5 trials are plotted. As shown in Fig. 6, the Noisy Deep Ensemble demonstrates improved test accuracy with increased ensemble members. Conversely, the Snapshot Ensemble [12] shows a plateau improvement in test accuracy despite increased ensemble members. This plateau is attributed to the lower diversity of predictions among ensemble members derived from a single model's learning process compared to those trained independently (Sect. 4.3). The Noisy Deep Ensemble exhibits scaling capabilities comparable to typical ensembles with uniform (uni) and Gaussian (norm) perturbation distributions.



**Fig. 5.** The effects of changes in perturbations on the accuracy



**Fig. 6.** The effects of changes in the number of ensemble members on the accuracy

## 5.3 Performance with Various Optimisers

We validate noisy deep ensemble is compatible with various optimizers such as RMSProp [8], AdamW [22]. The results are shown in Table 8. Because we use the same hyperparameters tuned for SGD, the performances of RMSProp and AdamW are slightly worse. However, we can see similar performance improvement as SGD.

**Table 8.** Comparison of optimization algorithms for different methods. The value in the table represents test accuracy on CIFAR-10.

Method	SGD	RMSProp	AdamW
Standard	<b>0.9158</b>	<b>0.8637</b>	<u>0.8742</u>
Single	0.8965	0.8017	0.8194
NDE (uni)	<b>0.9147</b>	<b>0.8392</b>	<b>0.8744</b>
NDE (norm)	0.9132	0.8351	0.8554

## 6 Conclusion

In this study, we proposed the Noisy Deep Ensemble, a method designed to make the training of neural network ensembles more efficient. This method demonstrated superior test accuracy compared to conventional ensemble methods across various CNN architectures on CIFAR-10 and CIFAR-100.

In some cases, Noisy Deep Ensemble performs worse than a single model (see Table 2). The uncertain nature of our perturbation process causes this. For example, the locations for weight perturbation were chosen randomly. From the perspective of neural network pruning, it has been shown that ignoring most weights selected appropriately has minimal impact on performance [18]. Insights gained in this area could be utilized to refine the selection of weights to perturb in the Noisy Deep Ensemble, such as preferentially perturbing weights with larger absolute values.

**Acknowledgments.** This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant-in-Aid for Scientific Research (C) under Grant 23K11164. This work was also supported by several competitive funds within the University of Fukui.

## References

1. An, G.: The effects of adding noise during backpropagation training on a generalization performance. *Neural Comput.* **8**, 643–674 (1996)
2. Bachman, P., Alsharif, O., Precup, D.: Learning with pseudo-ensembles. *Adv. Neural Inf. Process. Syst.* **27** (2014)
3. Chen, H., Shrivastava, A.: Group ensemble: learning an ensemble of convnets in a single convnet. CoRR [arxiv:2007.00649](https://arxiv.org/abs/2007.00649) (2020)
4. Chen, T., Goodfellow, I.J., Shlens, J.: Net2net: accelerating learning via knowledge transfer. CoRR [arxiv:1511.05641](https://arxiv.org/abs/1511.05641) (2015)
5. Fort, S., Hu, H., Lakshminarayanan, B.: Deep ensembles: a loss landscape perspective. arXiv preprint [arXiv:1912.02757](https://arxiv.org/abs/1912.02757) (2019)
6. Grandvalet, Y., Canu, S., Boucheron, S.: Noise injection: theoretical prospects. *Neural Comput.* **9**, 1093–1108 (1997)
7. Graves, A.: Practical variational inference for neural networks. In: *Neural Information Processing Systems* (2011)

8. Graves, A.: Generating sequences with recurrent neural networks. CoRR (2013)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2015)
10. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: International Conference on Learning Representations (2019)
11. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR [arxiv:1503.02531](#) (2015)
12. Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q.: Snapshot ensembles: train 1, get m for free. In: International Conference on Learning Representations (2017)
13. Humayun, A.I., Balestriero, R., Baraniuk, R.: Deep networks always grok and here is why. arXiv preprint [arXiv:2402.15555](#) (2024)
14. Kaplan, J., et al.: Scaling laws for neural language models. ArXiv [arxiv:2001.08361](#) (2020)
15. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2012)
17. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. In: Neural Information Processing Systems (2016)
18. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Neural Information Processing Systems (1989)
19. Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D.J., Batra, D.: Why m heads are better than one: training a diverse ensemble of deep networks. CoRR [arxiv:1511.06314](#) (2015)
20. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Trans. Pattern Anal. Mach. Intell. **40**, 2935–2947 (2016)
21. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: International Conference on Learning Representations (Poster) (2017)
22. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019)
23. Orvieto, A., Raj, A., Kersting, H., Bach, F.R.: Explicit regularization in overparametrized models via noise injection. In: International Conference on Artificial Intelligence and Statistics (2022)
24. Polyak, B.: Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Phys. **4**, 1–17 (1964)
25. Power, A., Burda, Y., Edwards, H., Babuschkin, I., Misra, V.: Grokking: generalization beyond overfitting on small algorithmic datasets. In: 1st Mathematical Reasoning in General Artificial Intelligence Workshop, ICLR 2021 (2021)
26. Seghouane, A.K., Moudden, Y., Fleury, G.A.: Regularizing the effect of input noise injection in feedforward neural networks training. Neural Comput. Appl. **13**, 248–254 (2004)
27. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR (2014)
28. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)

29. Sum, J., Ho, K.I.J.: Sniwd: simultaneous weight noise injection with weight decay for mlp training. In: International Conference on Neural Information Processing (2009)
30. Tan, M., Le, Q.V.: Efficientnet: rethinking model scaling for convolutional neural networks. In: Proceedings of the 36th International Conference on Machine Learning, pp. 6105–6114. ICML (2019)
31. Wan, L., Zeiler, M.D., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: International Conference on Machine Learning (2013)
32. Wasay, A., Hentschel, B., Liao, Y., Chen, S., Idreos, S.: Mothernets: rapid deep ensemble learning. In: Conference on Machine Learning and Systems (2018)
33. Wen, Y., Tran, D., Ba, J.: Batchensemble: an alternative approach to efficient ensemble and lifelong learning (2020)
34. Wen, Y., Vicol, P., Ba, J., Tran, D., Grosse, R.: Flipout: efficient pseudo-independent weight perturbations on mini-batches. In: International Conference on Learning Representations. ICLR (2018)
35. Wu, C., Wu, F., Qi, T., Huang, Y., Xie, X.: Noisytune: a little noise can help you finetune pretrained language models better. In: Annual Meeting of the Association for Computational Linguistics (2022)
36. Zhuang, J., Din, G., Yan, Y.: Weights augmentation: it has never ever ever let her model down (2024)



# LCNet: Lightning Hierarchical Convolution for Occupancy Flow Prediction

Yanjie Zhao<sup>(✉)</sup> and Zhongwen Xiao

Zhejiang Leapmotor Technology Co., Ltd., Kowloon, Hong Kong  
zhyj\_going@163.com

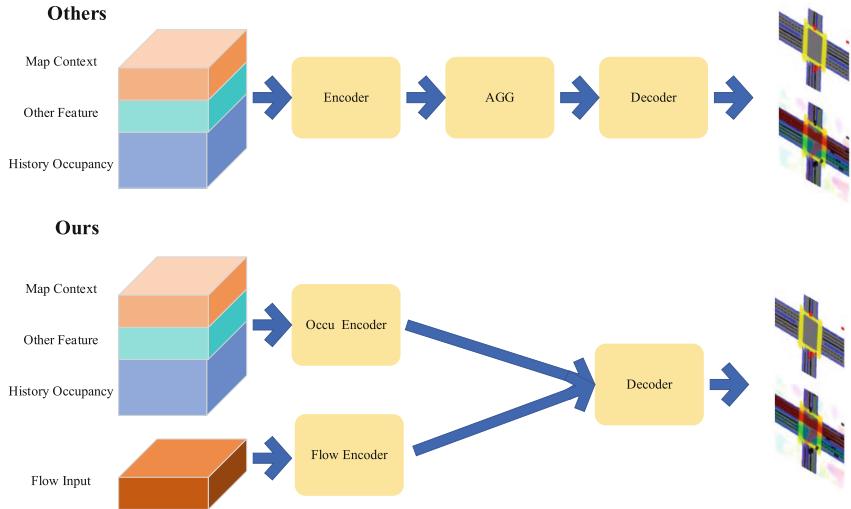
**Abstract.** Motion Prediction is a crucial part of Autonomous technique for autonomous driving. In order to predict entire scene agents, some occupancy flow prediction methods are developed. However, these methods exist the shortcomings of incomplete information representation or low computational efficiency. Therefore, this paper proposes a lightning hierarchical convolution network for occupancy flow prediction. The multiple inputs (map, flow and occupancy) are encoded by multi-stage hrnet module and swin-transformer module separately which can learn high-resolution information. And the hierarchical convolution decoder captures multi-scale features efficiently. The proposed method is comprehensively validated on the Waymo Open Motion Dataset. Our model achieves a Flow-Grounded Occupancy AUC of 0.8184. And compared with other comparable performance models, our lightning model inference time is as low as 0.1065 s.

**Keywords:** Occupancy Flow Prediction · Hierarchical Convolution · Lightning Model

## 1 Introduction

Currently, most autonomous technology stacks consist of six parts: lidar perception, vision perception, fusion perception, agent tracking, motion prediction, and path planning [1]. As a technology related to path planning, motion prediction plays a crucial role in autonomous driving technology, and it directly affects the effect of the agent's future predicted trajectory. Although more and more researchers are currently involved in autonomous driving research, motion prediction still needs to be vigorously promoted. Due to the complex environments of surrounding agents, it is difficult to accurately predict the future trajectory of the agent. Most proposed motion prediction algorithms take sequential data as input. And the input features are extracted through convolutional neural network (CNN), recurrent neural networks (RNN), etc. [2–4]. These methods have less complicated network architectures, but have the following disadvantages: 1) the global coordinate of agent need to be converted to local coordinate, it greatly increases the complexity of the input. 2) the agent to be predicted cannot fully interact with the surrounding environment. 3) the occluded agents which cannot be observed at current time cannot be predicted.

Waymo presented occupancy and flow prediction algorithms in 2022, which can effectively solve the input complexity generated by sequential data [5]. The input of occupancy enables the algorithm to predict all agents which within limits by ego. It can also predict the future occupancy of some unknown objects, which cannot be achieved by traditional sequential methods. Moreover, the flow can better reflect the future movement trend of the agent, including the agent's speed, quaternion, etc.



**Fig. 1.** Comparison of our method and most others method, our method creates a separate flow branch which interact flow feature and other feature abstractly. Caused of flow data and occupancy data has different feature distribution, same encoder cannot extract flow and occupancy feature together.

Up to now, most occupancy and flow prediction algorithms aim to improve the accuracy of the model but ignore the computational efficiency. The 1st and 2nd place in the 2022 Waymo Occupancy and Flow Prediction Challenge, HOPE [6] and StrajNet [7], have excellent manifestation, but they are computationally expensive. Although StrajNet has reduced the parameters of the network, limiting it to more dense layers, its parameters are still quite large. Therefore, we propose a lightning hierarchical convolution network, as shown in Fig. 1. Comparison of our method and most others method, our method creates a separate flow branch which interact flow feature and other feature abstractly. Caused of flow data and occupancy data has different feature distribution, same encoder cannot extract flow and occupancy feature together. First, to reduce the complexity of the input data, we represent a single map image as map data. Compared with most algorithms, this approach significantly reduces the complexity of the input data, but does not reduce the accuracy of the model. Due to the different distribution of the flue data and the simplicity of only 2 channels, we employ a single Swin-transformer layer [8] to encode the flue data individually. Meanwhile, we adopt HRNet [9] encoder to extract features of history occupancy and other data which can achieve multi-scale fusion and

feature extraction. To enlarge our receptive field without sacrificing the efficiency, we design stacked dilated 3D convolutional bottleneck structures decoder with soft attention (Soft-STD). In summary, our contributions are as follows:

- We design a novel lightning hierarchical convolutional framework for occupancy and flow prediction tasks. It can perform multi-scale fusion and feature extraction of occupancy, flow and other input data while maintaining high resolution.
- We design a novel Soft-STD decoder. It extracts major feature along time dimension for improving model’s accuracy efficiently.
- We validate the framework on large-scale real-world driving dataset, and the proposed model with a concise and lightning structure achieves state of the art.

## 2 Related Work

### 2.1 CNN for Motion Prediction

The CNN can not only efficiently extract features, but also calculate the temporal and spatial correlation of trajectories, and simplify the model structure. Therefore, CNN is widely used in motion prediction. Nikhil [10] utilized CNNs to predict the trajectories and achieve competitive results while reaching computational efficiency. Social-IWSTCNN [11] further proposed a novel design, namely the Social Interaction Extractor, to learn the spatial and social interaction features of pedestrians. To achieve real-time prediction of occupancy and flow prediction, we utilize CNN to extract features of the occupancy raster. Due to the different characteristics of occupancy, the occupancy along the time dimension on the raster map is different. Therefore, we choose the multi-scale feature fusion network HRNet to extract features occupying the input. Moreover, based on the HOPE [6], we propose the Soft-STD decoder which extract the most influential features for future motion from multi-scale feature.

### 2.2 Occupancy Flow Field for Motion Prediction

Forecasting the future motion through occupancy grids can date back to Chauffeur-Net [12], which predicts the future occupancy map to perform behavior planning in autonomous driving. HOPE [6] proposed a hierarchical spatial-temporal predictor comprising a deep-level multi-stage encoder-decoder, as well as 3 layers of aggregators [13] for fusing high level visual features. However, HOPE utilizes a large amount of data as input, including some that has little impact on the task. Meanwhile, it also has the deepest network architecture and high accuracy, but requires more time and GPU memory. Compared to HOPE, VectorFlow [14] and StrajNet [7] has a less-parameter network. VectorFlow is a simple but effective approach that fuses both vectorized and rasterized representations of traffic context through attention to predict occupancy flow fields for both observed agents and occluded agents. And StrajNet proposed a novel Multi-modal Hierarchical Transformer network that fuses the vectorized (agent motion) and visual (scene flow, map, and occupancy) modalities and jointly predicts the flow and occupancy of the scene. However, due to its vector inputs and massive multi-layer perception, it still requires a lot of time and GPU memory to implement the predictions. Unlike the above

methods, we select the most influential input data possible, fuse the map messages into a single map image and lightning our network. In addition, a single branch for flow input can also help extract flow feature accurately.

### 3 Methods

#### 3.1 Input and Output Representation

For the Occu Encoder, the input is the occupancy raster map  $OM$  which includes the occupancy information  $O$  and road map features  $M$ . The  $O$  consists of history ten frames and current frame occupation information, it is concatenated along the temporal dimension. We merge the occupancy of pedestrian and bicycle, thus the channel of occupancy is 22. The  $M$  is a single RGB format image data which consists of some static information (e.g. overall road map, stop sign, road edge, crosswalk, traffic light). Thus, the channel of  $OM$  is  $22 + 3 = 25$ . To be specific, the shape of  $OM$  is  $H * W * 25$ . This input data method can reduce the memory share and characteristic complexity of the data, thereby increasing the speed of algorithm training.

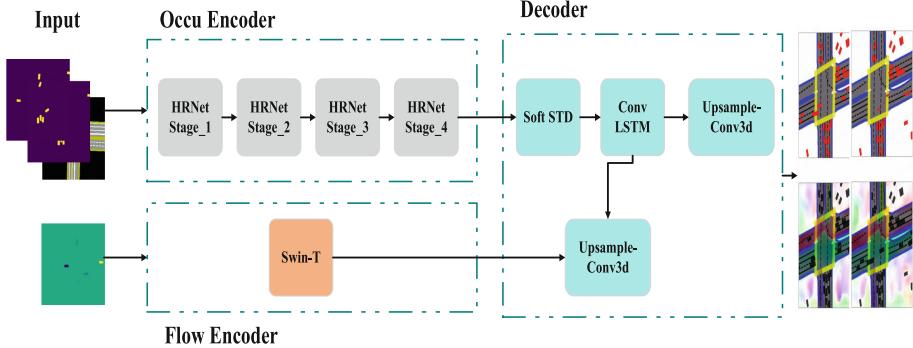
For the Flow Encoder, the input is the past flow data  $F$ . Because the occupancy data and flow data have large data distribution differences, the past flow data is encoded separately. One of the objectives of our model is to predict future occupancies and flow (motion) of only vehicle over 8 s into the future. Therefore, we only encode the flow of the vehicle which is a two-dimensional vector denoting the flow along  $d_x$  and  $d_y$  directions. To be specific, the shape of  $F$  is  $H * W * 2$ .

For the model output, our model predicts future occupancy of all vehicles that are present or not present at the current timestep  $t$ , for 8 s into the future. More specifically, the predictions are 8 occupancy grids capturing future occupancy of all currently-visible  $\hat{O}_t^b$  and currently-occluded  $\hat{O}_t^c$ ,  $t \in (0, \dots, 7)$  vehicles at 8 different waypoints separately. Each occupancy grid  $\hat{O}_t^b$  and  $\hat{O}_t^c$  are  $H * W * 1$  array containing values in range  $[0, 1]$  indicating the probability that some part of some currently-observed or currently-occluded vehicle will occupy that grid cell. And our model also predicts future flow of all vehicles (currently observed or occluded), for 8 s into the future. More specifically, the predictions are 8 flow fields  $\hat{F}_t$ ,  $t \in (0, \dots, 7)$ , capturing future flow of all vehicles at 8 different waypoints.

#### 3.2 Encoder

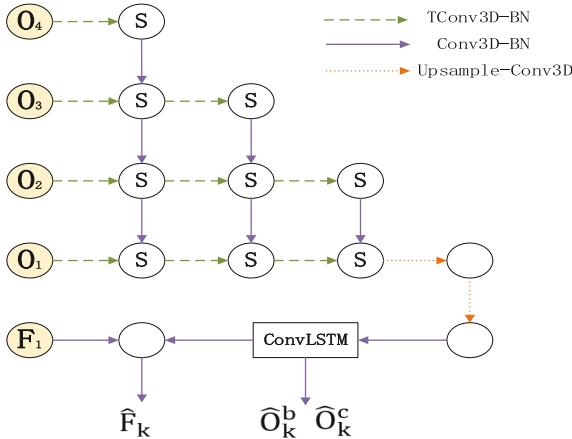
As shown in Fig. 2. An overview of the LCNet model., the encoder part of our proposed LCNet consists of an occupancy encoder and a flow encoder. Recently, HRNet-based models have achieved great success on detection and segmentation [15–17]. The HRNet encoder is mainly implemented through hierarchical convolution. And it starts from a high-resolution subnetwork as the first stage, gradually adds high-to-low resolution subnetworks one by one to form more stages, and connects the multiresolution subnetworks in parallel. Then it conducts repeated multi-scale fusions such that each of the high-to-low resolution representations receives information from other parallel representations over and over, leading to rich high-resolution representations. Thus, our

occupancy encoder constructed by HRNet encoder which composed four stages from HRNet for maintaining high-resolution. And the occupancy encoder has four output variables which size are (H/4, W/4, 128), (H/8, W/8, 256), (H/16, W/16, 512) and (H/32, W/32, 1024) respectively.



**Fig. 2.** An overview of the LCNet model.

The flow encoder is constructed by a basic swin transformer layer [8]. More specifically, each Swin-Transformer module is a two-layer Transformer with both window self-attention and shifted window self-attention. It enables global and intersected attention-based interaction modeling. Thus, the flow encoder can extract critical flow information.



**Fig. 3.** A novel hierarchical convolution network.

### 3.3 Decoder

As shown in Fig. 2, the decoder part of our proposed LCNet consists of the soft STD, ConvLSTM [18] and upsample-Conv3d modules. We design a novel hierarchical convolution network as shown in Fig. 3. The soft STD is similar as HOPE’s decoder STD, which use 3D temporal transposed convolutional network to upsample along time dimension. But not equal to HOPE’s, we attend into soft-attention module to extract main feature in multi-scale features, which aims to extract the most influential features of motion. And we utilize this structure to upsample feature along time dimension, and then aggregate decoded temporal feature by 3d convolution. Then the ConvLSTM module decodes the feature for forecasting time series. Finally, the  $\widehat{O}_k^b$ ,  $\widehat{O}_k^c$  and  $\widehat{F}_k$  are output by upsample-Conv3d modules.

### 3.4 Loss

The model is trained with supervised observed and occluded occupancy and flow field losses, and a self-supervised flow-traced loss. The binary logistic cross entropy loss [5] is adopted to supervise both the observed and occupancy and occluded occupancy prediction. The observed occupancy loss  $L_{obs}$  and occluded occupancy loss  $L_{occ}$  are defined as:

$$L_{obs,occ} = \sum_{t=1}^{T_{pred}} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \mathcal{B}(O_t(x,y), \widehat{O}_t(x,y)) \quad (1)$$

In the above formula,  $\mathcal{B}$  is the binary logistic cross entropy function,  $O_t$  is the predicted occupancy and  $\widehat{O}_t$  is the ground truth occupancy at time step  $t$ .

The flow loss is the MAE [20] regression loss with respect to the ground truth flow. To decouple the flow prediction and occupancy prediction tasks, the loss is further weighted by the ground truth occupancy. The flow loss  $L_{flow}$  is defined as:

$$L_{flow} = \sum_{t=1}^{T_{pred}} \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \|F_t(x,y) - \widehat{F}_t(x,y)\|_1 \widehat{O}_t(x,y) \quad (2)$$

In the above formula,  $F_t$  is the predicted flow and  $\widehat{F}_t$  is the ground truth flow at time step  $t$ .

The flow-traced loss consists of the binary logistic cross entropy loss [5], and focal loss [19] to further boost the Flow-Grounded Occupancy AUC metric. Also, to be consistent with the Flow-Grounded Occupancy AUC metric, instead of recursively applying the warping process on current occupancy  $O_0$ , we directly warp the ground truth  $\widehat{O}_{t-1}$  at the previous time step with the predicted flow  $F_t$ , as shown in the formula:

$$W_t = F_t \circ \widehat{O}_{t-1} \quad (3)$$

And the flow-traced loss  $L_{traced}$  is defined as:

$$L_{traced} = \sum_{t=1}^{T_{pred}} \sum_{x=0}^{w-1} \lambda_b \mathcal{B}(W_t(x,y) O_t(x,y), \widehat{O}_t(x,y)) + \lambda_f \mathcal{F}(W_t(x,y) O_t(x,y), \widehat{O}_t(x,y)) \quad (4)$$

The final loss can be described as:

$$L = \frac{1}{hwT_{pred}} (500L_{obs} + 500L_{occ} + L_{flow} + L_{traced}) \quad (5)$$

## 4 Experiments

### 4.1 Experimental Setup

We train and evaluate our model using the Waymo Open Dataset [21], which includes over 500,000 training samples covering a variety of real-world driving scenarios and dynamic interactions between traffic agents, including vehicles, bicycles, and pedestrians. Each sample consists of 1 s historical trajectory and 8 s future trajectory, with a sampling frequency of 10 HZ. The input and output rasterized image resolutions are  $H = 256$ ,  $W = 256$ , representing an area of  $80 * 80 \text{ m}^2$  in the real world. WOMD splits 485,568 samples for training and some interesting scenarios, including 4499 samples for validation and testing.

To fairly evaluate the performance of our method, we follow the standard metrics proposed in the challenge [5]. The occupancy metrics include AUC and Soft-IoU. Treating the occupancy of each grid cell as a separate binary prediction, the AUC computes  $\text{AUC}(O_t^k, \hat{O}_t^k)$  for agent type k (vehicle/pedestrian) using a linearly-spaced set of thresholds in  $[0, 1]$  to compute pairs of precision and recall values and estimate the area under the PR-curve. And the Soft-IoU metric measures the soft intersection-over-union between ground-truth and predicted occupancy grids. The Soft-IoU can be described as:

$$\text{Soft - IoU}(O_t^k, \hat{O}_t^k) = \frac{\sum_{x,y} O_t^k \cdot \hat{O}_t^k}{\sum_{x,y} O_t^k + \hat{O}_t^k - O_t^k \cdot \hat{O}_t^k} \quad (6)$$

In the above formula, where argument  $(x, y)$  and have been omitted for brevity.

Flow metrics only include EPE, which measures metric measures the mean L2 distance between the ground-truth flow field and predicted flow field. The EPE can be described as:

$$\text{EPE} = \|F_t^k(x, y) - \hat{F}_t^k(x, y)\|_2 \quad (7)$$

### 4.2 Implementation Details

We represent the convolutional bottleneck as the down sampling convolution in the model, which consists of two  $1*1$  convolutional layers and a  $3*3$  convolutional layer. To mitigate over-fitting and accelerate fitting of model, we utilize batch-norm after each  $3*3$  convolution layer. Due to the large scale of input occupancy, we use a distributed training strategy on 8 Nvidia A100 GPUs with a total batch-size of 16. AdamW optimizer is used with an initial learning-rate 2e-3, and the learning-rate decays by a factors of 50% every 3 epochs. The total training epochs are set to 15.

### 4.3 Quantitative Results

As shown in Table 1 above, HOPE achieved the best performance on the benchmark, with a combined AUC of 0.839. But as HOPE describes, it uses large-scale networks such as RegNetX-320G, Swin-Transformer, and sparse convolution, which causes it to require more time and GPU memory for training and inference. In fact, if we need high-accuracy model to predict on our autonomous driving vehicles, it will require chips with more computing power and be more expensive. Nonetheless, there are still certain difficulties in achieving real-time predictions. Look-Around [22] also leverages pre-trained large-scale convolutional networks, just like HOPE [6]. In addition, StrajNet [7] and Vector-Flow [14] convert open datasets into vectors, and then utilize vector networks to extract vector trajectory features.

**Table 1.** Summary of the testing performance on the waymo occupancy and flow prediction benchmark

Metrics	Observed Occupancy		Occluded Occupancy		Flow	Combined	
MODEL	AUC	Soft-IOU	AUC	Soft-IOU	EPE	AUC	Soft-IOU
HOPE [6]	<b>0.803</b>	0.235	0.165	0.017	3.672	<b>0.839</b>	0.633
Look Around [22]	0.801	0.234	0.139	0.029	<b>2.619</b>	0.825	0.549
StrajNet [7]	0.778	0.491	<b>0.178</b>	<b>0.045</b>	3.204	0.785	0.531
Temp-Q	0.757	0.393	0.171	0.041	3.308	0.778	0.465
VectorFlow [14]	0.755	0.488	0.174	0.045	3.583	0.767	0.531
3D-STCNN [23]	0.691	0.412	0.115	0.021	4.181	0.733	0.468
Motionnet [24]	0.694	0.411	0.141	0.032	4.275	0.732	0.469
FTLS	0.618	0.318	0.085	0.019	9.612	0.689	0.431
<b>OURS</b>	0.789	<b>0.588</b>	0.102	0.034	3.279	0.818	<b>0.674</b>

Our method can speed up model inference, reduce model parameters, and make it easier to deploy on autonomous vehicles. However, compared with fully convolutional networks such as HOPE and Look-Around, StrajNet and Vector-Flow cannot achieve high-precision prediction. Combining the priorities of HOPE and StrajNet and mitigating their problems, our proposed method achieves high-precision prediction, with a combined AUC reaching 0.818. And our method has achieved two best metrics Soft-IoU for observed and combined occupancy prediction. Compared to the StrajNet, they improved by 19% and 23% respectively. Furthermore, as shown in Table 2, our method requires less time for model inference. Compared to HOPE and StrajNet, our model takes approximately 1ms to infer model, which is a time reduction of 97% and 41% respectively.

**Table 2.** Comparison of time-costs and model-param.

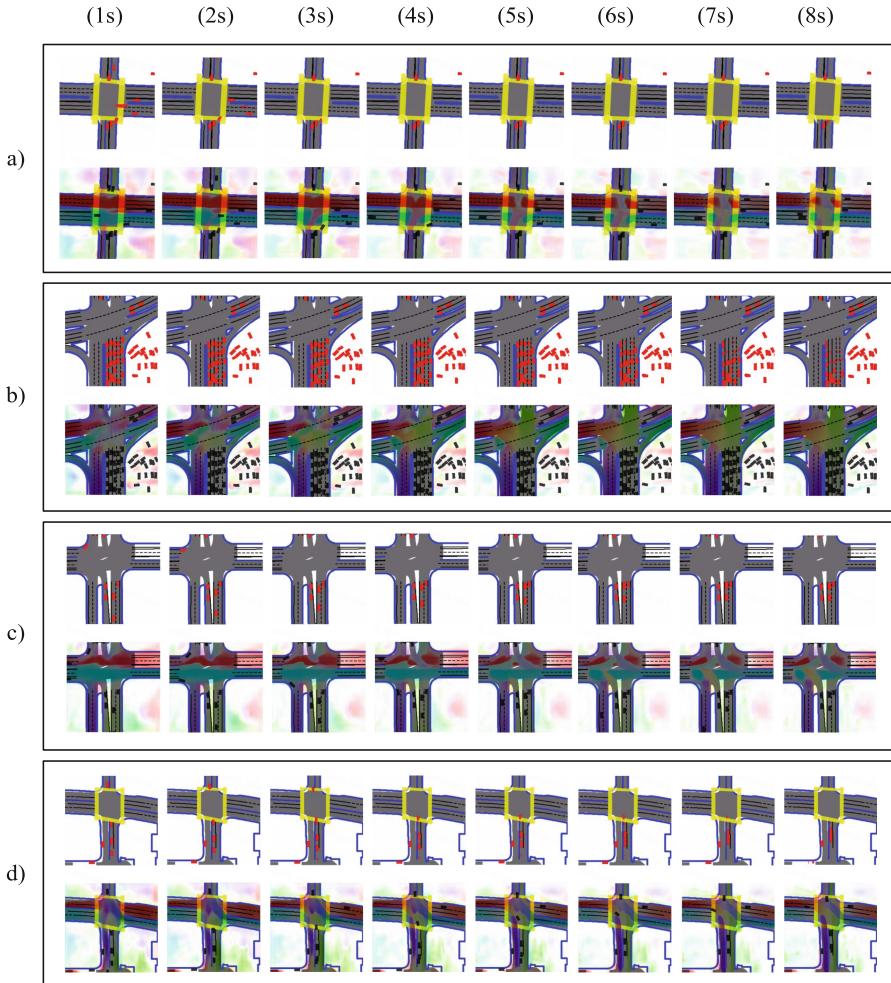
Model	Time-Costs/s	Model-Param/Mb
HOPE	4.48617	151.484
StrajNet	0.17761	<b>23.434</b>
<b>OURS</b>	<b>0.10650</b>	93.734

#### 4.4 Qualitative Results

To intuitively evaluate our test results, we visualize the test results for several representative driving scenarios in Fig. 4. In Fig. 4 a) the lateral flow was divided by traffic signals gradually. As time went on, we hypothesize that the traffic lights had phased out, vehicles were parked at crosswalks, and flow was diverted at intersections. In Fig. 4b), the left-turn flow gradually became more apparent. In the first few seconds, left-turning vehicles stopped at the intersection. From the 6th second, the left-turning flow began to continue and the lateral flow began to be cut off. In Fig. 4c), there were multiple vehicles parked in front of the intersection. As time went on, the bottom-up flow became gradually apparent, including three directions: turning left, going straight, and turning right. In Fig. 4d), the multimodal output of the model was showed. In the first few seconds, the top-down flow branched in two directions, including going straight and turning left. And in the final seconds, the flow started to be single branch gradually, which only included left-turn flow.

#### 4.5 Ablation Study

We conduct an ablation study to investigate the impact of the key modules in our proposed framework, namely the flow separation branching module and the Soft-STD temporal dimension up sampling module. Therefore, we train two ablation models without Soft-STD module and Flow-Separate branch separately, and all ablation models are validated on the same test set. We report the AUC of the occupancy metrics and the flow EPE in Table 3, which show that the overall performance of the two ablation models is degraded. Due to the difference between flow and occupancy data distributions, using the same encoder to extract occupancy and flow features degrades prediction performance. Creating a separate branch to separately encode flow data can help better extract flow and occupancy characteristics. In addition, the import of Soft-STD module improves the prediction performance. Compared to the ablated model without the Soft-STD module, the combined AUC improved by 2.9%, the flow EPE reduced by 18.9%. Since the Soft-STD module can better select the main features that affect future trajectory prediction and rich time main features, combining the Soft-STD module can further improve the model performance, especially for prediction along the time dimension.



**Fig. 4.** Prediction visualization of 4 scenes. For each scene, the first row shows the predicted occupancy and the second row shows the predicted flow.

**Table 3.** Ablation studies of flow-separate branch and soft-STD.

Flow-Sep	Soft-STD	Observed AUC	Occluded AUC	GF_AUC	Flow-EPE
✗	✗	0.7507	0.0822	0.7715	4.533
✓	✗	0.7676	0.9821	0.7951	4.046
✓	✓	<b>0.789</b>	<b>0.102</b>	<b>0.818</b>	<b>3.279</b>

## 5 Conclusion

In this paper, we propose a lightning hierarchical convolution network for occupancy flow prediction. The multiple inputs (map, flow and occupancy) are encoded by multi-stage hrnet module and swin-transformer module separately which can learn high-resolution information. And the hierarchical convolution decoder captures multi-scale features efficiently. Finally, we get occupancy and flow prediction by a single conv-lstm layer. Validated by lots of experiments on Waymo Open Dataset, our method achieves faster inference and maintains high-precision predictions. And the ablation studies show that the separate flow branch and the Soft-STD module can significantly improve prediction performance.

## References

- Parekh, D., et al.: A review on autonomous vehicles: progress, methods and challenges. *Electronics* **11**(14), 2162 (2022)
- Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2018)
- Liang, M., et al.: Learning lane graph representations for motion forecasting. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. Springer International Publishing (2020)
- Zhou, Z., et al.: Hivt: hierarchical vector transformer for multi-agent motion prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
- Mahjourian, R., et al.: Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robot. Automation Lett.* **7**(2), 5639–5646 (2022)
- Hu, Y., et al.: HOPE: Hierarchical spatial-temporal network for occupancy flow prediction. arXiv preprint [arXiv:2206.10118](https://arxiv.org/abs/2206.10118) (2022)
- Liu, H., Huang, Z., Chen, Lv.: Multi-modal hierarchical transformer for occupancy flow field prediction in autonomous driving. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2023)
- Liu, Z., et al.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)
- Sun, K., et al.: High-resolution representations for labeling pixels and regions. arXiv preprint [arXiv:1904.04514](https://arxiv.org/abs/1904.04514) (2019)
- Nikhil, N., Morris, B.T.: Convolutional neural network for trajectory prediction. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops (2018)
- Zhang, C., Berger, C., Dozza, M.: Social-IWSTCNN: a social interaction-weighted spatio-temporal convolutional neural network for pedestrian trajectory prediction in urban traffic scenarios. In: 2021 IEEE Intelligent Vehicles Symposium (IV). IEEE (2021)
- Bansal, M., Krizhevsky, A., Ogale, A.: ChauffeurNet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint [arXiv:1812.03079](https://arxiv.org/abs/1812.03079) (2018)
- Hu, A., et al.: Fiery: future instance prediction in bird’s-eye view from surround monocular cameras. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)
- Huang, X., et al.: VectorFlow: Combining Images and Vectors for Traffic Occupancy and Flow Prediction. arXiv preprint [arXiv:2208.04530](https://arxiv.org/abs/2208.04530) (2022)

15. Wang, J., et al.: Deep high-resolution representation learning for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **43**(10), 3349–3364 (2020)
16. Yuan, Y., Chen, X., Wang, J.: Object-contextual representations for semantic segmentation.“Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16. Springer (2020)
17. Cheng, B., et al.: Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
18. Shi, X., et al.: Convolutional LSTM network: a machine learning approach for precipitation nowcasting. *Advances in neural information processing systems* **28** (2015)
19. Lin, T.-Y., et al.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (2017)
20. Willmott, C.J., Matsuura, K.: Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Res.* **30**(1), 79–82 (2005)
21. Ettinger, S., et al.: Large scale interactive motion forecasting for autonomous driving: the waymo open motion dataset. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)
22. Dmytro, P.: Waymo open dataset occupancy and flow prediction challenge solution: Look around (2022). <https://storage.googleapis.com/waymo-uploads/files/research/OccupancyFlow/Dmytro1.pdf>
23. He, Z., Chow, C.-Y., Zhang, J.-D.: Stcnn: a spatio-temporal convolutional neural network for long-term traffic prediction. In: 2019 20th IEEE International Conference on Mobile Data Management (MDM), pp. 226–233. IEEE (2019)
24. Wang, Y., Long, M., Wang, J., Yu, P.S.: Spatiotemporal pyramid network for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1529–1538 (2017)



# FedTS: Leveraging Teacher-Student Architecture in Federated Learning Against Model Heterogeneity in Edge Computing Scenarios

Zihong Lin<sup>✉</sup>, Yucheng Tao<sup>✉</sup>, and Haopeng Chen<sup>(✉)</sup>

Shanghai Jiao Tong University, Shanghai 200240, China  
`{923048992,taoyucheng,chen-hp}@sjtu.edu.cn`

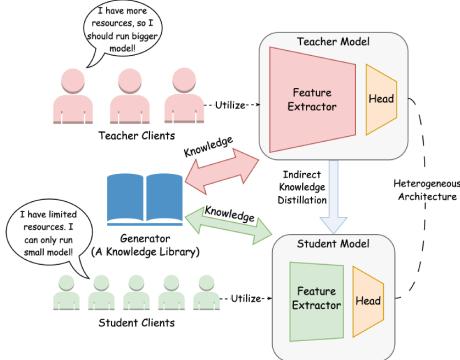
**Abstract.** Current federated learning research assumes uniform model architecture across client devices. However, in edge computing, client resources often vary significantly. Using the same model may waste resources in powerful clients. Although adapting model complexity to client resources is reasonable, it introduces model architecture heterogeneity, which challenges many existing algorithms. Hence, we propose the FedTS algorithm, using data-free knowledge distillation to enable large models to assist smaller models, addressing both model and resource diversity in federated learning. The experiments show that our algorithm outperforms the baseline on multiple datasets(12.65% on CIFAR-10 and 8.73% on CINIC-10, compared to the baseline).

**Keywords:** Federated Learning · Heterogeneous Model Architecture · Knowledge Distillation

## 1 Introduction

A deep neural network model typically needs substantial data to achieve high performance. However, data collection can be costly due to the dispersion of data and the privacy and permission issues that arise when attempting to integrate it. Thus, federated learning, as an emerging machine learning paradigm, has been proposed to overcome the limitations of traditional centralized learning and address the challenge of achieving global model training while protecting user privacy [6, 21, 24]. For example, FedAvg [26] illustrates a scenario in which mobile devices collaborate to train a model without transmitting their data to a central server. It proposes a classic model for federated learning, where  $K$  clients train their models locally based on their respective datasets  $D_i$  and upload their model weights to the server. The server aggregates the uploaded weights to form a global model and broadcasts it to each client (Fig. 1).

Numerous studies focus on addressing the challenge of heterogeneous data, often characterized by non-iid data distribution [19, 23, 40]. They aim to alleviate the performance degradation resulting from heterogeneous data through



**Fig. 1.** The proposed architecture. Clients have imbalanced resources and non-iid data. Resource-rich clients assume the role of teachers, employing larger models to effectively utilize their resources and improve the overall average performance of the model. All clients collectively train a generator, enabling knowledge sharing without compromising privacy and fulfilling data-free and indirect knowledge distillation.

techniques such as data augmentation [5, 15], incorporating regularization terms [29, 36, 38], and devising specialized aggregation algorithms [33, 36, 37]. However, to the best of our knowledge, few studies have focused on model heterogeneity in federated learning.

Edge resources are typically constrained, heterogeneous, and dynamic, making resource management a significant challenge [10]. Resource imbalance is common in edge computing scenarios. Edge computing nodes include smartphones, personal computers, single-chip microcomputers, etc. [10, 22]. Their computing capability differs significantly. Consider an edge computing setting where clients have different resource capacities and can support various complexities of deep learning models. We assume clients with limited resources can only execute small models like MLPs, while others with more abundant resources can handle larger models, such as ResNets. In traditional federated learning, these more powerful clients cannot fully utilize their resource advantages (They have to use underperforming smaller models in combination with other clients). Furthermore, it is often challenging to obtain a public dataset because of privacy and permission issues; the need for publicly available datasets challenges implementing traditional knowledge distillation (the teachers need to share a joint dataset with the students, violating the federated learning rule).

Our research focuses on fully harnessing the resource advantages of specific clients in situations where model architectures differ, aiming to drive the rapid convergence of smaller models and enhance overall model performance. In our approach, two types of models are utilized, namely the teacher and student models, with the former containing more parameters. We use a data-free knowledge distillation [41] method to facilitate mutual learning between the teacher and the student, eliminating the necessity for public datasets.

Our contribution lies in the proposal of the FedTS algorithm, which not only maintains the performance(or minimal performance degradation) of the original smaller model but also enhances resource utilization, leading to a substantial improvement in client average accuracy. This algorithm offers a solid approach for judicious resource utilization and addressing model heterogeneity challenges in edge computing environments.

## 2 Related Work

### 2.1 Traditional Federated Learning

Many existing federated learning algorithms primarily target model optimization. Some algorithms enhance feature alignment through the inclusion of specific regularization terms. For instance, FedPAC [36] applies the L2-normalization between the local and global centroid representation as the regularization term to align the latent representations of clients to a common center. Meanwhile, FedCR [38] and FedSR [29] employ CMI(conditional mutual information) based on information theory to restrict information stored in the latent representation, encouraging the model to learn only essential information while disregarding falsely correlated background. Additionally, FedProx [18] introduces a proximal term to ensure the local model adheres to the global model. Some algorithms employ unique aggregation techniques. For example, FedPAC [36] quantifies the collective benefit of classifiers from individual clients, while pFedGraph [37] computes the similarity matrix of client model weights to derive aggregation weights. Both methods convert weight aggregation into an optimization problem. Although these algorithms effectively address non-iid data distribution, they assume clients share the same model architecture.

### 2.2 Heterogeneous Federated Learning

Currently, there are ongoing research endeavors addressing model(or resource) heterogeneity. FedCS [31] and FedMCCS [1] try to tackle resource heterogeneity problems, but they are based on the same model architecture. FedHiSyn [17] clusters devices into a small number of categories based on their computing capacity, but it still assumes that all devices run the same model, which does not fully exploit the potential of resource-rich clients. FedClassAvg [11] introduces a specialized and effective federated learning algorithm addressing heterogeneous model scenarios. However, its experiments solely focus on scenarios with minor differences in the number of parameters. FedDF [20] applies ensemble distillation to fuse heterogeneous models. Nevertheless, FedDF requires an additional dataset for the distillation operations, bringing significant overhead during the training phase. FedHeNN [25] empowers individual clients to use personalized models without enforcing a shared architecture, yet the approach demands additional public datasets. KT-pFL [39], based on knowledge transfer, employs a knowledge coefficient matrix to facilitate federated learning for clients with varying models, albeit requiring a shared dataset. On the other hand, FedGen

[41] utilizes a generator to provide data-free knowledge distillation. While it's proved effective by experiments, the algorithm needs further adaptation to align with the specific problem context we have introduced.

In summary, existing research either overlooks the impact of model heterogeneity, necessitates additional public datasets, or does not align with the scenarios we have introduced. Conversely, our algorithm can facilitate collaborative federated learning for two significantly different models without the requirement for additional datasets, thereby enhancing resource utilization and model average performance in edge computing scenarios.

### 3 Methodology

#### 3.1 Problem Formulation

In our hypothetical scenario,  $K$  clients partake in federated learning, with those possessing ample resources forming the teacher set  $T$ , while the remainder as the student set  $S$ . We split a neural network into a feature extractor  $f(\theta^f; x)$  and a head  $g(\theta^g; z)$  (a classifier in classification problem). The feature extractor takes input  $x$  and outputs a latent representation  $z$ , while the head receives  $z$  and performs tasks such as classification. We reconcile between fully heterogeneous and homogeneous models by having all teacher(student) models share the same feature extractor structure. Teachers and students form distinct logical clusters, and the teacher and student clusters each aggregate their global feature extractors like FedAvg [26]. All models employ the same head structure, which is typically simple; in classification scenarios, the head is often the final fully connected layer, and FedPAC [36] directly assumes it to be a linear classifier. By sharing only a minimal amount of structure, the feature extractor can adopt a more flexible framework, enabling resource-rich clients to utilize deeper neural networks for more accurate feature extraction.

Based on the above assumption, our objective can be written as:

$$\min_{\theta_k} \frac{1}{K} \sum_{k=1}^K \mathcal{L}(g(\theta_k^g; f_k(\theta_k^f; \mathbf{x}_k)), \mathbf{y}_k) \quad (1)$$

Here  $D_k = (\mathbf{x}_k, \mathbf{y}_k)$  is the local dataset of client  $k$ , and we designate the feature extractor as  $f_k$ .

#### 3.2 FedTS Algorithm

Complex models are often capable of extracting more accurate and deeper features. Effectively leveraging the knowledge acquired by large models to enhance the learning of smaller models is a critical and challenging task, and multiple studies suggest that knowledge distillation [8] can accomplish this goal. The objective of knowledge distillation in federated learning [20] is proposed in the following manner:

$$\min_{\theta_k} \frac{1}{K} \sum_{k=1}^K D_{KL}(g(\theta_k^g; f(\theta_k^f; \mathbf{x})) || g(\theta^g; f(\theta^f; \mathbf{x}))) \quad (2)$$

The primary issue with this approach is the requirement for a shared dataset  $\hat{D}$ . Due to privacy and permission concerns, not all federated learning participants have access to a shared dataset, leading to a lack of generalization for algorithms reliant on it. To implement knowledge distillation in federated learning without compromising privacy or becoming data-dependent, we introduce a generator  $G(w^G; y) : \mathcal{Y} \rightarrow \mathcal{Z}$  that maps labels  $y$  to latent representations  $z$ . Here,  $w^G$  is the parameter of the generator.

**Optimizing Generator.** The generator can be trained by the server during the aggregation phase using the uploaded model weights of the clients. Its objective function can be expressed as:

$$\begin{aligned} G^* &= \arg \min_G \mathbb{E}_{y \sim p(y)} \mathbb{E}_{z \sim r(z|y)} \sum_{k=1}^K w_k \mathcal{L}(g(\theta_k^g; z), y) \\ &\approx \arg \min_G \sum_{c \in C_s} \sum_{k=1}^K w_k^c \mathcal{L}(g(\theta_k^g; z_c), c) \end{aligned} \quad (3)$$

Here, the loss function  $\mathcal{L}$  is cross-entropy loss. We randomly sample some labels  $c \in C_s$  as a training dataset ( $|C_s| = N_s$ ,  $N_s$  is a hyperparameter). We denote the generator  $G$  as a variational posterior probability  $r(z|y)$ , and  $z_c \sim r(z|y = c)$  is the latent representation generated by the generator.  $w_k^c$  is a weight term used to control the extent to which the generator learns from different clients:

$$w_k^c = \eta p_k^c r_k \text{softmax}\left(\frac{1}{H(g(\theta_k^g; z_c))}\right) \quad (4)$$

where  $\eta$  is a hyperparameter acting as a learning rate.  $p_k^c = \frac{|D_k^c|}{\sum_{k=1}^K |D_k^c|}$  is the proportion of dataset size for client  $k$  given label  $c$ , which indicates that the client with larger training datasets may be more valuable for learning.  $r_k$  is another control weight, which can be computed by:

$$r_k = \begin{cases} 1 & \text{if } k \in T \\ \frac{E_{cur}}{E_{max}} \tau & \text{if } k \in S \end{cases} \quad (5)$$

where  $E_{cur}$  is the current global training epoch,  $E_{max}$  is the max number of epochs,  $\tau \in [0, 1]$  is a hyperparameter (the bigger it is, the more the generator will learn from students). Constrained by its capabilities, small models may face challenges in extracting precise and intricate features. Nonetheless, the data and knowledge from clients with small models are equally significant. As a result, we enable the teacher model to take the lead at the beginning of training, guiding the student model in swiftly acquiring feature extraction capabilities. Subsequently, the student progressively engages in the training.

Meanwhile, we should learn not only from clients with larger training volumes but also from those with higher confidence. Previous researches [2, 32] suggest

that lower information entropy indicates greater confidence. Therefore, we compute the reciprocal of the information entropy and normalize it through softmax to encourage the generator to prioritize learning from highly confident clients.

In practice, the generator can be either deterministic or probabilistic. The former directly generates a latent representation  $z$ . To generate different latent representations given the same label  $y$ , the generator will also accept a random noise  $\epsilon$  as input, namely  $G \sim r(z|y, \epsilon)$ , to produce distinct outputs. Instead, the latter produces a Gaussian distribution( $\mu_z, \sigma_z^2$ ). We follow VAE [13] to use reparameter trick to generate  $z \sim (\mu_z, \sigma_z^2)$ (Eq. 6). For clarity in writing, we have omitted  $\epsilon$  in the paper.

$$z = \mu_z + \sigma_z \epsilon, \epsilon \sim (0, 1) \quad (6)$$

**Optimizing Local Model.** Upon completion of the generator training, the server will distribute the generator's weights to each client. Throughout the local training phase, the generator can function as a supportive repository of knowledge, offering guidance for training both the feature extractor and the head. The loss function of local training can be written as:

$$\begin{aligned} \mathcal{L}_k = & \lambda_1 \mathcal{L}^{CE}(g(\theta_k^g; f_k(\theta_k^f, x_k)), y_k) \\ & + \lambda_2 \mathcal{L}^{MSE}(G(w^G; y_k), f_k(\theta_k^f, x_k)) \\ & + \lambda_3 \mathcal{L}^{CE}(g(\theta_k^g; G(w^G; y_s)), y_s) \\ & + \lambda_4 \mathcal{L}^{norm}(f_k(\theta_k^f, x_k)) \end{aligned} \quad (7)$$

where  $(x_k, y_k) \in D_k$ ,  $y_s$  represents randomly sampled labels, its size  $|y_s|$  is determined by a hyperparameter  $N_g$ , while  $\lambda_1, \dots, \lambda_4$  denote learning rate hyperparameters. CE in the first term of Eq. 7 denotes cross-entropy, which refers to the prediction error.

In the second term, MSE stands for mean squared error, quantifying the difference between the local model's latent representation and the generator's output, often termed a latent loss. This term aims to guide the model in generating the right latent representation.

Besides the local dataset, we will randomly sample labels, and the generator will produce additional latent representations as supplementary data for head training in each round. The third term represents the error in knowledge distillation for the head(The head takes the generated latent representation  $G(w^G; y_s)$  as input and outputs a corresponding prediction  $y' = g(\theta_k^g; G(w^G; y_s))$ ). Since the distribution of generator  $r(z|y) \approx p(z|y)$  (Eq. 8, will be explained later), the prediction  $y'$  should align with the sampled label  $y_s$ . This term guides the head in better classifying given latent representation. Meanwhile, the generated supplementary dataset extends the available dataset and improves the generalization of the head.

To obtain a more compact hidden feature space, we introduced an L2 regularization term(the fourth term) to do explicit feature alignment(in this manner, we guide the generator to generate a latent representation close to zero) [29].

$$\begin{aligned} & \max_{\theta_k} \mathbb{E}_{y \sim p(y)} \mathbb{E}_{z \sim r(z|y)} [\log(y|z; \theta_k)] \\ & \equiv \min_{\theta_k} \mathbb{E}_{y \sim p(y)} \mathbb{E}_{z \sim r(z|y)} D_{KL}[(r(z|y) || p(z|y; \theta_k))] \end{aligned} \quad (8)$$

The Eq. 8 suggests that, for any variational posterior probability  $r(z|y)$ , it will gradually converge towards the true posterior distribution  $p(z|y)$  with progressing training [41]. The second term in Eq. 7 directs the local feature extractor to generate latent representations that closely resemble  $r(z|y)$ . Because the generator is shared, eventually,  $p_1(z|y) \approx p_2(z|y) \approx \dots \approx r(z|y)$ , resulting in implicit feature alignment; moreover, as the generator's training involves all clients, it amalgamates knowledge from each client, enabling the local model's head to encounter additional latent representations, thus enhancing generalization.

**Global Aggregation.** For classification problems, the head function  $g(\theta_k^g; z) \sim p_k(y|z)$ . In scenarios where the data is non-iid, the label distribution among clients might exhibit a significant imbalance, resulting in a departure of  $p_k(y|z)$  from other clients. In extreme cases, some clients may possess data exclusively from a single class, causing their feature extractor to generate only latent representations for that class. Consequently, this can lead to overfitting during the training of the head. Our approach aims to alleviate this issue by reducing the weights assigned to such clients during the aggregation process, yielding more accurate global aggregation results.

$$\theta_{\text{global}}^g = \sum_{k=1}^K \alpha_k \theta_k^g \quad (9)$$

Eq. 9 illustrates the proposed approach for aggregating head weights. In contrast to FedAvg [26], which utilizes the dataset size ratio  $p_k = \frac{|D_k|}{\sum_{k=1}^K |D_k|}$ , we employ the adapted weights  $\alpha_k$ . We propose the following objective to compute  $\alpha_k$ .

$$\min_{\alpha_1, \dots, \alpha_K} \sum_{k=1}^K (\alpha_k - p_k)^2 + \gamma \sum_{k=1}^K \alpha_k d_k \quad (10)$$

where the first term is proved valid in previous work [37], while the second term enables divergent weights to receive lesser aggregation weights.  $\gamma$  is a hyper-parameter and  $d_k = -\frac{1}{K} \sum_{j=1}^K \cos(\Delta\theta_k^g, \Delta\theta_j^g)$  represents the average cosine distance (negative cosine similarity) between the changes in the current head weights and the changes in the head weights of other clients ( $\Delta\theta_k^g = \theta_k^g - \theta_{\text{init}}^g$ ,  $\theta_{\text{init}}^g$  refers to the initial weights).

The equation be transformed into a convex optimization problem:

$$\begin{aligned} A &= \arg \min_{\mathbf{x}} \mathbf{x}^T \mathbf{x} + (-2P + \gamma D)^T \mathbf{x} \\ \text{s.t. } & I^T \mathbf{x} = 1, -\mathbf{x} \leq 0, \end{aligned} \quad (11)$$

**Algorithm 1.** FedTS Algorithm

---

**Input:** client number  $K$ , local datasets  $D_1, D_2, \dots, D_K$ , teacher set  $T$ , student set  $S$ , local training epochs  $E$ , global communication epochs  $M$ .

**Output:**  $\theta_k^f, \theta_k^g, w^G, k \in \{1, \dots, K\}$ .

Initialize  $\theta_{\text{teacher}}^f, \theta_{\text{student}}^f, \theta_{\text{global}}^g, w^G$ .

**procedure** GlobalUpdate

- for** Each communication epoch  $m = 1, \dots, M$  **do**
- Sample clients to participate in epoch  $m$ .
- for** Each selected client  $k$  **do**
- $\theta_{\text{global}}^f = \begin{cases} \theta_{\text{teacher}}^f, & \text{if } k \in T \\ \theta_{\text{student}}^f, & \text{otherwise} \end{cases}$
- $\theta_k^f, \theta_k^g \leftarrow \text{ClientUpdate}(\theta_{\text{global}}^f, \theta_{\text{global}}^g, w^G)$
- end for**
- $w^G \leftarrow \text{TrainGenerator}(\theta_1^g, \dots, \theta_K^g)$  (via Eq.3)
- $\theta_{\text{teacher}}^f \leftarrow \sum_{k \in T} p_k \theta_k^f$
- $\theta_{\text{student}}^f \leftarrow \sum_{k \in S} p_k \theta_k^f$
- $\theta_{\text{global}}^g \leftarrow \sum_{k \in T} \alpha_k \theta_k^g$  (via Eq.9)
- end for**

**procedure** ClientUpdate( $\theta_{\text{global}}^f, \theta_{\text{global}}^g, w^G$ )

- for** Each local epoch  $e = 1, \dots, E$  **do**
- for** Mini-batch epoch  $b \in D_k$  **do**
- Update local weights via Eq.7.
- end for**
- end for**
- return** local weights  $\theta_k^f, \theta_k^g$

---

where  $A = [\alpha_1, \alpha_2, \dots, \alpha_K] \in \mathbb{R}^K$ ,  $P = [p_1, p_2, \dots, p_K] \in \mathbb{R}^K$  and  $D = [d_1, d_2, \dots, d_K] \in \mathbb{R}^K$  are three vectors.

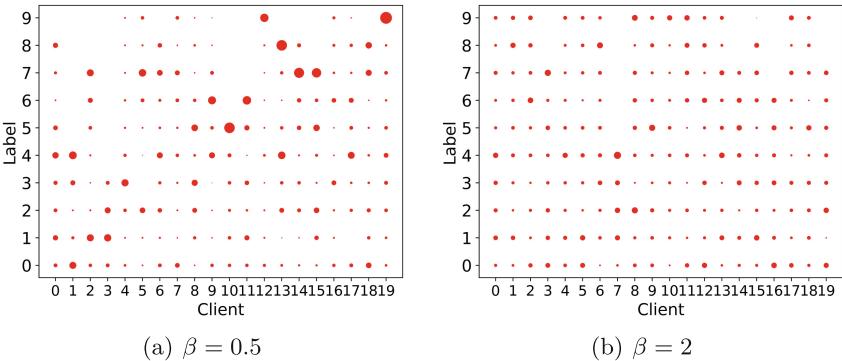
**Overall Workflow.** Algorithm 1 presents the comprehensive workflow of the FedTS algorithm. During the local training phase, parameter updates are based on the loss function 7. We train the generator using the uploaded weights in the global aggregation phase. Subsequently, the teacher and student feature extractors are aggregated independently, followed by the aggregation of the global classifier.

Privacy is guaranteed since we transmit the generator (knowledge) rather than the data. Notice that the generator typically demands minimal parameters as it does not require direct feature extraction. Instead, it adapts a posterior probability distribution. Meanwhile, it is trained on the server side(which could be a data center with powerful computing ability). So, it will bring minimal additional computational burden to the clients.

## 4 Experiment Results

### 4.1 Experiment Setup

**Datasets.** We use four popular datasets: MNIST [4] with 70,000 images, Fashion-MNIST [35] with 70,000 images, CIFAR-10 [14] with 60,000 images and CINIC-10 [3] with 270,000 images. To simulate heterogeneous data, we follow previous work [11, 37] to utilize the Dirichlet distribution to generate non-iid client label distributions. The Dirichlet distribution is governed by a parameter  $\beta$ , whereby a lower  $\beta$  value results in a more imbalanced label distribution. Figure 2 shows the label distribution using different  $\beta$ s on the Fashion-MNIST dataset.



**Fig. 2.** Non-iid label distribution of different  $\beta$  on Fashion-MNIST dataset. The size of each point represents the number of labels. The X-axis refers to clients while the y-axis refers to labels.

**Models.** In the context of our scenario modeling, we assume that teachers possess ample resources, enabling them to run complex models. Conversely, students are constrained by limited resources, thus restricting their ability to run only small-scale models. We employ a simple MLP for the student model regarding the feature extractor. For the teacher model, we utilize ResNet18 as the backbone (which has a parameter count two orders of magnitude higher than the former, as shown in Table 1). Meanwhile, we use a single fully connected layer as the head for all models. We employ a 128-dimensional latent representation to capture deep-level features adequately.

Previous work has proved that the architecture of the generator won't significantly affect the model's performance [41]. We utilize a simple MLP for the generator so that the generator won't add more burden to the student clients.

**Experiment Settings.** We follow previous work(FedPAC [36], FedGen [41], etc.) to use 20 clients. We ensure that all clients fully participate in all training rounds and perform a complete iteration of their local datasets during each local training round. This approach is beneficial in reducing communication

frequency since, within edge computing scenarios, communication overhead is often non-negligible.

**Metrics.** To check the generalization ability of models, we compute the accuracy using a global test dataset instead of each client’s local test dataset. For example, we split the CIFAR-10 dataset into training and test datasets. Then, we split the training dataset into  $K$  parts based on Dirichlet distribution while keeping the test dataset as the global test dataset.

A subset of clients is designated as teachers, while the remaining serve as students. We compare the model performance by varying the percentages of teachers(40%, 20%) to showcase the impact of teacher percentages on the model’s performance.

**Hyperparameters.** We set up the hyperparameters following some previous work [41]. You can check more details in the code we provided.

All experiments were performed on our server using Ubuntu 20.04, Python 3.11, and PyTorch 2.1.0.

## 4.2 Ablation Study

**Baselines.** We select FedAvg [26] as one of our baseline methods, widely acknowledged as the SOTA(state-of-the-art) approach in federated learning. We employ FedAvg’s training results solely using the student model as one baseline. Additionally, we introduce a modified version of FedAvg called FedAvg-TS, which incorporates a Teacher-Student Architecture. We aim to evaluate whether integrating the teacher-student architecture adversely affects the average performance of clients utilizing the student model.

Considering the significant differences in the features extracted by the teacher and student models, a common way to align the features is by calculating the center of the latent representation  $\bar{z}$  and using L2 normalization  $\|z - \bar{z}\|_2^2$  as a regularization term. Therefore, we refer to this method as FedL2Reg [29], serving as one of our baselines. In addition, we also employ other practical federated learning algorithms such as FedProx [18] and pFedGraph [37] as our baselines (they are modified to apply the teacher-student architecture).

FedClassAvg [11] is an algorithm focusing on heterogeneous models, so we also use it as a baseline.

**Table 1.** Comparison on the number of model parameters for different datasets. The teacher model is much bigger than the student model, while the generator is similar to the student model.

Dataset	Student	Teacher	Generator
MNIST	$1.2 \times 10^5$	$1.1 \times 10^7$	$1.4 \times 10^5$
Fashion-MNIST	$1.2 \times 10^5$	$1.1 \times 10^7$	$1.4 \times 10^5$
CIFAR-10	$4.1 \times 10^5$	$1.1 \times 10^7$	$1.4 \times 10^5$
CINIC-10	$4.1 \times 10^5$	$1.1 \times 10^7$	$1.4 \times 10^5$

**Table 2.** Student and teacher model average accuracy comparisons on CIFAR-10 and CINIC-10 using 20 clients. The number of teachers being 0 means we use only student models.

Method	Teachers	Model	CIFAR-10		CINIC-10	
			Dir(0.5)	Dir(2)	Dir(0.5)	Dir(2)
FedAvg	0	Student	35.23±0.00	43.07±0.00	29.74±0.00	36.07±0.02
FedAvg-TS	4	Student	34.90±0.46	44.04±0.29	29.00±0.06	35.57±0.00
		Teacher	55.80±2.83	72.55±1.78	49.06±0.03	56.72±0.00
FedAvg-TS	8	Student	34.85±0.69	42.53±0.04	30.10±0.01	35.20±0.00
		Teacher	58.73±0.45	74.24±0.10	46.68±0.46	57.66±0.00
FedProx	4	Student	34.24±0.07	42.17±0.02	29.04±0.02	35.55±0.01
		Teacher	61.04±0.32	73.82±0.10	48.97±0.36	56.81±0.15
FedProx	8	Student	34.46±0.02	42.53±0.02	30.12±0.03	35.18±0.02
		Teacher	60.97±0.02	74.37±0.20	46.70±0.12	57.31±0.02
FedL2Reg	4	Student	37.28±0.02	42.00±0.00	30.92±0.00	34.95±0.01
		Teacher	56.12±0.05	72.83±0.05	47.65±0.03	57.19±0.03
FedL2Reg	8	Student	36.76±0.02	42.00±0.01	32.63±0.01	34.70±0.03
		Teacher	53.98±0.02	71.16±0.02	44.18±0.04	57.49±0.04
pFedGraph	4	Student	34.35±0.01	42.13±0.05	29.04±0.02	35.48±0.01
		Teacher	61.12±0.07	74.38±0.26	48.75±0.19	56.41±0.15
pFedGraph	8	Student	34.49±0.01	42.55±0.03	30.13±0.00	35.15±0.01
		Teacher	61.27±0.01	74.43±0.07	46.64±0.21	57.41±0.13
FedClassAvg	4	Student	34.54±0.00	40.98±0.01	25.29±0.01	29.14±0.01
		Teacher	61.33±0.04	76.69±0.48	50.82±0.01	58.34±0.01
FedClassAvg	8	Student	34.47±0.02	40.45±0.01	27.01±0.01	28.88±0.01
		Teacher	61.36±0.06	75.25±0.20	47.96±0.01	59.01±0.01
<b>FedTS(Ours)</b>	4	Student	36.26±0.00	43.37±0.02	29.74±0.01	35.78±0.01
		Teacher	60.49±0.10	74.45±0.04	49.92±0.25	57.45±0.11
<b>FedTS(Ours)</b>	8	Student	35.89±0.01	43.45±0.00	30.73±0.02	35.63±0.01
		Teacher	60.17±0.00	74.13±0.00	47.21±0.01	58.56±0.01

**Model Performance Analysis and Comparison.** We aim to improve **overall model performance** while maintaining the student and teacher model performance. In this way, the teacher models utilize its rich resources while the students learn better under the guidance of the teachers. It's a win-win situation.

Table 2 and 4, 3 and 5 display the average accuracy of student models  $\bar{A}_{\text{student}}$ , average accuracy of teacher models  $\bar{A}_{\text{teacher}}$  and all models  $\bar{A}$  over four datasets for various algorithms. The bolded values are the highest ones, while the blue-labeled ones are the second-highest. The results demonstrate that utilizing the teacher-student architecture significantly improves  $\bar{A}$ . As an example, when considering  $\beta = 0.5$  and  $|T| = 8$ , our algorithm achieved a noticeable **12.65%** improvement compared to the baseline FedAvg on the **CIFAR-10** dataset, **8.73%** on the **CINIC-10**, etc. This is because the involvement of complex models naturally enhances overall performance.

**Table 3.** Model average accuracy comparisons on CIFAR10 and CINIC-10 using 20 clients. The number of teachers being 0 means we use only student models. We've bolded the highest accuracy and labeled the second-highest accuracy as blue.

Method	Teachers	CIFAR-10		CINIC-10	
		Dir(0.5)	Dir(2)	Dir(0.5)	Dir(2)
FedAvg	0	35.23±0.00	43.07±0.00	29.74±0.00	36.07±0.02
FedAvg-TS	4	39.07±0.20	49.73±0.12	33.01±0.04	39.80±0.00
FedAvg-TS	8	44.40±0.60	55.21±0.06	36.74±0.18	<b>44.18±0.00</b>
FedProx	4	39.60±0.01	48.50±0.01	33.05±0.05	39.80±0.02
FedProx	8	45.06±0.01	55.27±0.09	36.75±0.03	44.03±0.02
FedL2Reg	4	41.05±0.02	48.17±0.01	34.26±0.01	39.40±0.01
FedL2Reg	8	43.65±0.01	53.66±0.01	<b>37.25±0.02</b>	43.81±0.00
pFedGraph	4	39.70±0.02	48.58±0.09	32.99±0.03	39.66±0.02
pFedGraph	8	45.20±0.01	<b>55.30±0.01</b>	36.73±0.09	44.05±0.05
FedClassAvg	4	39.90±0.00	48.13±0.10	30.40±0.01	34.98±0.01
FedClassAvg	8	<b>45.22±0.02</b>	54.38±0.08	35.39±0.01	40.94±0.01
<b>FedTS(Ours)</b>	4	41.10±0.01	49.59±0.01	33.79±0.05	40.11±0.02
<b>FedTS(Ours)</b>	8	<b>45.61±0.01</b>	<b>55.72±0.00</b>	<b>37.32±0.02</b>	<b>44.80±0.01</b>

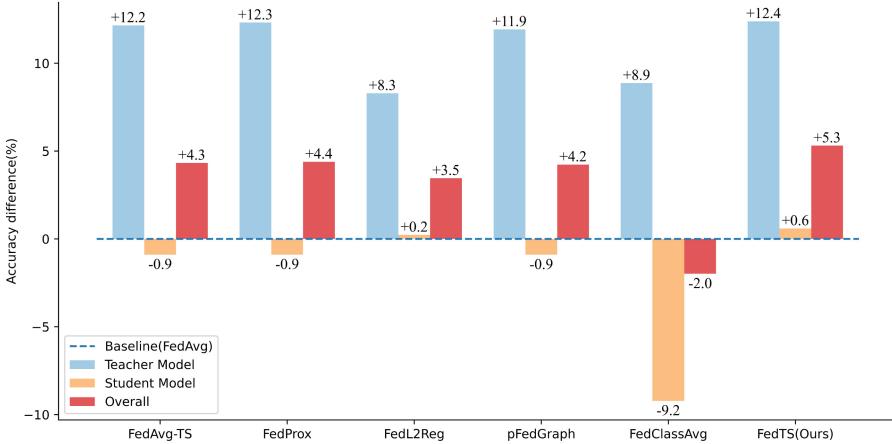
Our algorithm achieves the highest accuracy over all four datasets (the bolded values). Meanwhile, the second-highest accuracies(the blue-labeled values) are achieved by different algorithms over different datasets, which means other algorithms can't guarantee a stable result in various scenarios. In contrast, the proposed algorithm is more efficient and stable under different circumstances.

Our algorithm outperforms other algorithms due to effective feature alignment and knowledge distillation. While algorithms like pFedGraph offer effective head aggregation methods, they do not guide aligning features between teacher and student models. FedL2Reg, as demonstrated in our discussion on the visualization of latent representation, attempts to align features rigidly. Meanwhile, FedClassAvg does not consider global feature alignment, resulting in a lower accuracy when the model architecture differs significantly. Likewise, FedAvg and FedProx do not incorporate feature alignment mechanisms, so their performance is lower than ours.

Figure 3 displays the difference in performance between the models and the baseline(FedAvg) over four datasets( $|T| = 8, \beta = 0.5, 2.0$ ). If this value is greater than zero, it indicates an improvement compared to the baseline; conversely, if it is less than zero, it indicates a decrease. It shows that the proposed algorithm achieved the highest improvement in student, teacher and overall model performance. In contrast, while showing overall improvements, other algorithms may have achieved them at the expense of student model performance. Only a few algorithms have shown improvements in student model performance compared to the baseline, namely **FedL2Reg** and **FedTS(ours)**. However, the average

improvement in student models for FedL2Reg is only **+0.2**, which is lower than our results(**+0.6**).

In summary, in the presence of the teacher model, the performance of the student model surpasses the training results with only student models(FedAvg, the first row of the table). Meanwhile, the teacher model significantly enhances the average model performance of federated clients, which aligns with our expected goals.



**Fig. 3.** The difference in performance between the model and the baseline(FedAvg) over multiple datasets. The blue one refers to  $\bar{A}_{\text{teacher}} - \bar{A}$ , the orange one refers to  $\bar{A}_{\text{student}} - \bar{A}$  and the red one refers to  $\bar{A}$ . (Color figure online)

## 5 Conclusion

In the edge computing scenario, clients typically have imbalanced resources and non-iid data distributions. Moreover, a public dataset can be rare or expensive. Building upon this foundation, we propose the following hypothesis: resource-rich clients can run more complex models, while resource-constrained clients can use smaller models. As this approach introduces complexity and model heterogeneity, we propose the FedTS algorithm to tackle these issues. We introduce a generator to implement data-free knowledge distillation without a shared dataset. The generator takes label  $y$  as inputs and produces latent representation  $z$ , guiding the training of client models. Comprehensive experiments demonstrate that the proposed algorithm outperforms other algorithms on multiple datasets, such as CIFAR-10.

Future work can further relax the restrictions on model architecture in federated learning and explore how completely heterogeneous models can engage in mutual learning. Furthermore, it is also possible to explore federated learning in the context of model heterogeneity in the field of domain generalization.

## Appendix A Test Results on MNIST and Fashion-MNIST dataset

**Table 4.** Student and teacher model average accuracy comparisons on MNIST and Fashion-MNIST using 20 clients. The number of teachers being 0 means we use only student models.

Method	Teachers	Model	MNIST		Fashion-MNIST	
			Dir(0.5)	Dir(2)	Dir(0.5)	Dir(2)
FedAvg	0	Student	91.93±0.00	96.19±0.00	76.25±0.00	84.46±0.00
FedAvg-TS	4	Student	90.93±0.00	95.46±0.01	76.22±0.02	83.68±0.00
		Teacher	87.26±0.09	97.30±0.00	60.19±1.90	83.88±0.04
FedAvg-TS	8	Student	90.87±0.01	94.60±0.01	74.37±0.01	83.24±0.00
		Teacher	92.67±0.11	98.26±0.07	76.50±0.65	85.42±0.74
FedProx	4	Student	90.81±0.00	95.47±0.00	76.31±0.00	83.80±0.00
		Teacher	84.74±0.00	97.68±0.00	60.50±0.00	84.00±0.00
FedProx	8	Student	90.86±0.00	94.67±0.01	74.47±0.02	83.46±0.00
		Teacher	91.77±0.00	98.27±0.01	76.90±0.10	85.17±0.00
FedL2Reg	4	Student	96.54±0.01	85.79±0.01	76.71±0.00	83.96±0.00
		Teacher	81.06±0.01	96.36±0.01	59.40±0.00	81.32±0.00
FedL2Reg	8	Student	94.10±0.01	96.25±0.01	74.59±0.00	83.75±0.00
		Teacher	85.87±0.01	97.68±0.01	69.62±0.00	79.29±0.00
pFedGraph	4	Student	90.78±0.00	95.47±0.01	76.28±0.00	83.79±0.00
		Teacher	84.64±0.00	97.68±0.01	60.23±0.00	84.17±0.00
pFedGraph	8	Student	90.87±0.00	94.68±0.00	74.44±0.00	83.42±0.00
		Teacher	90.80±0.00	98.27±0.00	77.23±0.00	82.31±0.00
FedClassAvg	4	Student	73.82±0.01	85.79±0.01	52.36±0.18	71.10±0.05
		Teacher	81.69±0.01	92.40±0.01	60.20±0.00	82.39±0.00
FedClassAvg	8	Student	74.02±0.00	84.23±0.00	59.65±0.00	70.45±0.00
		Teacher	81.48±0.00	94.43±0.00	65.89±0.00	78.50±0.00
<b>FedTS(Ours)</b>	4	Student	94.42±0.00	96.75±0.00	77.50±0.00	85.07±0.00
		Teacher	82.32±0.00	97.86±0.00	63.79±0.00	84.15±0.00
<b>FedTS(Ours)</b>	8	Student	94.54±0.00	96.46±0.00	76.09±0.01	84.90±0.00
		Teacher	92.18±0.00	98.20±0.00	77.79±0.01	83.78±0.00

## Appendix B Visualization of latent representation

KPCA [27,34], also known as Kernel Principal Component Analysis, is a non-linear extension of the conventional Principal Component Analysis (PCA) methodology. It's a dimensionality reduction technique to analyze and extract

**Table 5.** Model average accuracy comparisons on MNIST and Fashion-MNIST using 20 clients. The number of teachers being 0 means we use only student models. We’ve bolded the highest accuracy and labeled the second-highest accuracy as blue.

Method	Teachers	MNIST		Fashion-MNIST	
		Dir(0.5)	Dir(2)	Dir(0.5)	Dir(2)
FedAvg	0	91.93±0.00	96.19±0.00	<b>76.25±0.00</b>	<b>84.46±0.00</b>
FedAvg-TS	4	90.20±0.02	95.83±0.00	73.02±0.37	83.72±0.01
FedAvg-TS	8	91.60±0.05	96.07±0.03	75.22±0.27	84.16±0.30
FedProx	4	89.59±0.00	95.91±0.00	73.15±0.00	83.84±0.00
FedProx	8	91.22±0.00	96.11±0.01	75.44±0.03	84.15±0.00
FedL2Reg	4	91.34±0.00	96.36±0.01	73.25±0.00	83.43±0.00
FedL2Reg	8	90.81±0.00	96.82±0.01	72.60±0.00	82.00±0.00
pFedGraph	4	89.55±0.00	95.86±0.01	73.07±0.00	83.87±0.00
pFedGraph	8	90.84±0.00	96.11±0.01	75.56±0.00	82.98±0.01
FedClassAvg	4	75.39±0.01	87.11±0.01	58.63±0.04	73.36±0.04
FedClassAvg	8	77.01±0.00	88.31±0.01	62.15±0.00	73.67±0.01
<b>FedTS(Ours)</b>	4	<b>92.00±0.01</b>	<b>96.97±0.00</b>	74.76±0.01	<b>84.89±0.00</b>
<b>FedTS(Ours)</b>	8	<b>93.60±0.00</b>	<b>97.16±0.00</b>	<b>76.77±0.01</b>	84.45±0.00

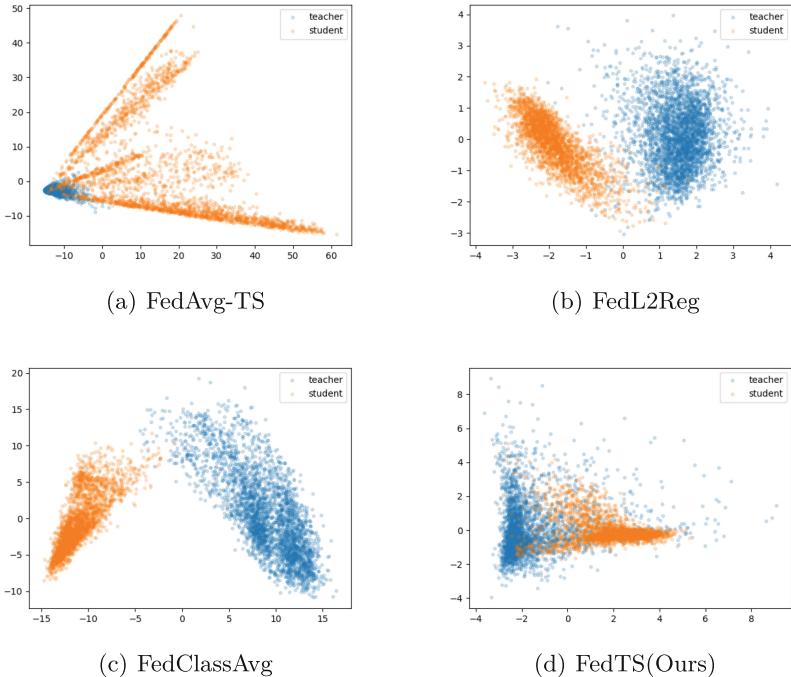
the most significant features from high-dimensional data. It proves to be an effective method for identifying abnormal or outlier data points that deviate from the general trend [9, 16, 30].

As depicted in Fig. 4, we employ KPCA to visualize the latent representation of the t-shirt class from the Fashion-MNIST dataset. The outcomes, arranged from left to right, are yielded by the **FedAvg-TS**, **FedL2Reg**, **FedClassAvg**, and **FedTS(Ours)** algorithms. Due to the absence of explicit guidelines for generating the latent representation in FedAvg-TS, its visualization appears disorganized and lacks alignment. Although FedL2Reg endeavors to align the generated latent representation towards the center using L2 regularization, significant structural disparities between the teacher and student models lead to substantial inconsistencies in the latent representation. Consequently, coercing them towards center alignment fails to produce the intended effect. The visualization results indicate that although the latent representations of both teacher and student models converge towards the center, they form distinct clusters and fail to achieve genuine alignment.

Likewise, FedClassAvg applies local feature representation learning using a supervised contrastive loss [7, 12], but it does not consider the scenario where model complexities differ significantly. Latent representation generated by heterogeneous models can vary greatly, so feature alignment is needed.

Conversely, our approach regards the teacher-generated latent representation as more effective and recognizes the significance of the student’s knowledge. We involve the teacher and student in the generator’s training but assign a lower

weight to the student. Guided by the generator, the student gradually assimilates the teacher’s feature extraction techniques, and we progressively increase its weight. Eventually, the generator assimilates comprehensive knowledge from all clients. As mentioned, we align each client’s posterior probabilities  $p_k(z|y)$  towards a joint distribution  $r(z|y)$ . The visualization results confirm our attainment of this objective: the latent representations of both the teacher and student models concentrate in the same region.

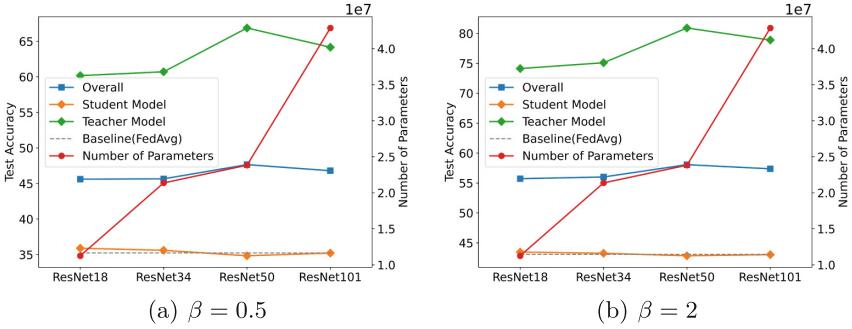


**Fig. 4.** The visualization of the latent representation of t-shirts generated by different algorithms on the Fashion-MNIST dataset. We use KPCA [27, 34] to reduce the representation tensor into 2-dimension. The orange points denote student-generated results, and the blue represent teacher-generated results.

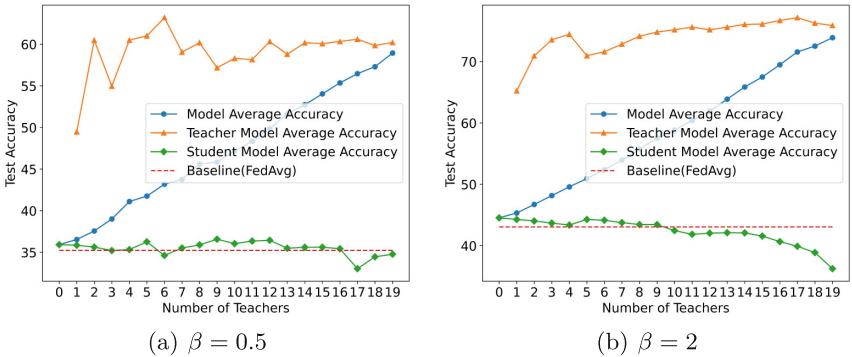
## Appendix C Tests on the Impact of Teacher Model Architecture

To examine the impact of teacher model architectures on model average performance, we conducted experiments using ResNet18, ResNet34, ResNet50 and ResNet101 as teacher models on the CIFAR-10 dataset with 8 teachers.

According to Fig. 5, enhancing the complexity of the teacher model results in a slight improvement of the overall model performance. However, it also leads to



**Fig. 5.** The number of model parameters and accuracy under different teacher model architectures. The red curve corresponds to the y-axis on the right, while the rest correspond to the left. (Color figure online)



**Fig. 6.** The model performance under different teacher percentages on CIFAR-10.

a decline in the average accuracy of the student model. For knowledge distillation, if the teacher model is too complex, the student may struggle to learn the knowledge provided by the teacher, leading to a decline in model performance [28]. Hence, there is a tradeoff between augmenting the overall performance and sustaining the performance of the student model. The most suitable teacher model architecture can be chosen based on specific requirements, while blindly increasing the complexity of the model may result in lower marginal benefits.

## Appendix D Tests on the Impact of Teacher Percentage

To examine the impact of teacher percentage on model average performance, we conducted experiments on the CIFAR-10 dataset with  $0, 1, \dots, 19$  teachers(20 clients in total). Figure 6 displays the experimental results, which show that our algorithm consistently yields stable results across multiple settings. As the teacher ratio increases, the model's average performance steadily improves.

Meanwhile, varying the number of teacher models has a relatively minor impact on the performance of student models, with most student models maintaining performance above the baseline except when the number of teachers reaches some threshold(17 when  $\beta = 0.5$ , 10 when  $\beta = 2.0$ ), which suggests that in practical applications, we can flexibly partition clients into teacher and student models based on specific criteria, achieving higher resource utilization rates. We can set an appropriate teacher ratio to ensure overall performance improvement while maintaining the performance of student models.

## References

1. AbdulRahman, S., Tout, H., Mourad, A., Talhi, C.: Fedmccls: multicriteria client selection model for optimal iot federated learning. *IEEE Internet Things J.* **8**(6), 4723–4735 (2020)
2. Cheng, S., Wu, J., Xiao, Y., Liu, Y.: Fedgems: federated learning of larger server models via selective knowledge fusion. arXiv preprint [arXiv:2110.11027](https://arxiv.org/abs/2110.11027) (2021)
3. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: Cinic-10 is not imangenet or cifar-10. arXiv preprint [arXiv:1810.03505](https://arxiv.org/abs/1810.03505) (2018)
4. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **29**(6), 141–142 (2012)
5. Duan, M., et al.: Astraea: self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In: 2019 IEEE 37th International Conference on Computer Design (ICCD), pp. 246–254. IEEE (2019)
6. Ghosh, T., et al.: A privacy-preserving federated-mobilenet for facial expression detection from images. In: International Conference on Applied Intelligence and Informatics, pp. 277–292. Springer, Heidelberg (2022). [https://doi.org/10.1007/978-3-031-24801-6\\_20](https://doi.org/10.1007/978-3-031-24801-6_20)
7. Gunel, B., Du, J., Conneau, A., Stoyanov, V.: Supervised contrastive learning for pre-trained language model fine-tuning. arXiv preprint [arXiv:2011.01403](https://arxiv.org/abs/2011.01403) (2020)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint [arXiv:1503.02531](https://arxiv.org/abs/1503.02531) (2015)
9. Hoffmann, H.: Kernel pca for novelty detection. *Pattern Recogn.* **40**(3), 863–874 (2007)
10. Hong, C.H., Varghese, B.: Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv. (CSUR)* **52**(5), 1–37 (2019)
11. Jang, J., Ha, H., Jung, D., Yoon, S.: Fedclassavg: local representation learning for personalized federated learning on heterogeneous neural networks. In: Proceedings of the 51st International Conference on Parallel Processing, pp. 1–10 (2022)
12. Khosla, P., et al.: Supervised contrastive learning. *Adv. Neural. Inf. Process. Syst.* **33**, 18661–18673 (2020)
13. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint [arXiv:1312.6114](https://arxiv.org/abs/1312.6114) (2013)
14. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
15. Li, D., Wang, J.: Fedmd: heterogenous federated learning via model distillation. arXiv preprint [arXiv:1910.03581](https://arxiv.org/abs/1910.03581) (2019)

16. Li, D., Wong, W.E., Wang, W., Yao, Y., Chau, M.: Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means. In: 2021 8th International Conference on Dependable Systems and Their Applications (DSA), pp. 551–559. IEEE (2021)
17. Li, G., Hu, Y., Zhang, M., Liu, J., Yin, Q., Peng, Y., Dou, D.: Fedhisyn: a hierarchical synchronous federated learning framework for resource and data heterogeneity. In: Proceedings of the 51st International Conference on Parallel Processing, pp. 1–11 (2022)
18. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. *Proc. Mach. Learn. Syst.* **2**, 429–450 (2020)
19. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of fedavg on non-iid data. arXiv preprint [arXiv:1907.02189](https://arxiv.org/abs/1907.02189) (2019)
20. Lin, T., Kong, L., Stich, S.U., Jaggi, M.: Ensemble distillation for robust model fusion in federated learning. *Adv. Neural. Inf. Process. Syst.* **33**, 2351–2363 (2020)
21. Liu, J., Huang, J., Zhou, Y., Li, X., Ji, S., Xiong, H., Dou, D.: From distributed machine learning to federated learning: a survey. *Knowl. Inf. Syst.* **64**(4), 885–917 (2022)
22. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: a survey. *IEEE Commun. Surv. Tutor.* **23**(4), 2131–2165 (2021)
23. Ma, X., Zhu, J., Lin, Z., Chen, S., Qin, Y.: A state-of-the-art survey on solving non-iid data in federated learning. *Futur. Gener. Comput. Syst.* **135**, 244–258 (2022)
24. Mahmud, M., Kaiser, M.S., McGinnity, T.M., Hussain, A.: Deep learning in mining biological data. *Cogn. Comput.* **13**(1), 1–33 (2021)
25. Makhija, D., Han, X., Ho, N., Ghosh, J.: Architecture agnostic federated learning for neural networks. In: International Conference on Machine Learning, pp. 14860–14870. PMLR (2022)
26. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
27. Mika, S., Schölkopf, B., Smola, A., Müller, K.R., Scholz, M., Rätsch, G.: Kernel pca and de-noising in feature spaces. *Adv. Neural Inf. Process. Syst.* **11** (1998)
28. Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H.: Improved knowledge distillation via teacher assistant. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34, pp. 5191–5198 (2020)
29. Nguyen, A.T., Torr, P., Lim, S.N.: Fedsr: a simple and effective domain generalization method for federated learning. *Adv. Neural. Inf. Process. Syst.* **35**, 38831–38843 (2022)
30. Nguyen, V.H., Golinvval, J.C.: Fault detection based on kernel principal component analysis. *Eng. Struct.* **32**(11), 3683–3691 (2010)
31. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019-2019 IEEE International Conference on Communications (ICC), pp. 1–7. IEEE (2019)
32. Pereyra, G., Tucker, G., Chorowski, J., Kaiser, L., Hinton, G.: Regularizing neural networks by penalizing confident output distributions. arXiv preprint [arXiv:1701.06548](https://arxiv.org/abs/1701.06548) (2017)
33. Sannara, E., Portet, F., Lalanda, P., German, V.: A federated learning aggregation algorithm for pervasive computing: evaluation and comparison. In: 2021 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp. 1–10. IEEE (2021)

34. Schölkopf, B., Smola, A., Müller, K.-R.: Kernel principal component analysis. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0020217>
35. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint [arXiv:1708.07747](https://arxiv.org/abs/1708.07747) (2017)
36. Xu, J., Tong, X., Huang, S.L.: Personalized federated learning with feature alignment and classifier collaboration. arXiv preprint [arXiv:2306.11867](https://arxiv.org/abs/2306.11867) (2023)
37. Ye, R., Ni, Z., Wu, F., Chen, S., Wang, Y.: Personalized federated learning with inferred collaboration graphs (2023)
38. Zhang, H., Li, C., Dai, W., Zou, J., Xiong, H.: Fedcr: personalized federated learning based on across-client common representation with conditional mutual information regularization (2023)
39. Zhang, J., Guo, S., Ma, X., Wang, H., Xu, W., Wu, F.: Parameterized knowledge transfer for personalized federated learning. Adv. Neural. Inf. Process. Syst. **34**, 10092–10104 (2021)
40. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint [arXiv:1806.00582](https://arxiv.org/abs/1806.00582) (2018)
41. Zhu, Z., Hong, J., Zhou, J.: Data-free knowledge distillation for heterogeneous federated learning. In: International Conference on Machine Learning, pp. 12878–12889. PMLR (2021)



# Physics-Informed Antisymmetric Recurrent Neural Networks for Solving Nonlinear Partial Differential Equations

Pavodi Maniamfu<sup>1</sup>(✉) , U. A. Md. Ehsan Ali<sup>1</sup> , Ko Sakai<sup>2</sup> , and Keisuke Kameyama<sup>2</sup>

<sup>1</sup> Degree Programs in Systems and Information Engineering and Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8577, Japan

{maniamfu,ali}@adapt.cs.tsukuba.ac.jp

<sup>2</sup> Institute of Systems and Information Engineering, University of Tsukuba, Tsukuba 305-8577, Japan

{sakai,keisuke}@cs.tsukuba.ac.jp

**Abstract.** Physics-informed neural network (PINN) is a new paradigm for solving partial differential equations (PDEs). It is a data-driven approach, where layered neural networks approximate the solution of the PDE of interest via regularization to satisfy the PDE conditions. The PINN employs multilayer perceptrons (MLPs) in conjunction with numerical methods, e.g. Runge-Kutta (RK) to model discrete-time PDEs. However, the present networks' structure exhibits constraints inherent in conventional numerical discretization approaches, particularly in handling iterative time-stepping processes. This is a critical challenge, which arises from an architectural standpoint. In addition, the PINN requires a dual-optimization approach to attain convergence, which can be computationally expensive. In this work, we propose physics-informed antisymmetric recurrent neural network (PIARNN), which encodes RK intermediate stages in the hidden state of the recurrent neural networks, thus enhancing the capabilities of the architecture of the PINN in learning complex nonlinear PDEs. We parameterize the recurrent connections with a strictly upper-triangular parameter vector, which achieves more robust and stable learning than the standard PINN. We conducted the experiments on Allen-Cahn and Burger's equations, where the proposed framework achieved superior prediction performance across various RK stages while maintaining fewer learnable parameters. It was observed that the PIARNN can achieve a better result in approximating the solution of the PDE with a single optimizer than the PINN.

**Keywords:** PINN · Recurrent Neural Network · Partial Differential Equation · Runge-Kutta method

## 1 Introduction

Multilayered neural networks approximate almost any function. They are a class of universal approximators [9]. This property together with their nonlinearity

composition allows them to tackle complex problems beyond the linear mappings in fields such as computer vision [16], natural language processing [25], and autonomous driving [8]. Their success has been attributed to large available data [24] and the advancement both in high-performance computers and software infrastructures for training sophisticated models and learning methods [1,17].

Recently, multilayer perceptrons (MLPs) have been utilized not only as an alternative but in tandem with numerical approaches as a new paradigm to solve partial differential equations (PDEs) [22]. Conventionally, the training process for neural network-based algorithms demanded a huge amount of labeled data [16]. However, in numerous scientific settings, the cost of data acquisition can be costly. The existence of prior knowledge in these settings offers a valuable resource for regularization, compensating for the scarcity of labeled data.

Physics-informed neural network (PINN) [22] is a novel approach, which utilizes a few samples of data to approximate the solution of the PDE via regularization that satisfies the PDE conditions. Despite its success in various applications such as gas dynamics, chemical kinetics [6, 13], it faces some challenges. The latter includes the complexity of the loss landscape's geometry, the noise generated by the physics loss and an expensive training cost.

In this paper, we propose to utilize a robust and stable Recurrent Neural Network (RNN) architecture for the PINN framework. This architecture extends from the antisymmetric RNN proposed in [5]. This approach is motivated by a dynamical system's perspective that extends the capabilities of the standard RNN to handle long-term dependencies in sequence learning. In contrast to the antisymmetric RNN which requires a dense square matrix to enforce the antisymmetric property to ensure optimal performance, the proposed approach employs a strictly upper triangular connection matrix to maintain the required antisymmetry. This makes PIARNNs more efficient in terms of the number of parameters, which results in a significantly smaller model size while maintaining superior performances. We focus on the discrete-time modeling, where PIARNNs incorporates the implicit Runge-Kutta (RK) time-stepping formalism [10] to construct a structured predictive algorithm.

The main contributions of this work are as follows:

- We integrate RNN with the RK time-stepping methods in PINN's context. This approach allows to enhance the stability and expressivity of PINNs in discrete-time modeling.
- The antisymmetric property provides two key advantages: first, it stabilizes the learning process of the PINN framework, resulting in a stable learning curve. Second, RNNs, constrained by an upper triangular connection matrix, have fewer parameters while maintaining better performance.
- We conduct a comparative analysis between the PINN and PIARNN frameworks in terms of the model size and show that a smaller model can achieve a better solution precision.

## 2 Physics-Informed Neural Networks

Physics-informed neural networks (PINNs) represent a framework in three stages for modeling both continuous- and discrete-time PDEs. The first building block is a layered neural network for facilitating the learning of the solution to the PDE. The second component is the physics-informed zone, where the residuals of the PDE are computed using the input coordinates and model parameters via automatic differentiation. The last building component comprises the discrete-time model, wherein the solution to the PDE is approximated through a numerical time-stepping scheme, with RK methods being a prime example.

Suppose the PDE that governs a spatio-temporal dynamical system of  $u(x, t)$  is known as follows,

$$f(x, t) := u_t + N[u] = 0, \quad x \in \Omega, \quad t \in [0, T] \quad (1)$$

where  $u_t := \frac{\partial u}{\partial t}$  denotes the partial derivative of  $u$  w.r.t time. Here, function  $N$  is a nonlinear differential operator. Set  $\Omega$  is a subset of  $\mathbb{R}^D$ , and represents the spatial domain. Interval  $[0, T]$  denotes the time span that confines the period over which the solution is formulated. Given some initial and boundary conditions, the problem can be solved by utilizing a data-centric approach with a relatively limited amount of data. This formalism is known as physics-informed neural networks (PINNs) [22], where the PDE residuals and their boundary conditions are incorporated into the loss function as a soft constraint regularization. This novel approach allows the integration of prior scientific knowledge about  $u(x, t)$ , in the form of differential equations, into the learning process of neural networks.

The neural network in Fig. 1a takes a spatial scalar  $x$  as an input at a fixed time  $t$ . This scalar feature undergoes a transformation through the network to output a  $q + 1$ -dimensional feature vector. These outputs correspond to the Runge-Kutta stages  $u^{n+c_i}(x)$ , for  $i = 1, \dots, q$ . They are generated in parallel due to the nature of the MLP's architecture. The network's predictions are used to compute the derivatives of the nonlinear operator  $N[.]$  w.r.t the input coordinate.

The solution is learned using the sum of squared errors that minimizes the discrepancy between the governing equation and the neural network's approximations. It is defined as:

$$SSE = SSE_n + SSE_b \quad (2)$$

where  $SSE_n$  denotes the residual error as:

$$SSE_n = \sum_{j=1}^{q+1} \sum_{i=1}^{N_n} |u_j^n(x^{n,i}) - u^{n,i}|^2, \quad (3)$$

and  $SSE_b$  enforces the boundary conditions of the PDE of interest.

In discrete-time modeling, the PINN is used with Runge-Kutta (RK) time-stepping schemes. The latter is a family of iterative numerical approaches used to solve nonlinear differential equations by discretizing their temporal dynamics [10]. RK methods encapsulate both implicit and explicit time-stepping schemes

including the forward Euler method, which is seen as the first-order numerical procedure in the RK schemes. RK methods can be used in synergy with neural networks to improve the accuracy and stability of numerical approaches.

Applying RK methods to the PDE in Eq. (1), results in this discrete form:

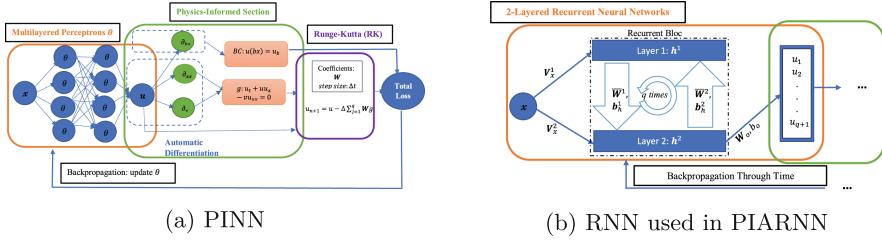
$$\begin{aligned} u_i^n &= u^{n+c_i} + \Delta t \sum_{j=1}^q a_{ij} N[u^{n+c_j}], \quad i = 1, \dots, q, \\ u_{q+1}^n &= u^{n+1} + \Delta t \sum_{j=1}^q b_j N[u^{n+c_j}], \end{aligned} \quad (4)$$

where  $u^{n+c_j}(x) = u(t^n + c_j \Delta t, x)$  for  $j = 1, \dots, q$  and the superscript  $n + c_j$  denotes the intermediate stages used to calculate the solution at  $u^{n+1}$ .  $a_{ij}, b_j, c_j$  are the parameters of the RK methods, which are determined together with the exact solution. The training and test data-set are generated using conventional spectral methods. Given initial conditions and the periodic boundary of the PDE in Eq. (1), the equation can be integrated up to a final time  $T$  using a fourth-order Runge-Kutta scheme [22].

The idea of the PINN is to constrain the solution to abide by the physical laws of the system. In this way, the solution of the PDE is learned by the neural network, often implemented by leveraging the power of automatic differentiation [4]. The model can also be applied to high-dimensional nonlinear PDEs inherent in various physical phenomena at inference.

In contrast to traditional numerical approaches, PINNs, as highlighted in [11], enable a mesh-free algorithm as the automatic differentiation guarantees the approximation of the differential operator w.r.t to the network's parameters. This is considered to be a crucial milestone in numerical analysis. Additionally, when addressing complex nonlinear PDEs, physics-based numerical techniques often lead to computational demands. Under these scenarios, PINNs emerge as an efficient data-driven approach.

Despite the successful application of the PINN, it encounters critical challenges. First, the geometry of the loss landscape can be very hard and complex to optimize with a first-order optimization approach, such as Adam [14]. Consequently, this complexity requires implementing a dual-optimization approach, wherein a second-order optimizer, such as L-BFGS [18], is used for a high-precision solution. Second, backpropagated gradients from the MLPs – the backbone architecture of the PINN – can become corrupted by the differential operator [2], which results in a very unstable learning curve. During the learning process, the MLPs generate early-stage predictions that are noisy due to the random initialization [7]. This noise impacts the accuracy of the differential operator's derivatives, leading to a noisy physics loss, which in turn results in erroneous backpropagated gradients. Lastly, the PINN framework with many trained parameters requires a large training cost to attain convergence, which can be computationally burdensome.



**Fig. 1.** Comparative Architectures. MLP in (a) is replaced by the RNN in (b)

### 3 Related Works

Several works analyze the appropriate architecture for PINNs. [12] introduces conservative PINNs (cPINNs) to extend PINNs capability through a discrete-domain decomposition for conservation laws. Later, it is further developed to extended PINNs (XPINNs) [11], which employs multiple MLPs in subdivided space-time domains, tailored to the PDE's complexity. Similarly, [20] proposed a parareal physics-informed neural network (PPINN) to accelerate the convergence of the PINN. In line with our framework, physics-informed attention-based neural networks (PIANNs) [23] combines recurrent neural networks and attention mechanisms to address the network architecture suited for learning complex PDE behaviors, particularly for solving the Buckley-Leverett problem.

In [19], the authors introduced backward-compatible PINNs (bc-PINNs), which adopts a sequential learning strategy across time segments. They used transfer learning, which retains features from earlier segments to facilitate the training of subsequent temporal segments. As highlighted in [2], this approach can be effective in diminishing the noise in the physics loss, resulting in a stable model.

Some lines of work focused on mitigating PINN failures. In [15], the authors studied the PINN loss landscape for complex PDEs. There, they proposed a regularization technique, where the PINN learns from simple to complex equations progressively. In addition, rather than learning the entire space-time at once, they reframed the problem as a sequence-to-sequence learning task, resulting in reduced errors. A critical investigation of failure modes is also proposed in [3], where a simple residual neural network is used to speed up training and smoothen the loss landscapes for easier optimization.

Most approaches rely on the MLP approximator, although effective, it can face challenges in managing the iterative aspects of numerical approaches when used in discrete-time modeling. In addition, the backpropagated gradients from the MLP lead to an unstable learning, which requires transfer learning as a correction measure. In the following section, we propose a novel approach to achieve a stable learning more efficiently.

## 4 Physics-Informed Antisymmetric Recurrent Neural Networks

We combine a recurrent neural network (RNN), which relies on the stability of an antisymmetric property, and RK methods to improve the stability and expressivity of the network's architecture used to train physics-informed models.

### 4.1 Recurrent Neural Networks

A Recurrent neural network (RNN) is a particular type of neural network structured by an internal hidden memory  $\mathbf{h} \in \mathbb{R}^N$ , with input parameters  $\mathbf{V}_x \in \mathbb{R}^{N \times M}$ , recurrent parameters  $\mathbf{W}_h \in \mathbb{R}^{N \times N}$ , recurrent bias  $\mathbf{b}_h \in \mathbb{R}^N$ , output parameters  $\mathbf{W}_y \in \mathbb{R}^{q \times N}$ , and output bias  $\mathbf{b}_y \in \mathbb{R}^q$  where  $M$  is the size of the input sequence,  $N$  denotes the number of hidden units, and  $q$  is the size of the output. This structure makes RNN capable of learning dynamical patterns that require temporal development. Given a sequence of input  $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_S)$ , where  $\mathbf{x}_i \in \mathbb{R}^M$ , RNN learns a sequence of output representation  $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_S)$ , where each element  $\mathbf{y}_i \in \mathbb{R}^q$  is computed recursively as:

$$\begin{aligned}\mathbf{h}_s &= f(\mathbf{V}_x \mathbf{x}_s + \mathbf{W}_h \mathbf{h}_{s-1} + \mathbf{b}_h) \\ \mathbf{y}_s &= \mathbf{W}_y \mathbf{h}_s + \mathbf{b}_y,\end{aligned}\tag{5}$$

where  $\mathbf{h}_s$  is the hidden state at time  $s$ , and  $f$  is an activation function. This form of RNN has been extensively developed to cope with sequence-learning tasks such as machine translation [25], and various time-sequential problems.

### 4.2 Antisymmetric RNN

The RNN mentioned above has an unstructured representation of the hidden state. This limitation manifests as a significant instability, which is further amplified by the recursive nature of the network architecture. The residual RNN [26], defined with skip connections as follows:

$$\mathbf{h}_s = \mathbf{h}_{s-1} + f(\mathbf{V}_x \mathbf{x}_s + \mathbf{W}_h \mathbf{h}_{s-1} + \mathbf{b}_h),\tag{6}$$

extends RNN's ability to maintain stability across long sequence learning. This approach, together with the antisymmetric RNN proposed in [5] can be one of the many efficient ways to stabilize the learning of the RNN, thus solving the instability of the PINN framework.

Given the following RNN hidden state:

$$\mathbf{h}_s = \mathbf{h}_{s-1} + \epsilon f(\mathbf{V}_x \mathbf{x}_s + (\mathbf{W}_h - \mathbf{W}_h^\top - \gamma \mathbf{I}) \mathbf{h}_{s-1})\tag{7}$$

where  $\mathbf{h}_s$  is the current hidden state while  $\mathbf{h}_{s-1}$  is the previous hidden state.  $\mathbf{I}$  is the identity matrix.  $\mathbf{W}_h \in \mathbb{R}^{N \times N}$  is the recurrent parameter,  $\mathbf{V}_x \in \mathbb{R}^{N \times M}$  is the input parameter and  $\mathbf{b}_h \in \mathbb{R}^N$  is the bias term. Scalars  $\epsilon$  and  $\gamma$  are hyperparameters and can be tuned freely.

The antisymmetric property is maintained by the recurrent parameter. A matrix  $\mathbf{W}$  is antisymmetric if  $\mathbf{W} \in \mathbb{R}^{N \times N}$  and satisfies  $\mathbf{W}^\top = -\mathbf{W}$ . It has properties such as the eigenvalues of  $\mathbf{W}$  are all imaginary  $\text{Re}(\lambda_i(\mathbf{W})) = 0$ , which maintains the recurrent block stable [5].

**Upper Triangular Antisymmetric RNN.** However, if the above antisymmetric recurrent matrix is a dense square matrix,  $\mathbf{W}_h \in \mathbb{R}^{N \times N}$ , the computation can be demanding. In this work, we parameterize the recurrent weight  $\mathbf{W}_h$  with a strictly upper triangular matrix where all entries on the diagonal are zeros. This results in a more efficient antisymmetric RNN with  $\frac{n(n-1)}{2}$  degree of freedom. The number of learnable parameters required to maintain the antisymmetric property is significantly lower compared to a full square matrix, which requires  $n^2$  computation to achieve the same result. During training, a matrix mask  $\mathbf{A}$  of the same size with  $\mathbf{W}_h$  is defined as follows,

$$A_{ij} = \begin{cases} 1 & \text{if } i < j \\ 0 & \text{otherwise} \end{cases}$$

and applied to the parameter gradients as,  $\mathbf{W}_h = \mathbf{A} \odot \nabla \mathbf{W}_h$ , where  $\odot$  denotes the Hadamard product, to maintain the upper triangular property.

### 4.3 RNN with Runge-Kutta Methods

Given the recursive nature of the RK methods and RNN in Eq. (7), we proceed to implement a 2-layered RNN as follows:

$$\begin{aligned} \mathbf{h}_s^{(1)} &= \mathbf{h}_{s-1}^{(2)} + \epsilon f \left( \mathbf{V}_x^{(1)} \mathbf{x}_s + \overline{\mathbf{W}}^{(1)} \mathbf{h}_{s-1}^{(2)} + \mathbf{b}_h^{(1)} \right) \\ \mathbf{h}_s^{(2)} &= \mathbf{h}_s^{(1)} + \epsilon f \left( \mathbf{V}_x^{(2)} \mathbf{x}_s + \overline{\mathbf{W}}^{(2)} \mathbf{h}_s^{(1)} + \mathbf{b}_h^{(2)} \right), \end{aligned} \quad (8)$$

with an output layer  $\mathbf{y}_s$ , defined by a fully-connected layer as,  $\mathbf{W}_y \mathbf{h}_s^{(2)} + \mathbf{b}_y$ , where  $\mathbf{h}_s^{(1)}$  represents the hidden state in the first layer, and  $\mathbf{h}_s^{(2)}$  the hidden state in the second layer. The underscript  $s$  ranges from 0 to  $q$ , determining the number of iterations in the recurrent block, which is followed by the calculation of the output  $\mathbf{y}_s$ . In Eq. (8) above,  $\overline{\mathbf{W}}^{(1)}$  and  $\overline{\mathbf{W}}^{(2)}$  are defined as:

$$\overline{\mathbf{W}}^{(1)} = \mathbf{W}_h^{(1)} - \mathbf{W}_h^{(1)\top} - \gamma \mathbf{I}, \quad \overline{\mathbf{W}}^{(2)} = \mathbf{W}_h^{(2)} - \mathbf{W}_h^{(2)\top} - \gamma \mathbf{I}. \quad (9)$$

In the physics-informed standard structure represented in Fig. 1a, the RNN in Fig. 1b replaces the multilayer perceptrons, while maintaining the rest of the structure. The goal of the RNN is to process recursively the intermediate stages of the RK methods, a feature which is not found in feedforward MLP(s).

#### 4.4 Training of the RNN

The RNN receives a spatial coordinate  $x$  as a scalar at a particular time  $t$ . This single-dimensional feature, including the initial hidden state  $\mathbf{h}_{s-1}^{(2)}$  are fed to the first layer to obtain  $\mathbf{h}_s^{(1)}$ , which is fed, together with the input feature, to the second layer to get the final hidden state  $\mathbf{h}_s^{(2)}$ . This procedure is processed iteratively  $q$  number of times according to the RK order predefined.

Unlike the MLP, the RNN computes sequentially the intermediate values  $q$  times, where the current hidden state's estimation depends on the previous hidden state. This is a fundamental aspect of the RK methods, where each  $q$  stage depends on the previous computation. The RNN will output the final hidden state  $\mathbf{h}_q^{(2)}$  as a result of this interdependency, which is forwarded to the subsequent fully-connected layer. The final obtained estimations allow to compute the derivatives of the PDE of interest, where the solutions are combined with the RK coefficients to generate new estimations.

Finally,  $SSE_n$  and  $SSE_b$  in Eq. 2 are used to minimize the difference between these estimations and the PDE exact solution. The error is backpropagated through time to update RNN parameters through an optimizer, such as Adam. In this way, the RNN is trained for a number of epochs to learn the solution. During inference, a scalar coordinate  $x$  from a different fixed time  $t$  is provided to the trained network to evaluate the model's ability to approximate the solution.

### 5 Experiments

We have selected two different partial differential equations to evaluate the performance of the proposed model.

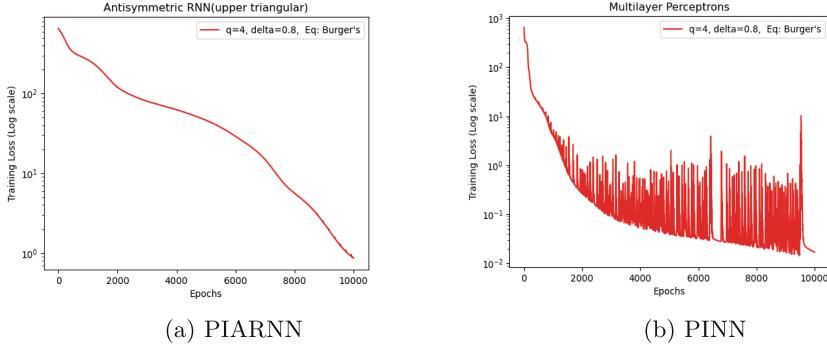
#### 5.1 Allen-Cahn Equation

**Implementation.** The Allen-Cahn equation is a fundamental equation used in various fields, particularly in physics, to describe phase separation processes. It is a family of reaction-diffusion equations, particularly known for its application in describing phase separation in multi-component alloy systems, including order-disorder transitions. It plays a crucial role in studying phenomena like crystal growth, image segmentation and the motion by mean curvature flows.

Given the following nonlinear Allen-Cahn equation along with periodic boundary conditions:

$$\begin{aligned} u_t - 0.0001u_{xx} + 5u^3 - 5u, \quad x \in [-1, 1], \quad t \in [0, 1], \\ u(0, x) = x^2 \cos(\pi x), \quad u(t, -1) = u(t, 1), \quad u_x(t, -1) = u_x(t, 1), \end{aligned} \tag{10}$$

where  $u_t$  represents the partial derivative of  $u$  w.r.t time  $t$ , indicating how  $u$  evolves over time. The second term involves the second partial derivative of  $u$  w.r.t the space  $x$ , scaled by a small coefficient. The nonlinear terms are associated with the reaction kinetics of the system, which can lead to complex dynamics.



**Fig. 2.** Examples of the training Loss measured in log scale.

The given initial condition specifies the state of  $u$  at time  $t = 0$  across the spatial domain  $x \in [-1, 1]$ . The periodic boundary condition  $u(t, -1) = u(t, 1)$  and  $u_x(t, -1) = u_x(t, 1)$  describes the behavior of the system at the boundaries  $x = -1$  and  $x = 1$ , creating a “loop” of behavior that results in a cyclic or periodic structure.

If we apply the classical RK methods with an arbitrary number of  $q$  stages to the nonlinear operator  $N[.]$  in Eq. (10), we obtain the following discrete form:

$$N[u^{n+c_j}] = -0.0001u_{xx}^{n+c_j} + 5(u^{n+c_j})^3 - 5u^{n+c_j}. \quad (11)$$

**Evaluation Measures.** The training set and test set are used as in PINNs. The training set consists of 200 initial points that are randomly sampled from the exact solution at time  $t = 0.1$ , shown in Fig. 3a. Similar to the PINN, the PIARNN’s framework is trained to learn  $u_{q+1}, q = 1, 2, \dots, q+1$  for predicting the solution  $u_t$  of the given PDE over the entire spatio-temporal domain at time  $t = 0.9$ . The loss function of the PIARNN has a similar structure as the PINN’s loss function described in Eq. 2, with the following specific boundary loss:

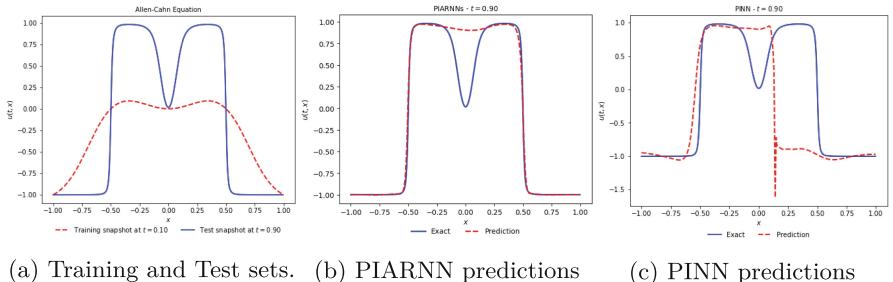
$$\begin{aligned} SSE_b = & \sum_{i=1}^q [u^{n+c_i}(-1) - u^{n+c_i}(1)]^2 + [u^{n+1}(-1) - u^{n+1}(1)]^2 \\ & + \sum_{i=1}^q [u_x^{n+c_i}(-1) - u_x^{n+c_i}(1)]^2 + [u_x^{n+1}(-1) - u_x^{n+1}(1)]^2 \end{aligned} \quad (12)$$

Here,  $SSE_b$  enforces the boundary conditions in the loss function.

**Conditions.** The proposed model and the standard PINN are trained with a single optimizer, Adam, at a learning rate of 0.001 for 10000 epochs. Meanwhile, the PIARNN’s training is also carried out with learning rate scheduling (LRS), utilizing a step-based reduction strategy,  $\eta = \eta \times \delta^{\lfloor \frac{p}{P} \rfloor}$ , for LRS, where the initial

learning rate  $\eta$  is decreased by a factor of  $\delta$  every  $\bar{p}$  epochs, and  $p$  denotes the current epoch. The values of  $\eta$ ,  $\delta$ , and  $\bar{p}$  are 0.1, 0.1, 3500, respectively. The PIARNN's architecture consists of 2 layers of 64 units while the PINN maintains the original structure in [22] with 4 hidden layers, each with 200 neurons. Hyperbolic tangent is used as the activation function for both architectures.

**Results.** In Table 1, we present a comparison of three conditions, all utilizing a single optimizer, Adam. Remarkably, the PIARNN model outperforms the standard PINN across various RK stages. From  $q = 2$  to  $q = 100$  RK time steps, it improves incrementally the performance as the number of  $q$  increases. Interestingly, the degree of freedom of the network architecture, shown in Table 4, is only approximately 4.6k because of the upper triangular antisymmetric constraint. This does not amount to any loss in finding a superior solution in the parameter search, but rather it achieves a stable learning. The mean performance of the PIARNN from  $q = 2$  to  $q = 100$  measured in  $L_2$  norm error is 3.2e-01 compared to the mean performance of the PINN, which is 1.0e+00.



**Fig. 3.** Allen-Cahn Equation with RK  $q = 16$ . Adam optimizer. Training and test snapshots are sampled at  $t = 0.10$ ,  $t = 0.9$ , respectively

**Table 1.** Allen-Cahn: Adam optimizer

RK $q$	$L_2$ Norm Error		
	PIARNNs (LRS)	PIARNNs	PINNs
2	<b>2.4e-01</b>	3.5e-01	1.0e+00
4	<b>2.5e-01</b>	6.6e-01	7.3e-01
8	<b>2.5e-01</b>	2.5e-01	1.3e+00
16	<b>2.3e-01</b>	2.3e-01	9.2e-01
32	3.0e-01	<b>2.6e-01</b>	1.3e+00
64	<b>1.3e-01</b>	2.6e-01	2.8e-01
100	9.3e-01	<b>2.9e-01</b>	1.2e+00
<b>Mean Error</b>	3.3e-01	<b>3.2e-01</b>	1.0e+00

**Table 2.** Allen-Cahn: Adam and L-BFGS optimizers

RK $q$	$L_2$ Norm Error		
	PIARNNs (LRS)	PIARNNs	PINNs
2	<b>2.2e-01</b>	3.9e-01	1.3e+00
4	6.0e-02	<b>1.0e-01</b>	1.6e-02
8	<b>8.2e-03</b>	1.9e-02	5.7e-02
16	2.2e-02	<b>1.4e-02</b>	5.6e-02
32	<b>1.1e-02</b>	6.0e-03	2.3e-01
64	1.6e-01	6.4e-03	<b>5.2e-03</b>
100	4.2e-02	<b>4.7e-03</b>	6.99e-03
<b>Mean Error</b>	<b>7.4e-02</b>	7.7e-02	2.3e-01

**Table 3.** Single optimizer: Adam with fixed  $q = 4$ 

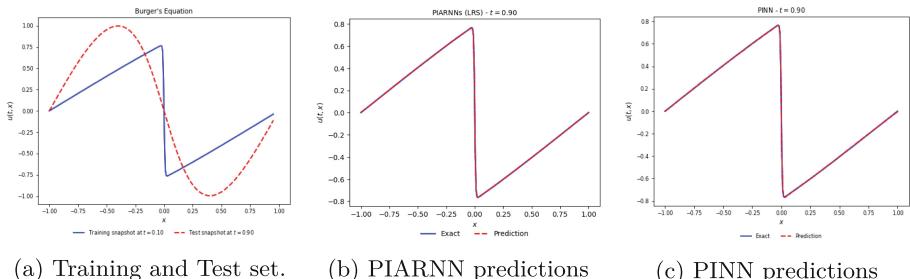
Hidden layer size (units)	Number of Params	$L_2$ error
32	1.2k	7.9e-01
64	4.6k	2.5e-01
128	17.4k	3.2e-01
256	67.5k	2.9e-01

**Table 4.** Comparison of Parameter Counts and  $L_2$  Norm Errors for MLPs and RNNs Models. Adam optimizer.  $q = 4$ 

Equations	Number of parameters		$L_2$ Norm Error	
	RNN	MLP	RNN	MLP
Allen Cahn	$\approx 4.6k$ (units = 64)	$\approx 122k$	2.5e-01	7.3e-01
Burger's	$\approx 3k$ (units = 50)	$\approx 8k$	5.7e-02	3.4e-02

Unlike the PINN, the PIARNN trained with a single optimizer shows a robust and stable performance despite the number of RK time steps. In Fig. 3b, the PIARNN's predictions closely follow the exact solutions better than the PINN in Fig. 3c. PINN's predictions diverge more significantly from the exact solution, especially around the jumps, where it does not capture the sharp transitions. However, the PIARNN appears to handle these regions slightly better.

As for the dual-optimization technique, where Adam and L-BFGS are used, the results are reported in Table 2. From  $q = 2$  to  $q = 100$ , PIARNNs achieve state-of-the-art performance with a mean error of 7.7e-02 compared to 2.3e-01 achieved by PINNs. The PIARNN requires only a few parameters to attain optimal results. Table 3 shows that increasing the hidden state size beyond 128 does not significantly improve performance.

**Fig. 4.** Burger's Equation with RK  $q = 100$ . Adam optimizer. Training and test snapshots are sampled at  $t = 0.10$ ,  $t = 0.9$ , respectively.

## 5.2 Burger's Equation

**Implementation.** Burger's equation is a fundamental partial differential equation used in the field of fluid dynamics to model shock waves and nonlinear wave propagation. In one-dimensional space, Burger's equation along with Dirichlet boundary conditions is given as:

$$\begin{aligned} u_t + uu_x - (0.01/\pi)u_{xx} &= 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ u(0, x) &= -\sin(\pi x), \quad u(t, -1) = u(t, 1) = 0 \end{aligned} \quad (13)$$

where  $u_t$  denotes the partial derivative of  $u$  w.r.t  $t$ . The nonlinear term  $uu_x$  represents the advection term while  $u_{xx}$  accounts for the diffusion effects along with the viscosity coefficient  $0.01/\pi$ . Applying the implicit RK time-stepping scheme in Eq. (4) to the nonlinear operator of Eq. (13) yields:

$$N[u^{n+c_j}] = u^{n+c_j}u_x^{n+c_j} - (0.01/\pi)u_{xx}^{n+c_j}. \quad (14)$$

**Evaluation Measures.** The training set and the test set are used as generated in the PINNs, and illustrated in Fig. 4a. The training points consist of 250 points sampled from the exact solution at  $t = 0.10$ , similar to the Allen-Cahn Equation. The PIARNN is trained to predict the solution further away at  $t = 0.9$ . The RNN is trained to minimize the loss in Eq. (2) with the following boundary loss:

$$SSE_b = \sum_{i=1}^q (|u^{n+c_i}(-1)|^2 + |u^{n+c_i}(1)|^2) + (|u^{n+1}(-1)|^2 + |u^{n+1}(1)|^2) \quad (15)$$

where -1 and 1 are the spatial boundaries enforced in the loss function. A Runge-Kutta time-stepping scheme is applied to the PIARNN's estimations. 250 training data points are sampled from the true solution at  $t = 0.1$ . The PIARNN is trained to predict the solution at  $t = 0.9$  with a time step size  $\Delta t = 0.8$ .

**Conditions.** As for the Burger's experiments, the PIARNN trained with learning rate scheduling achieves (LRS) competitive results. The PIARNN maintains the same architecture and conditions used in the Allen-Cahn experiments while PINN's original architecture consists of 4 hidden layers, each with 50 neurons. We used gradient norm clipping [21], particularly for  $q$  higher than 16.

**Results.** The performance of the model using Adam optimizer is reported in Table 5. Both, the PIARNN and the PINN, are trained for various RK stages, from  $q = 2$  to  $q = 100$ . The mean performance of the PIARNN with learning rate scheduling measured in  $L_2$  norm is  $6.2e-02$  compared to  $2.3e-02$  achieved by the PINN. Regardless of the number of implicit RK stages, our framework maintains stable performance with fewer learnable parameters to learn the solution.

As for the dual optimization approach, where Adam and L-BFGS are used, the result is reported in Table 6. From  $q = 2$  to  $q = 100$ , the PIARNN with learning rate scheduling achieves a mean performance of approximately  $3.3e-02$  compared to the PINN mean error,  $4.8e-02$ . Once again, our architecture shows a robust performance, while maintaining a small network size.

**Table 5.** Burger’s Equation optimizer: Adam

RK $q$	$L_2$ Norm Error		
	PIARNNs (LRS)	PIARNNs	PINNs
2	1.4e-01	1.1e-01	<b>1.0e-01</b>
4	2.6e-01	<b>3.1e-02</b>	3.4e-02
8	1.3e-02	1.2e-02	<b>8.0e-03</b>
16	6.5e-03	3.8e-02	<b>6.4e-03</b>
32	1.0e-02	4.8e-02	<b>5.6e-03</b>
64	<b>2.3e-03</b>	1.1e-01	4.4e-03
100	5.7e-03	1.0e-02	<b>5.0e-03</b>
Mean Error	6.2e-02	5.1e-02	<b>2.3e-02</b>

**Table 6.** Burger’s Equation optimizers: Adam and L-BFGS

RK $q$	$L_2$ Norm Error		
	PIARNNs (LRS)	PIARNNs	PINNs
2	1.6e-01	<b>1.4e-01</b>	2.2e-01
4	<b>2.1e-02</b>	3.8e-02	5.4e-02
8	<b>1.6e-02</b>	4.7e-02	5.8e-02
16	1.0e-02	2.5e-02	<b>1.1e-03</b>
32	2.0e-03	3.0e-02	<b>7.0e-04</b>
64	6.5e-03	2.7e-02	<b>7.8e-04</b>
100	1.7e-02	2.4e-02	<b>1.2e-03</b>
Mean Error	<b>3.3e-02</b>	4.7e-02	4.8e-02

## 6 Discussion

In this work, the PINN is revisited from an architectural perspective. Our proposed approach, physics-informed antisymmetric recurrent neural network (PIARNN), integrates the RNN with implicit RK schemes. It offers several benefits.

### 6.1 Quality Estimations

The standard PINN can produce poor results when trained with a single optimizer, such as Adam, in complex settings as in the Allen-Cahn equation. Moreover, the performance can drastically change based on the RK  $q$  stage. However, the PIARNN, not only achieves superior results than the PINN, but also maintains a stable performance across any RK  $q$  stage.

In the Burger’s situation, both models, the PINN and the PIARNN produced better results using a single optimizer, with the PINN having a slightly better mean performance. In the dual optimization technique, the PIARNN achieves a mean error that is substantially lower than the PINN across various RK  $q$  stages.

### 6.2 Compact Model

A dual-optimization technique is often required to attain convergence in PINN’s context. This approach can be time-consuming and resource demanding, especially when the architecture of the neural networks is complex. The PIARNN enforces the antisymmetric property in the recurrent connections of the RNN through the application of the upper triangular matrix, which results in a parameter-efficient model. The degree of freedom is much less, about 3.8%, shown in Table 4, of the parameters compared to the PINN for  $q = 4$ , resulting in significantly fewer parameters without any loss of the performance.

We found out that the antisymmetric property achieves a stable learning, which corrects the “corrupted” gradients, as shown in Fig. 2.

### 6.3 Limitations

The study [22] highlights that in discrete-time modeling, the performance largely depends on the number of RK stages  $q$  and the time-step size  $\Delta t$ . Higher  $q$  generally improves performance. Due to the limitation of the parameter search of the PIARNN, a second optimizer, such as L-BFGS, has limited effect compared to the PINN, particularly in Burger's case with  $q$  over 16. Although the reason is not obvious now, it could be a direction for the extension of this work. Furthermore, we plan to conduct a more comprehensive evaluation of the proposed approach with advanced PINN baseline models as part of our future work.

## 7 Conclusion

In this work, we revisited the framework of the PINN from an architectural standpoint. We proposed a novel approach based on the recurrent neural network (RNN) to solve partial differential equations. The proposed framework integrates the RNN with RK methods to process  $q$  intermediate stages more efficiently. Enforcing the antisymmetric property with a strictly upper-triangular connection matrix stabilized the learning process of the PINN model, and reduced the number of learnable parameters required to learn the solution of the PDE. In addition to the stability, the experiments conducted show that the PIARNN can achieve superior prediction performance using various RK  $q$  stages compared with the PINN.

**Acknowledgment.** This work was supported by JSPS Kakenhi grant number JP23H03697.

## References

1. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation. USENIX Association, Savannah (2016)
2. Basir, S.: Investigating and mitigating failure modes in physics-informed neural networks (PINNs). Commun. Comput. Phys. (2022)
3. Basir, S., Senocak, I.: Critical investigation of failure modes in physics-informed neural networks. In: AIAA SCITECH 2022 Forum, p. 2353 (2022)
4. Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. **18**(153), 1–43 (2018)
5. Chang, B., Chen, M., Haber, E., Chi, E.H.: AntisymmetricRNN: a dynamical system view on recurrent neural networks. In: Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA (2019)
6. De Florio, M., Schiassi, E., Ganapol, B., Furfarro, R.: Physics-informed neural networks for rarefied-gas dynamics: thermal creep flow in the bhatnagar-Gross-Krook approximation. Phys. Fluids **33**(4) (2021)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, vol. 9, pp. 249–256 (2010)

8. Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G.A.: A survey of deep learning techniques for autonomous driving. *J. Field Robot.* **37**, 362–386 (2020)
9. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**(5), 359–366 (1989)
10. Iserles, A.: *A First Course in the Numerical Analysis of Differential Equations*. Cambridge Texts in Applied Mathematics, 2nd edn. Cambridge University Press, Cambridge (2008)
11. Jagtap, A.D., Karniadakis, G.E.: Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition-based deep learning framework for nonlinear partial differential equations. *Commun. Comput. Phys.* **28**(5), 2002–2041 (2020)
12. Jagtap, A.D., Kharazmi, E., Karniadakis, G.E.: Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. *Comput. Methods Appl. Mech. Eng.* **365**, 113028 (2020)
13. Ji, W., Qiu, W., Shi, Z., Pan, S., Deng, S.: Stiff-PINN: physics-informed neural network for stiff chemical kinetics. *J. Phys. Chem. A* (2021). <https://doi.org/10.1021/acs.jpca.1c05102>
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference for Learning Representations, San Diego, CA, USA (2015)
15. Krishnapriyan, A.S., Gholami, A., Zhe, S., Kirby, R.M., Mahoney, M.W.: Characterizing possible failure modes in physics-informed neural networks. *Adv. Neural. Inf. Process. Syst.* **34**, 26548–26560 (2021)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. *Commun. ACM* **60**(6), 84–90 (2017)
17. Lew, J., et al.: Analyzing machine learning workloads using a detailed GPU simulator. In: IEEE International Symposium on Performance Analysis of Systems and Software, Madison, WI, USA (2019)
18. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**(1–3), 503–528 (1989)
19. Matthey, R., Ghosh, S.: A novel sequential method to train physics-informed neural networks for Allen Cahn and Cahn Hilliard equations. *Comput. Methods Appl. Mech. Eng.* **390**, 114474 (2022)
20. Meng, X., Li, Z., Zhang, D., Karniadakis, G.E.: PPINN: parareal physics-informed neural network for time-dependent PDEs. *Comput. Methods Appl. Mech. Eng.* **370**, 113250 (2020)
21. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training Recurrent Neural Networks. In: Proceedings of the 30th International Conference on Machine Learning, pp. 1310–1318 (2013)
22. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019)
23. Rodriguez-Torrado, R., et al.: Physics-informed attention-based neural network for hyperbolic partial differential equations: application to the Buckley-Leverett problem. *Sci. Rep.* **12**(1), 7557 (2022). <https://doi.org/10.1038/s41598-022-11058-2>
24. Sejnowski, T.J.: The unreasonable effectiveness of deep learning in artificial intelligence. *Proc. Natl. Acad. Sci. U.S.A.* **117**(48), 30033–30038 (2020). <https://doi.org/10.1073/pnas.1907373117>

25. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **27** (2014)
26. Yue, B., Fu, J., Liang, J.: Residual recurrent neural networks for learning sequential representations. *Information* **9**(3), 56 (2018)



# APS: An Adaptive Policy Switching Framework to Improve the Generalization of Branching Policy

Ce Zhang<sup>1,2</sup>, Dapeng Li<sup>1,2</sup>, Lin Lin<sup>1</sup>, Xinyue Lu<sup>1,2</sup>, and Guoliang Fan<sup>1(✉)</sup>

<sup>1</sup> Institute of Automation, Chinese Academy of Sciences, Beijing, China

{zhangce2023, lidapeng2020, lin.lin, luxinyue2023, guoliang.fan}@ia.ac.cn

<sup>2</sup> School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

**Abstract.** Mixed Integer Linear Programming (MILP) is a type of NP-hard problem that is widely used in industries but challenging to solve. Recently, there has been increasing interest in using Deep Learning (DL) to accelerate solving MILPs, particularly through learning to branch. However, the validity of DL models is based on the assumption that the test set and the training set are drawn from the same distribution, while in MILPs, the distribution shift often appears. To improve the generalization of the branching policy, we introduce the domain generalization method and propose an Adaptive Policy Switching (APS) framework. The experiments demonstrate that APS outperforms the previous state-of-the-art (SOTA) method, reducing the number of nodes by 18% on completely unseen MILPs.

**Keywords:** Mixed Integer Linear Programming · Learning to Branch · Domain Generalization · Adaptive Policy Switching

## 1 Introduction

Mixed Integer Linear Programming (MILP), a key branch of mathematical optimization, plays a significant role in fields such as transportation [6], scheduling [8], and planning [28]. The defining characteristic of MILP problems is that some variables must satisfy integer constraints, meaning their feasible solutions can only be integer values. While linear programming problems can be efficiently solved using methods such as the simplex [26] or interior point [29], adding integer constraints introduces significant non-convexity, making these problems considerably more challenging to solve.

To deal with the integer constraints, Branch-and-Bound (B&B) [21] is also commonly employed, which solves MILPs by branching them into smaller subproblems and using bound to prune away subproblems that cannot contain optimal solutions. Solving MILPs using B&B involves making several critical decisions that impact the size of the B&B tree. Two of the most fundamental

decisions are branching variable selection (BVS) and node selection. By reducing the size of the tree, the solving time can be decreased accordingly. This paper focuses specifically on the BVS aspect.

In previous works, some expertly-crafted heuristic rules are often used to make these decisions [1, 4]. However, some heuristic rules, such as strong branching, can be time-consuming, and others may not be applicable to all MILP instances. These heuristic rules generally do not account for the specific characteristics of the instances being solved.

Consequently, recent research increasingly focuses on using deep learning to replace heuristic rules [7, 32, 42]. Among these works, the study by Gasse et al. [11] is foundational, establishing MILP as a bipartite graph and processing it using Graph Convolutional Neural Networks (GCNN). However, applying deep learning to solve MILP and other combinatorial optimization problems poses a significant challenge in generalization. A network trained on one MILP instance may perform poorly on a heterogeneous instance. Although Gupta et al. [14] contend that this approach is quite reasonable, as practitioners typically focus on solving a specific problem repeatedly, addressing the generalization problem remains an important research theme. Improving the generalization can significantly expand the applicability of neural networks in MILP. Therefore, Liu et al. [25] employs adversarial instance augmentation to generate new instances and improve policy generalization. Zarpellon et al. [40] abandon the use of GCNN in favor of a TreeGate network, which is not dependent on specific MILP instances.

In this paper, building on the work of TreeGate policy [40], we further improve the generalization of the BVS policy via domain generalization (DG). Firstly, we observed that although all instances share a higher order in the same representational space, there is still some out-of-distribution (OOD) data in the test set, which the policy may not generalize to effectively. This scenario is very similar to the motivation of DG, which aims to train a model capable of generalizing to a previously unseen test (target) domains [38]. Therefore, we employ Meta-Learning based Adversarial Domain Augmentation (M-ADA) [30], a data generation-based DG method, to generalize the policy to the target domain. Our contributions can be summarized as follows:

- We identified the presence of OOD data in the target domain, which hinders the generalization of the BVS policy.
- We propose an Adaptive Policy Switching (APS) framework to improve the generalization of the BVS policy. Within this framework, we train both an original policy and a special policy for OOD data. Additionally, we develop a discriminator that helps us adaptively switch the BVS policy based on the states in the target domain.
- We evaluate the performance of the proposed method on the benchmarks established by [40]. Our method significantly outperforms pseudocost branching, a heuristic rule, and the GCNN method. Moreover, our method surpasses the SOTA TreeGate policy [40], reducing the number of nodes by 18%.

## 2 Related Work

A seminal work in BVS policy is [11], which models the MILP instance as a bipartite graph and employs GCNN to encode the branching policies. This study uses the results of strong branching as labels and applies imitation learning to train the BVS policy neural network. Many subsequent works have built on this neural network architecture or training framework, exploring improvements in areas such as reinforcement learning, BVS computation simplification and generalization.

### 2.1 Reinforcement Learning

In recent years, reinforcement learning has garnered growing attention [15, 34, 35], leading to the development of various subfields, including multi-agent reinforcement learning [22, 23, 31, 41], model-based reinforcement learning [16, 17], and offline reinforcement learning [10, 19], among others. Consequently, there has been a natural progression toward utilizing reinforcement learning to train BVS policies in some studies.

Etheve et al. [9] initially highlighted that the primary challenge in training BVS policy using reinforcement learning is the misalignment between the reward and the overarching goal of minimizing the total number of nodes in the B&B tree. It can be demonstrated that when the node selection method employed is Depth First Search (DFS), the reward and the global objective align. However, the application of DFS can also decelerate the solving process. Consequently, subsequent research has focused on overcoming the constraints imposed by DFS. Scavuzzo et al. [33] introduced a Tree Markov Decision Process (MDP) that aligns more closely with the B&B tree expansion process without relying on DFS. Based on the specific characteristics of B&B, Parsonson et al. [27] revised the data storage approach in the replay buffer, enabling a reasonable reward independent of DFS.

### 2.2 BVS Computation Simplification

The complex GCNN performs well on GPU hardware. However, in practical applications where computation is often limited to CPUs, GCNN does not demonstrate a speed advantage. To address this issue, Gupta et al. [14] proposed a hybrid approach that combines GCNN and Multi-Layer Perceptrons (MLP), employing GCNN only once at the branch delimiting root node before switching to MLP. Additionally, Kuang et al. [20] found through experimental analysis that BVS policy may not necessitate complex networks for representation. Instead, simple symbols can effectively represent this policy. Consequently, they employed a deep symbol discovery method to identify the BVS policy.

### 2.3 Generalization

To improve the generalization of BVS policy, Liu et al. [25] employ adversarial instance augmentation, generating diverse instances by masking variables,

constraints and edges in the instance bipartite graphs, and incorporating these instances into the training set. Zarpellon et al. [40] start from another perspective and abandon the bipartite graph structure. Their primary hypothesis suggests that parameterizing the state of the B&B search tree can enhance generalization. Under this state, they build new input features and a new neural network structure that can be applied to any MILP instance.

### 3 Preliminaries

#### 3.1 Branch-and-Bound Algorithm

The standard form of MILPs is as follows:

$$\begin{aligned} \min \quad & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n \\ & x_j \in \mathbb{Z}, \forall j \in \mathbf{I}, \end{aligned} \tag{1}$$

where  $A \in \mathbb{R}^{m \times n}$ ,  $c \in \mathbb{R}^n$ ,  $b \in \mathbb{R}^m$ ,  $\mathbf{I}$  is the set of subscripts of all integer variables, only these variables need to satisfy the integer constraints.

B&B algorithm is a fundamental algorithm framework for solving MILPs, comprising three main steps: branch, bound and prune. Consider a MILP problem  $\mathbf{M}$ , and let  $\mathbf{R}$  denote its relaxation obtained by ignoring the integer constraints. Define the optimal objective value as  $z^*$ , the lower bound of  $z^*$  as  $\underline{z}$  and the upper bound of  $z^*$  as  $\bar{z}$ . It follows that  $\underline{z} \leq z^* \leq \bar{z}$ .

The simplex method is employed to solve  $\mathbf{R}$ . In the optimal solution of  $\mathbf{R}$ , select a variable  $x_j = b_j$  that does not satisfy the integer constraint. During the branching process, add the constraints  $x_j \leq [b_j]$  and  $x_j \geq [b_j] + 1$  respectively to  $\mathbf{M}$ , thereby dividing  $\mathbf{M}$  into two subproblems:  $\mathbf{M}_1$  and  $\mathbf{M}_2$ . Here,  $[b_j]$  represents the largest integer not exceeding  $b_j$ .

In the bounding process, the optimal objective values of all the unbranching relaxation subproblems (i.e., all nodes at the end of the current tree) are compared, and  $\underline{z}$  is updated to the maximum objective value among them. For all optimal solutions of the subproblems that satisfy the corresponding integer constraints,  $\bar{z}$  is updated to the minimum objective value among these solutions. During pruning, a node is pruned if its optimal objective value exceeds  $\bar{z}$ , or if the relaxation problem cannot be solved.

Repeat the process of branching, bounding and pruning, until  $\bar{z} = \underline{z}$ , then stop the algorithm.

#### 3.2 Branching Rules

In the branching process of B&B described above Sect. 3.1, two decisions are involved: select a candidate variable to branch and select a node to branch. During the expansion of B&B tree, poor decisions can result in the creation of

nodes that do not contribute to finding the global optimal solution. Continuing to branch along these ineffective paths may only yield local optimal solutions that satisfy the integer constraints. As a result,  $\bar{z} \neq \underline{z}$  at these nodes, necessitating further iterations of the B&B algorithm.

At each node, the simplex method is employed to solve the relaxation subproblem, which incurs a computational cost. If variable selection and node selection are suboptimal, resulting in the branching of numerous ineffective nodes, significant time can be wasted, thereby substantially increasing the overall solving time for the MILP. Consequently, effective decision rules are essential for both variable selection and node selection to minimize the total number of nodes and reduce the solving time.

To make these two decisions, detailed heuristic rules can be found in [1, 4]. For variable selection, a score is computed for each candidate variable, and the variable with the highest score is selected for branching. The two commonly used scoring methods are linear scoring rule [24] and product scoring rule [2]. The linear scoring rule is as follows:

$$\text{score}(q^-, q^+) = (1 - \delta) \cdot \min\{q^-, q^+\} + \delta \cdot \max\{q^-, q^+\}, \quad (2)$$

where  $\delta$  is a hyperparameter,  $q^-$  and  $q^+$  are values associated with the two children of current node. The product scoring rule is as follows:

$$\text{score}(q^-, q^+) = \max\{q^-, \epsilon\} \cdot \max\{q^+, \epsilon\}, \quad (3)$$

with  $\epsilon = 10^{-6}$ .

Common rules for variable selection include pseudocost branching, strong branching, reliability branching, etc. Strong branching is a greedy rule that scores candidate variables based on which will yield the greatest improvement in the dual bound after branching. Pseudocost branching relies on historical data to guide current decisions. In SCIP [2], a leading open-source MILP solver, the default branching rule is reliable pseudocost (reldpscost) branching [3], which facilitates more “reliable” branching decisions early in the B&B tree.

Common rules for node selection include depth first search (DFS), best first search (BSF), among others. In this paper, we focus primarily on branching variable selection and do not delve into node selection. For the experiments conducted, we utilize the default node selection rule provided by SCIP.

### 3.3 Wasserstein Auto-Encoder

Wasserstein Auto-Encoder (WAE) [36] is an Auto-Encoder that employs Wasserstein distance as the measure of distribution difference. We define  $V_\phi$  as the WAE parameterized by  $\phi$ , which includes an encoder  $Q(\mathbf{e}|\mathbf{y})$  and a decoder  $G(\mathbf{y}|\mathbf{e})$ , where  $\mathbf{y}$  represents the input and  $\mathbf{e}$  represents the bottleneck embedding. When training  $V_\phi$ , we can adopt the training idea of GANs [13] and define a discriminator  $D_\gamma$  parameterized by  $\gamma$ . We assume a prior distribution  $P(\mathbf{e})$  for  $\mathbf{e}$  and

sample  $\{e_1, \dots, e_l\}$  from it. Simultaneously, we sample  $\{\tilde{e}_1, \dots, \tilde{e}_l\}$  from  $Q(\mathbf{e}|\mathbf{y})$ .  $D_\gamma$  can be updated by ascending:

$$L_D = \frac{\lambda}{l} \sum_{i=1}^l \log D_\gamma(e_i) + \log(1 - D_\gamma(\tilde{e}_i)), \quad (4)$$

where  $\lambda > 0$  is the regularization coefficient. This formulation means that the discriminator should classify data sampled from  $P(\mathbf{e})$  as the positive class and data sampled from the encoder  $Q(\mathbf{e}|\mathbf{y})$  as the negative class. And  $V_\phi$  can be updated by descending:

$$L_G = \frac{1}{l} \sum_{i=1}^l d(y_i, G(\tilde{e}_i)) - \lambda \cdot \log D_\gamma(\tilde{e}_i), \quad (5)$$

where  $\{y_1, \dots, y_l\}$  are sampled from the training set, and  $d(\cdot, \cdot)$  is a cost function that measures the discrepancy between the two inputs.

### 3.4 Meta-learning in Domain Generalization

Meta-learning, which can also be summarized as learning to learn, can be applied to domain generalization [30]. It mainly consists of three parts: meta-train, meta-test and meta-update. In meta-train, the loss function  $L$  of the model parameterized by  $\theta$  is computed using data from the source domain  $S$ :

$$\hat{\theta} \leftarrow \theta - \mu \nabla_\theta L(\theta; S), \quad (6)$$

where  $\mu$  is the learning rate. Then in meta-test,  $L(\hat{\theta}, S_k^+)$  is computed on the augmented domain  $S_k^+$  for  $k = 1, \dots, K$ , where  $K$  is the number of augmented domains. This step evaluates the model's ability to handle augmented domains. Finally, in meta-update, the loss functions of meta-train and meta-test are combined and  $\theta$  is updated by following:

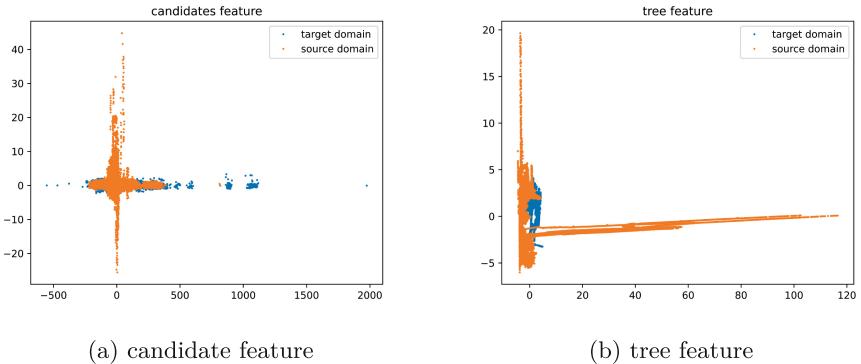
$$\theta \leftarrow \theta - \mu \nabla_\theta [L(\theta; S) + \sum_{k=1}^K L(\hat{\theta}, S_k^+)]. \quad (7)$$

## 4 Methodology

### 4.1 OOD Data Exists in Target Domain

In [40], there are two types of features used for the BVS policy: candidate feature  $C \in \mathbb{R}^{25 \times |C|}$  and tree feature  $Tree \in \mathbb{R}^{61}$ , where  $|C|$  represents the number of candidate variables. All features are related to the dynamic process of B&B tree. The candidate feature  $C \in \mathbb{R}^{25 \times |C|}$  captures the state of candidate variables, while the tree feature  $Tree \in \mathbb{R}^{61}$  captures the state of the search tree. From this perspective, all MILP instances can be handled in the same feature space by a neural network.

However, despite all instances sharing the same feature space, the features of different instances can vary significantly, particularly the candidate features. Consequently, OOD data may be present in the test set, that is, in the target domain. We apply Principal Component Analysis (PCA) [39] to reduce the high-dimensional features of both the training set and the test set to two dimensions, respectively. Figure 1 illustrates the results of dimensionality reduction for both the candidate features and the tree features.



**Fig. 1.** Candidate feature and tree feature after PCA dimensionality reduction in source (training) domain and target domain. Here the dataset comes from [40].

From Fig. 1a, we observe that there is some OOD data in the target domain compared to the source domain. In contrast, Fig. 1b shows that the source domain is more widely distributed than the target domain and can essentially encompass it.

Therefore, to further improve the generalization of the BVS policy on completely unknown target domains, it is necessary to generate candidate features that differ from those in the source domain and incorporate these data into the training process.

## 4.2 Special Policy Training

To train a special policy for OOD data, we use the M-ADA framework [30] for reference, which is based on adversarial augmentation [37]. The core idea of this framework is to generate data that causes the model's loss to increase and incorporate these data into the model training process. The generated data should be both diverse and different from the original data, while maintaining the same labels. Here, we assume that the generated candidate features are more likely to compensate for those OOD features in target domain because these features are more widely distributed than the features in the source domain. Moreover, the generated candidate features correspond to the same variable selection as the original candidate features.

During policy training, meta-learning is employed as described in Sect. 3.4 to achieve generalization on the target domain, which consists of three steps: meta-train, meta-test and meta-update [30].

We use TreeGate [40] as the neural network for our special policy. Under this framework, we define hidden state  $\mathbf{h}$  as the portion preceding the pooling layer.

**Adversarial Augmentation.** In data generation section, the overall objective function  $L_{ADA}$  is as follows:

$$L_{ADA} = L_{policy}(\theta; C) - \alpha L_{const}(\theta; \mathbf{h}) + \beta L_{relax}(\phi; C), \quad (8)$$

where  $\alpha$  and  $\beta$  are used to control the proportions of  $L_{relax}$  and  $L_{const}$  in  $L_{ADA}$ . The special policy is parameterized by  $\theta$ .  $L_{policy}$  is the loss function of the original BVS policy and is designed to ensure that the augmented data increases the policy's loss.  $L_{const}$  is employed to keep the selected variable unchanged by minimizing changes to the hidden state after augmentation. We have:

$$L_{const} = \frac{1}{2} \|\mathbf{h} - \mathbf{h}^+\|_2^2, \quad (9)$$

where  $\mathbf{h}^+$  is the hidden state of augmented data.

$L_{relax}$  is used to enhance the diversity of candidate features by increasing the reconstruction error of an Auto-Encoder. Specifically, we employ a WAE that is pre-trained on the source domain. In this WAE, the network of encoder adopts the NoTree structure [40], and the structure of decoder is similar to a reverse NoTree structure. The WAE training process is shown in Sect. 3.3. After pre-training, we can obtain a WAE  $V_\phi$  and a discriminator  $D_\gamma$ . Then, we have:

$$L_{relax} = \|C^+ - V_\phi(C^+)\|^2, \quad (10)$$

where  $C^+$  is the augmented candidate feature.

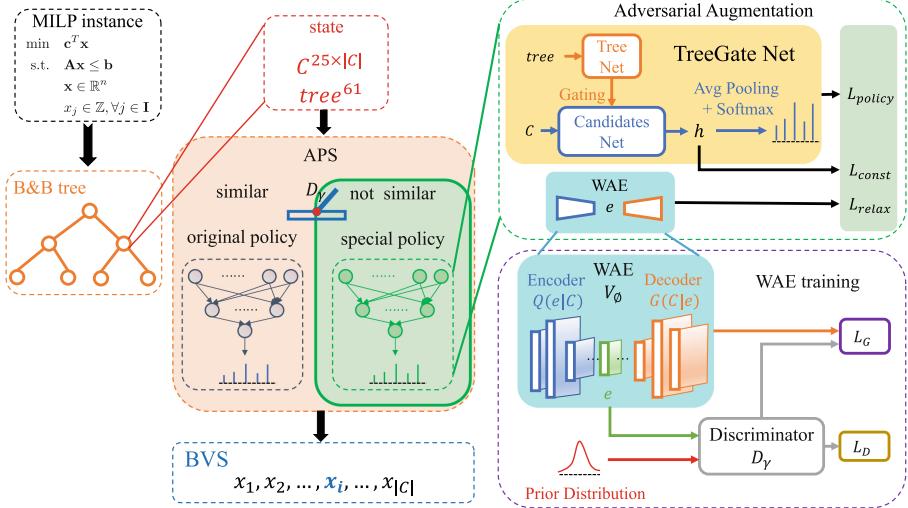
Given the precise form of the overall objective function, we iterate the following formula to generate the augmented data:

$$C_{t+1}^+ \leftarrow C_t^+ + \eta \nabla_{C_t^+} L_{ADA}(\theta, \phi; C_t^+, \mathbf{h}_t^+), \quad (11)$$

where  $\eta$  is the learning rate.

Simultaneously, each time a new augmented domain is generated, we re-train  $V_\phi$  with the data from this domain. The overall framework of adversarial augmentation is illustrated in Fig. 2.

**Meta-learning.** We employ the meta-learning method mentioned in Sect. 3.4 to train the special policy by combining source domain features  $C$  and augmented domain features  $C^+$ . The  $L$  in formula 6 and formula 7 need to be replaced with  $L_{policy}$ .



**Fig. 2.** The overall framework of APS. Under this framework, we can switch to the appropriate policy based on whether the current state is similar to the source domain.

### 4.3 Adaptive Policy Switching Framework

In practice, we can only train the BVS policy on a training set (source domain), while the instance we need to solve is completely unknown. The features from this unknown instance may be similar to those in the source domain, or they may be OOD. Therefore, the special policy obtained in Sect. 4.2 may not necessarily perform well on this unknown instance.

In summary, a natural approach is to use a combination of the original Tree-Gate policy and the special policy: the original policy should be used when the feature is similar to the source domain, and the special policy should be used when the feature is OOD. The key question, then, is how to determine whether the feature of the current state is similar to the source domain.

Coincidentally, during the WAE training, we happened to train a discriminator  $D_\gamma$ . Since the training data for the WAE pre-training comes from the source domain, the trained discriminator  $D_\gamma$  can classify data similar to the source domain as the positive class. With the help of discriminator  $D_\gamma$ , we can adaptively switch policies based on the feature of current state. The overall framework of APS is illustrated in Fig. 2.

## 5 Experiments

### 5.1 Settings

**Dataset.** In previous works, in order to test the generalization or transfer ability of the BVS policy, the number of constraints or variables is often increased to create heterogeneous instances. However, this approach does not fundamentally

change the structure of the original instance. Our goal is for the BVS policy to exhibit strong transfer performance on any instances, not just more complex versions of the original instance.

Thus, in [40], a dataset was constructed to better test the generalization of the BVS policy. The training set and the test set are composed of different instances from various versions of MIPLIB [5, 12, 18], as shown in Table 1.

**Table 1.** Details of the data set from [40]

Training Set	vpm2, swath1, stein27, sp150x300d, rmatr100-p10, rmatr100-p5, pp08aCUTS, neos648910, neos-476283, neos20, neos21, misc03, lseu, l152lav, istanbul-no-cutoff, eil33-2, dcmulti, air04, air05
Test Set	mine-166-5, ns1830653, rail507, neos18, map18, seymour1, neos11, nu25-pr12

As in [40], we train the BVS policy to imitate `relopscost`, the default BVS rule in SCIP. Here our goal is not to outperform the `relopscost`, as the expert rules in SCIP, like `relopscost`, are finely tuned over a large number of instances.

**Evaluation Criteria.** The ultimate goal of applying BVS policy is to accelerate MILP solving. However, the solving time is significantly affected by the hardware, and the time required may vary even when using the same BVS policy on different hardware. This variability can arise from other components of the solution process, such as the simplex method, which are unrelated to the BVS policy. Therefore, like most previous work, we use the number of nodes in the B&B tree as the evaluation criteria. To assess the generalization of the BVS policy, we test only on completely unknown instances from the test set.

**Baselines.** In the experiment, the baselines we selected include GCNN [11] and TreeGate [40]. In Sect. 1, we discussed that the GCNN serves as the foundational framework for numerous related studies. Within our work, the TreeGate model constitutes the core basis. Both the original policy and the special policy implemented in our study are built upon the TreeGate network architecture. Furthermore, TreeGate is recognized as the SOTA method in this field.

We also compare our method against the expert rule, pseudocost branching. Since our primary objective is to evaluate generalization, neural networks may perform poorly in some cases. Therefore, we include a random BVS policy as a worst-case scenario.

## 5.2 Results

The experimental results are shown in Table 2.

**Table 2.** The results of various policies. For each policy, we tested 5 different random seeds  $\{0, 1, 2, 3, 4\}$  on each instance separately and averaged the results. Additionally, we computed the average result across all instances. Each instance was subject to an upper time limit of 1 h for solving, and results that met the 1 h limit were marked \*. Note that the GCNN results are sourced from [40].

Instance	APS(ours)	TreeGate	GCNN	random	pseudocost
All Test	<b>5326.15</b>	6490.20	145664.35	151691.50	33951.98
mine-166-5	<b>15559.80</b>	20172.20	246793.20*	340937.20*	124821.00*
ns1830653	<b>10770.60</b>	14539.40	64815.20*	269550.00*	39278.20
rail507	6243.60	<b>4104.80</b>	76201.40*	77354.60*	4795.40
neos18	<b>3483.80</b>	6795.60	90682.40	295645.60*	93076.20
map18	1115.40	<b>481.40</b>	5613.20*	10330.60	1092.60
seymour1	2052.20	<b>1849.00</b>	323783.40*	173656.20*	2648.00
neos11	2431.80	2638.40	<b>1985.20</b>	27188.40	4335.60
nu25-pr12	<b>952.00</b>	1340.80	355440.80*	18869.40	1568.80

First, considering the average results across all test instances, APS outperforms all other methods, surpassing the previous SOTA, TreeGate, and reducing the number of nodes by 18%. The GCNN method performs significantly worse than both APS and TreeGate. In fact, its performance is even inferior to the heuristic rule pseudocost and is close to that of the random BVS policy. This result is expected because GCNN does not address the generalization problem, and the bipartite graph structure of the heterogeneous instances differs significantly from the source domain.

We then compared all policies on each instance. APS outperforms the other methods on instances such as mine-166-5, ns1830653, neos18, and nu25-pr12. However, it performs worse than TreeGate on rail507, map18, and seymour1, and is also outperformed by GCNN on neos11. Notably, these four instances where APS falls short are relatively simple. For simpler instances, state features are more likely to be included in the training set, and the special policy may struggle with such familiar features. If the discriminator fails to accurately identify whether a feature is similar to the source domain, the special policy may be incorrectly applied, leading to results that are slightly worse than TreeGate. In contrast, for more complex instances, particularly mine-166-5 and ns1830653, APS significantly reduces the number of nodes compared to TreeGate.

It should be noted that the TreeGate results presented here differ slightly from those in the original paper [40] due to variations in hardware. However, the original policy used in APS is identical to the TreeGate policy shown in Table 2, which sufficiently demonstrates that the use of APS improves generalization.

### 5.3 Ablation

We conducted the ablation experiment to demonstrate that the APS framework is crucial for achieving improvements. To do this, we evaluated the performance of the special policy, the original policy and APS separately. In this context, the original policy is TreeGate, and the experimental results for the special policy are provided in Table 3.

**Table 3.** Ablation experiment, the comparison of APS, original policy and special policy.

Instance	APS	Special Policy	TreeGate
All Test	<b>5326.15</b>	16132.90	6490.20
mine-166-5	<b>15559.80</b>	36800.80	20172.20
ns1830653	<b>10770.60</b>	44624.00	14539.40
rail507	6243.60	33267.40	<b>4104.80</b>
neos18	<b>3483.80</b>	7894.20	6795.60
map18	1115.40	<b>481.00</b>	481.40
seymour1	2052.20	2290.60	<b>1849.00</b>
neos11	<b>2431.80</b>	3266.00	2638.40
nu25-pr12	952.00	<b>439.20</b>	1340.80

We can see that the results of the special policy are worse than those of TreeGate and APS in most cases. This indicates that APS improves performance by adaptively switching the BVS policy based on the current state, which is the key factor in our approach’s superior performance compared to TreeGate.

This result is also consistent with Fig. 1, which shows that, in most cases, the features of the target domain are similar to those of the source domain, with only a small proportion being OOD. Therefore, it is sufficient to identify these OOD features rather than applying the special policy to all target domain features.

## 6 Conclusion

In this paper, we propose APS, an adaptive policy switching framework designed to improve the generalization of BVS policy on unknown instances. This approach leverages M-ADA, a domain generalization method to develop a specific policy for OOD data, and utilizes a discriminator to assess whether the state features of the current B&B tree are similar to those of the source domain. Experimental results demonstrate that APS achieves an 18% improvement in performance compared to TreeGate. However, our research is currently limited to the framework of imitation learning, which requires collecting `relabelcost` results or other expert branching results as labels to train policy. Our future work will focus on employing reinforcement learning in conjunction with APS,

aiming to develop a strong generalization policy based solely on interactions with the environment.

**Acknowledgements.** This study was supported by WFW-E3T0360101.

## References

1. Achterberg, T.: Constraint integer programming (2007)
2. Achterberg, T.: Scip: solving constraint integer programs. *Math. Program. Comput.* **1**, 1–41 (2009)
3. Achterberg, T.: branch\_relpscost.h file reference (Fri 21 June 2024). [https://scipopt.org/doc/html/branch\\_relpscost\\_8h.php](https://scipopt.org/doc/html/branch_relpscost_8h.php). Accessed 13 July 2024
4. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. *Oper. Res. Lett.* **33**(1), 42–54 (2005)
5. Achterberg, T., Koch, T., Martin, A.: Miplib 2003. *Oper. Res. Lett.* **34**(4), 361–372 (2006)
6. Barnhart, C., Laporte, G.: Handbooks in Operations Research and Management Science: Transportation. Elsevier, Amsterdam (2006)
7. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d’horizon. *Eur. J. Oper. Res.* **290**(2), 405–421 (2021)
8. Chen, Z.L.: Integrated production and outbound distribution scheduling: review and extensions. *Oper. Res.* **58**(1), 130–148 (2010)
9. Etheve, M., Alès, Z., Bissuel, C., Juan, O., Kedad-Sidhoum, S.: Reinforcement learning for variable selection in a branch and bound algorithm. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pp. 176–185. Springer, Cham (2020)
10. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: International Conference on Machine Learning, pp. 2052–2062. PMLR (2019)
11. Gasse, M., Chételat, D., Ferroni, N., Charlin, L., Lodi, A.: Exact combinatorial optimization with graph convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
12. Gleixner, A., et al.: Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Math. Program. Comput.* **13**(3), 443–490 (2021)
13. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol. 27 (2014)
14. Gupta, P., Gasse, M., Khalil, E., Mudigonda, P., Lodi, A., Bengio, Y.: Hybrid models for learning to branch. In: Advances in Neural Information Processing Systems, vol. 33, pp. 18087–18097 (2020)
15. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning, pp. 1861–1870. PMLR (2018)
16. Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering diverse domains through world models. arXiv preprint [arXiv:2301.04104](https://arxiv.org/abs/2301.04104) (2023)
17. Hansen, N., Su, H., Wang, X.: TD-MPC2: scalable, robust world models for continuous control. In: The Twelfth International Conference on Learning Representations (2024), <https://openreview.net/forum?id=Oxh5CstDJu>

18. Koch, T., et al.: Miplib 2010: mixed integer programming library version 5. *Math. Program. Comput.* **3**, 103–163 (2011)
19. Kostrikov, I., Nair, A., Levine, S.: Offline reinforcement learning with implicit q-learning. In: International Conference on Learning Representations (2022). <https://openreview.net/forum?id=68n2s9ZJWF8>
20. Kuang, Y., et al.: Rethinking branching on exact combinatorial optimization solver: the first deep symbolic discovery framework. In: The Twelfth International Conference on Learning Representations (2024)
21. Land, A.H., Doig, A.G.: An automatic method for solving discrete programming problems. Springer, Cham (2010)
22. Li, D., Lou, N., Zhang, B., Xu, Z., Fan, G.: Adaptive parameter sharing for multi-agent reinforcement learning. In: ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6035–6039 (2024). <https://doi.org/10.1109/ICASSP48485.2024.10447262>
23. Li, D., Xu, Z., Zhang, B., Zhou, G., Zhang, Z., Fan, G.: From explicit communication to tacit cooperation: a novel paradigm for cooperative marl. In: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems. AAMAS '24, pp. 2360–2362. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2024)
24. Linderoth, J.T., Savelsbergh, M.W.: A computational study of search strategies for mixed integer programming. *INFORMS J. Comput.* **11**(2), 173–187 (1999)
25. Liu, H., Kuang, Y., Wang, J., Li, X., Zhang, Y., Wu, F.: Promoting generalization for exact solvers via adversarial instance augmentation. arXiv preprint [arXiv:2310.14161](https://arxiv.org/abs/2310.14161) (2023)
26. Nelder, J.A., Mead, R.: A simplex method for function minimization. *Comput. J.* **7**(4), 308–313 (1965)
27. Parsonson, C.W., Laterre, A., Barrett, T.D.: Reinforcement learning for branch-and-bound optimisation using retrospective trajectories. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 4061–4069 (2023)
28. Pochet, Y., Wolsey, L.A.: Production Planning by Mixed Integer Programming, vol. 149. Springer, Cham (2006)
29. Potra, F.A., Wright, S.J.: Interior-point methods. *J. Comput. Appl. Math.* **124**(1–2), 281–302 (2000)
30. Qiao, F., Zhao, L., Peng, X.: Learning to learn single domain generalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12556–12565 (2020)
31. Rashid, T., Samvelyan, M., De Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S.: Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **21**(178), 1–51 (2020)
32. Scavuzzo, L., Aardal, K., Lodi, A., Yorke-Smith, N.: Machine learning augmented branch and bound for mixed integer linear programming. arXiv preprint [arXiv:2402.05501](https://arxiv.org/abs/2402.05501) (2024)
33. Scavuzzo, L., et al.: Learning to branch with tree MDPs. In: Advances in Neural Information Processing Systems, vol. 35, pp. 18514–18526 (2022)
34. Schulman, J.: Trust region policy optimization. CoRR abs/1502.05477 (2015)
35. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint [arXiv:1707.06347](https://arxiv.org/abs/1707.06347) (2017)
36. Tolstikhin, I., Bousquet, O., Gelly, S., Schoelkopf, B.: Wasserstein auto-encoders. In: International Conference on Learning Representations (2018). <https://openreview.net/forum?id=HkL7n1-0b>

37. Volpi, R., Namkoong, H., Sener, O., Duchi, J.C., Murino, V., Savarese, S.: Generalizing to unseen domains via adversarial data augmentation. In: Advances in Neural Information Processing Systems, vol. 31 (2018)
38. Wang, J., et al.: Generalizing to unseen domains: a survey on domain generalization. *IEEE Trans. Knowl. Data Eng.* **35**(8), 8052–8072 (2022)
39. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. *Chemom. Intell. Lab. Syst.* **2**(1–3), 37–52 (1987)
40. Zarpellon, G., Jo, J., Lodi, A., Bengio, Y.: Parameterizing branch-and-bound search trees to learn branching policies. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 3931–3939 (2021)
41. Zhang, B., et al.: Sequential asynchronous action coordination in multi-agent systems: a Stackelberg decision transformer approach. In: Forty-First International Conference on Machine Learning (2024). <https://openreview.net/forum?id=M3qRRkOuTN>
42. Zhang, J., et al.: A survey for solving mixed integer programming via machine learning. *Neurocomputing* **519**, 205–217 (2023)



# Efficient Pruning and Compression Techniques for Convolutional Neural Networks to Preserve Knowledge and Optimize Performance

Jakub Skrzyski<sup>(✉)</sup> and Adrian Horzyk<sup>(✉)</sup>

AGH University of Krakow, Mickiewicza Av. 30, 30-059 Krakow, Poland  
jskrzynski@student.agh.edu.pl, horzyk@agh.edu.pl

**Abstract.** Rapid growth in the complexity and size of Convolutional Neural Networks (CNNs) poses significant challenges in computational resources and energy consumption. This paper presents a novel approach to CNN optimization through a combined pruning and compression technique designed to preserve knowledge and enhance efficiency. Unlike traditional pruning methods that introduce sparsity and require specialized hardware, our method focuses on pruning entire filters based on their importance, followed by an innovative compression strategy that merges the least informative filters. This preserves the knowledge contained within these filters while maintaining the network's structural integrity. We propose a three-step algorithm: selecting low-information filters using entropy metrics, grouping similar filters, and merging these groups to retain critical information. Our approach significantly reduces the network size without compromising accuracy, as demonstrated using the CIFAR-10, MNIST, Fashion MNIST, and USPS datasets, as well as a modified and original VGG16 architecture. The experiments carried out have shown that our method achieves a substantial reduction in the number of parameters and Floating Point Operations (FLOPs), lowering computational costs by up to 86.82% while preserving up to 99% of the original accuracy of the model. This paper contributes to the field of deep learning by offering a scalable, hardware-agnostic solution for CNN optimization, making it highly suitable for deployment in resource-constrained environments. Future work will explore the application of this method to various architectures and datasets to further validate its efficacy and versatility.

**Keywords:** CNN pruning techniques · structured pruning · neural network compression · filter optimization in deep learning · filter merging · knowledge retention in pruned networks · entropy metrics · K-Medoids

---

We gratefully acknowledge the funding support by program “Excellence initiative-research university” for the AGH University in Krakow as well as the ARTIQ project: UMO-2021/01/2/ST6/00004 and ARTIQ/0004/2021.

## 1 Introduction

The demand for better-quality models is constantly increasing; new models are usually bigger and require more power and other computational resources to train and infer. The breakthrough of humans, the invention of convolutional neural networks, further accelerated the development of computational intelligence [13]. Today, we are facing a serious problem of lacking computational resources to run or prepare state-of-the-art models that can compete with the previous ones without increasing the demands for extra computational resources. In addition to limiting access to the best technology, it poses a significant environmental issue due to the rapidly growing demand for energy consumption. This raises a very significant topic of the possibility of compressing the networks as much as possible to reduce the demand for computational resources. In recent years, there have been quite a few discoveries in this field that have resulted in numerous publications, e.g., [3, 4, 6, 10, 12, 14–17, 19], which proposed methods that can significantly reduce network size. For example, [5] presents comprehensive comparisons of the currently available methods. It should be considered that it is not easy to optimize the network, as conventional, unstructured, pruning introduces sparsity into the network. To benefit from such an optimization, specialized hardware or software is needed to limit the executed operations considering the sparsity. As a solution to this problem, architectural pruning (also referred to as structured [5]) was proposed, which means pruning entire filters in convolutional networks [10]. It facilitates optimal execution even on regular CPUs or GPUs without a need for specialized hardware or software, as there is no sparsity in the network because whole clusters of weights are removed, imposing changes on the entire architecture but resulting in a network that may be calculated as the in an analogous way to the base one, without the need to handle sparsity. Moreover, such an approach significantly optimizes memory accesses, as data can be retrieved in a structured way, addressing the issue mentioned by [21] of an incomparably higher memory access cost to the computation cost.

Originally, simple methods were used to determine the informativeness of the filters. However, it was discovered that more complex metrics, such as the  $l_2$  norm, entropy, and geometric distance, perform significantly better in terms of balancing accuracy loss and reduced FLOPs [4, 17]. The main approach involves selecting the least important filters and either removing them through pruning or replacing them with more informative ones through grafting [17]. Various strategies exist to generate new and improved filters. Although pruning techniques primarily aim to compress the network, grafting preserves the existing architecture while enhancing the precision of the inference. Each method focuses on maximizing the efficiency of the given network. However, relying solely on either pruning or grafting presents significant drawbacks: pruning may not fully utilize the network architecture's potential, while grafting does not support network compression. The authors propose that preserving the knowledge stored in less important filters, rather than deleting them, can be highly beneficial. The core idea is to compress these less informative filters into fewer, while maintaining the same level of accuracy. The primary objective of the proposed solution is to

achieve better accuracy in a shorter time, with a potentially higher reduction in FLOPs compared to existing methods, by incorporating as much knowledge as possible into the new filters that replace the less important ones.

## 2 Description of the Introduced Method

### 2.1 General Method

In this article, we propose a new method that efficiently combines the benefits of pruning and grafting. Our method has been divided into three steps. Each of them is a research problem on its own and may be the subject of an interesting discussion. The general steps are defined as follows:

1. Choose filters containing little information: This step ensures that only the least important filters are passed for the following stages.
2. Group filters: During this step, filters are aggregated into groups of the most similar ones. This ensures that only similar filters are merged.
3. Merge filters in each group: This is the final step that introduces the gain in performance. Each group's filters are compressed into a single filter according to the chosen strategy. If this is not the last epoch of training, removed filters are zeroed out or filled with other data, depending on the adopted approach, which needs to be determined experimentally for each problem but not deleted entirely to make space for new knowledge.

---

**Algorithm 1.** General pruning and merging algorithm

---

```

1: for  $epoch_e$  in  $epochs$  do
2:   Train the network for a single epoch.
3:   for  $layer_i$  in  $layers$  do
4:     Sort filters in  $layer_i$  by amount of information they contain.
5:     Select a portion of least informative filters.
6:     Group the filters
7:     for  $group_k$  in  $groups$  do
8:       Merge filters from  $group_k$  into single one.
9:       Replace one of filters from  $group_k$  by the result of merging.
10:      if  $epoch_e$  is last epoch then
11:        Remove filters permanently from the structure of the network.
12:      else
13:        Free remaining filters belonging to  $group_k$  by either zeroing or filling with
           different weights than current, depending on the adopted approach, to
           allow new filters to be developed.
14:      end if
15:    end for
16:  end for
17: end for

```

---

It is possible to adapt different strategies to perform the above given steps, but for the sake of this paper, some of the most promising strategies were adopted to prove the concept. The main idea of the proposed algorithm is to run it at the end of each epoch, firstly freeing up space for new filters to be created, and in the last epoch, the freed-up filters may be safely deleted to achieve the final goal of compressing the network. The exact method, how to do it, is described by Algorithm 1. To further enhance the actual implementation and choice of hyperparameters, observations stated by [7] may be used.

The method may appear very similar to the one presented by [2], but the key difference is that in our method, we try to minimize the impact on the most developed filters by allowing the user to select the appropriate ratio of the best filters that should pass without being modified. Additionally, to minimize knowledge loss, instead of removing depleted filters, we compress them, causing similar filters to be merged into a single filter, maintaining potential useful features from the whole group.

## 2.2 Chosen Implementation

As [17] of the references point out, a very good measure of filter importance is the entropy; since the  $l_1$  and  $l_2$  norms are good in theory, they require certain conditions to be met [4], which causes it not to perform well in real-life situations. That makes these metrics prone to errors and causes incorrect filters to be selected. The conclusion drawn from the presented paper was subjected to tests; apparently, the  $l_1$  and  $l_2$  norms did not perform much worse, the difference in comparison to entropy was noticeable. Using entropy allowed for achieving increased accuracy, while both norms were oscillating around the base accuracy. For the reasons presented, entropy was chosen. The user can also decide on a fraction of the selected filters. In further description, let the **selection rate** be the fraction of filters that are selected out of all filters for further processing. First, the filters are ordered according to the entropy metrics, and then the subset containing the least informative filters with size defined by **selection rate** is created.

The grouping poses several challenges. The basic questions are how many groups should be created and how should the filters be grouped? For the sake of this paper, the following methodology was adopted and proved to work. The user specifies how many times the number of selected filters should decrease (a user-defined hyperparameter). This number is used to calculate the number of groups. Then, inspired by one of the referred papers [4], a geometric distance between the filters is calculated. With the calculated distances and the number of groups, the K-medoids algorithm is used. It results in, at most, K or, in specific cases, fewer clusters that are then passed into the merging step. In Algorithm 2, let us denote **compression factor** as a value by which the number of selected filters should be at least decreased. This value is passed as  $K$  for K-Medoids in this implementation [18].

The last and most beneficial step is to merge each cluster of filters. The ideas for completing this particular step may differ significantly. It may be done

by *min*, *max*, *sum*, and many more operations. We have studied the literature and performed experiments that have shown that averaging and taking maxima perform similarly, but averaging allows achieving better accuracy, while *max* results in oscillating around  $+/- 0.1\%$  of base accuracy. Based on the analysis of the experimental results, for the sake of this paper, as the most promising, the *avg* was chosen, that is, the numbers are averaged together for all members of the cluster with respect to their location in the filter. Let the  $G_{i,k}$  be the set of filters in  $k$ th group in  $i$ th layer and  $|G_{i,k}|$  be the number of filters in that group, then the averaged filter is given by (1):

$$\frac{\sum_{F_{i,j} \in G_{i,k}} (F_{i,j})}{|G_{i,k}|} \quad (1)$$

It is also important to realize that after pruning a high fraction of filters by a high factor, the network may need to be tuned in a couple of epochs. Next, pruning is conducted after each tuning epoch. One may also observe that this method introduces a slightly different approach to network design. We define the final network size using hyper-parameters, but, while training, the network is left with a spare space to evolve, a kind of slack that allows some experiments that are then integrated into a smaller space after each epoch, allowing better utilization of the resources.

### 3 Benchmarking Procedure

The first experiments were conducted in a limited resource environment; therefore, a trial was conducted to make a meaningful comparison. This trial was carried out on the same environment configuration as the main experiments with the same network architecture as the main one. Such an approach allowed inferring conclusions from the outcome of experiments that are not influenced by any external factors apart from the actual effect of method usage. Most of the referred works use the CIFAR-10 [9] as the benchmarking dataset, which was also adopted in these trials to compare the method with the results of the state-of-the-art methods. The performance of the model is measured by performing the classification. As VGG16 architecture was chosen, unfortunately, due to limited computational capacity in initial trials, it had to be modified by disabling a large part of convolutional filters to meet resource constraints. For the first experiments, only seven VGG blocks from the original VGG16 architecture were used. As the results were promising, subsequent experiments were conducted with the use of the full VGG16 architecture. The trials were also conducted on a few other datasets to demonstrate that the method is universal and robust.

#### 3.1 Measuring Procedure

The procedure for taking measurements is defined as follows. In each epoch, training is performed first; after that, the inferred results during training are

**Algorithm 2.** Detailed implementation of pruning and merging algorithm

**Hyperparameters:**  $selection\_rate_i$  - a fraction of the least effective filters to be selected for the compression procedure for the  $i$ th layer, later denoted as  $SR_i$ .  $compression\_factor_i$  - a number defining how many times the number of selected filters should be decreased after compression for the  $i$ th layer, later denoted as  $CF_i$ . Hyperparameters are defined for each layer subjected to the procedure, and for the skipped layers, it may be said that  $SR_i = 0$  and  $CF_i = 1$ .

**Notions:** The total number of epochs is denoted as  $e_{max}$  and  $e$  is the number of epoch. The number of layers subjected to the algorithm is denoted as  $i_{max}$  and then  $i$  is a layer index. Simmilarly to [10]  $n_i$  is a number of filters in the  $i$ th layer. Then let  $s_i$  and  $k_i$  denote a number of filters selected for grouping and a number of groups to be created in the  $i$ th layer respectively.

```

1: for  $e$  in  $1..e_{max}$  do
2:   Train the network for a single epoch
3:   for  $i$  in  $1..i_{max}$  do
4:     Compute the entropy of each filter in the layer  $i$ 
5:     Sort filters in  $i$  with respect to the calculated entropy
6:      $s_i = n_i * SR_i$ 
7:     Select  $s_i$  least informative filters according to the computed entropy and performed
       sorting
8:     Compute the geometrical distance between each pair of filters in layer  $i$ 
9:      $k_i = \text{floor}(n_i/CF_i)$ 
10:    Cluster selected filters according to the K-Medoids algorithm with  $K = k_i$ 
11:    for  $g$  in  $1..k_i$  do
12:      Compute the average of all filters in the group  $g$  in a manner that fields of a
        resulting filter are averaged values occupying the same positions in the source
        filters.
13:      Replace the first filter in the group with the one created using the average
14:      if  $e == e_{max}$  then
15:        Remove remaining filters belonging to  $g$  permanently from the structure of the
           network
16:        Remove the corresponding biases
17:        Update batch normalization layer according to if it exists.
18:      else
19:        Zero values of the remaining filters belonging to  $g$ 
20:        Zero the corresponding biases
21:        Update the batch normalization layer according to if it exists.
22:      end if
23:    end for
24:  end for
25: end for
```

The geometric distance is computed analogously to the distance in a Cartesian coordinate system, the square root of the sum of squares of differences of values occupying the same position in filters. Let  $F_{i,j}$  be the  $j$ th filter of the  $i$ th layer, then assuming the filters are three-dimensional of equal shapes, indexed by the coordinates  $x, y, z$ , the geometric distance for the filters  $k$  and  $l$  of the  $i$ th layer is given by (2). This could be represented as a  $l_2$  norm of the difference of filters as presented in (3).

$$\sqrt{\sum_{x=0}^{x_{max}} \sum_{y=0}^{y_{max}} \sum_{z=0}^{z_{max}} (F_{i,k}[x, y, z] - F_{i,l}[x, y, z])^2} \quad (2)$$

$$\|F_{i,k} - F_{i,l}\|_2 \quad (3)$$

used to compute the training accuracy. In the third step, the validation metrics are calculated in the validation set. Then, pruning is executed, and accuracy measurements are performed again for the validation dataset. This means that the training metrics are directly influenced by pruning from the previous epochs, giving insight into the recovery abilities of the network.

### 3.2 Calculating Number of Pruned Filters

To calculate the number of pruned filters for each layer  $i$ , it is necessary to know  $SR_i$  and  $CF_i$ , as well as the initial number of filters denoted here as  $n_i$ . The number of selected filters is defined as  $n_i \cdot SR_i$ . The number of filters created by the compression algorithm is equal to the number of filters divided by the compression factor; then, it is possible to calculate it as shown in (4).

$$\frac{n_i \cdot SR_i}{CF_i} \quad (4)$$

Then, to achieve the number of filters that are going to be deleted, we may subtract the number of created filters (4) from the number of the initially selected filters. The formula is presented in (5)

$$n_i \cdot SR_i - \frac{n_i \cdot SR_i}{CF_i} \quad (5)$$

Then, we may simplify the whole expression to achieve the final formula, allowing the calculation of a number of pruned filters. The formula is presented in (6). Then, it may also be simplified, as in (7), to achieve just a deleted filter rate compared to the original number.

$$n_i \cdot SR_i \cdot (1 - CF_i^{-1}) \quad (6)$$

$$SR_i \cdot (1 - CF_i^{-1}) \quad (7)$$

This directly influences a number of performed FLOPs as the method takes advantage of pruning the structure of the network instead of weights. The number of FLOPs for each layer can be calculated as the product of the number of filters, their size, the number of input channels, and their size. Single pruning will then affect two layers as it decreases the number of filters in a current layer and the number of created feature maps passed to the next layer as input channels.

Let  $DR_i$  be the rate of filters deleted compared to the original number of filters for the  $i$ th layer, described in (7) and  $FO_i$  be the original number of FLOPs for the  $i$ th layer, then the number of FLOPs after pruning is defined as presented by (8).

$$FO_i * (1 - DR_i) * (1 - DR_{i-1}) \quad (8)$$

As presented in the formulas above, the level of pruning may be defined as a functional relation of the architecture and hyperparameters. That is a huge advantage as it is known a priori, which is the desired outcome. In conclusion, lower training costs as careful planning can precede the training phase.

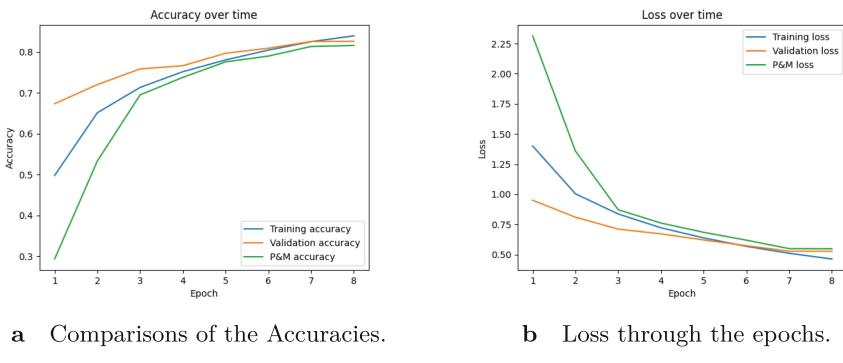
### 3.3 Complexity

Having described the algorithm, it is necessary to analyze the cost of running it. Therefore, computational complexity needs to be discussed. In each phase of the algorithm, we first need to calculate the entropy by performing an operation that is linear with respect to the size of a given kernel. This operation is repeated for every kernel. This gives a complexity of  $O(n_i * v_i)$  where  $v_i$  is the filter size in the  $i$ th layer. The second step is the calculation of distance that takes into account  $n_i \cdot SR_i \cdot (1 - CF_i^{-1})$  filters. The operation of distance calculation is quadratic as it computes the pairwise distance. Each pair is calculated in a time that corresponds to the size of the filter. In total, the second step is done in  $(n_i \cdot SR_i \cdot (1 - CF_i^{-1}))^2 \cdot v_i$  operations for each layer. The third step, K-medoids, is performed in a complexity of  $O(mxitr * k_i * s_i^2)$  where  $mxitr$  is a hyperparameter that controls the maximal number of iterations of K-medoids. The overall complexity highly depends on the selected configuration of the algorithm. The total complexity of the computation of a single layer is given by  $O(n_i * v_i) + O((n_i \cdot SR_i \cdot (1 - CF_i^{-1}))^2 \cdot v_i) + O(mxitr * k_i * s_i^2)$ .

## 4 Experiments

### 4.1 The Initial Trial

The initial trial was conducted with a small filter selection rate (0.4 for the 1st to 5th and 0.2 for 6th layers) and a small compression factor defined by (2). This ensures that only 40% and 20% of the worst filters are selected, and then their number is decreased by 2. Based on these assumptions, the  $0.4 \cdot n_i - 0.4 \cdot n_i/2 = 0.2 \cdot n_i$  filters were removed for the first layers.



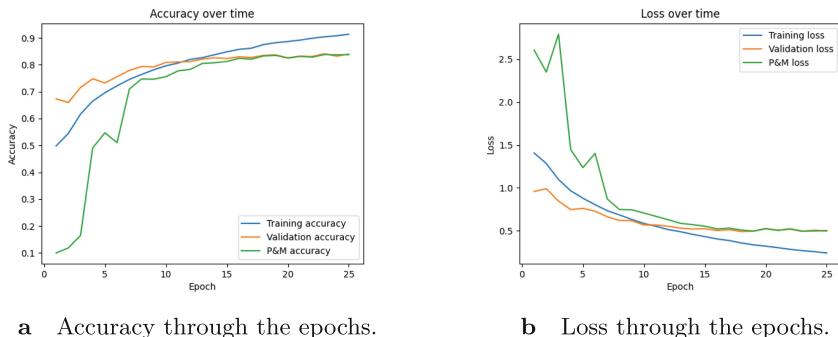
**Fig. 1.** Comparisons of the Accuracies and Losses for the first trial described in Sect. 4.1.

Examining the results, one can easily see that after each pruning round, the network easily regains its capabilities, which is visualized by the constant growth

of the training accuracy in Fig. 1a, and it is also clearly visible by the loss in Fig. 1b. Bearing in mind that training accuracy is directly affected by pruning and is computed while training, not after, it shows how quickly the network is recovering. Additionally, we may observe that the accuracy after pruning also quickly increases. According to data collected and presented in Fig. 1a, the network shows signs of adopting pruning just after the fourth epoch of training with pruning. Then, the curve of pruning accuracy tends closer to the validation accuracy (removing overfitting). It is also important to recall that here, the training accuracy curve is computed a bit differently than conventionally, as this metric is also used to measure the speed of adopting the network's change. This answers a potential question of why the training curve is lower than the validation one.

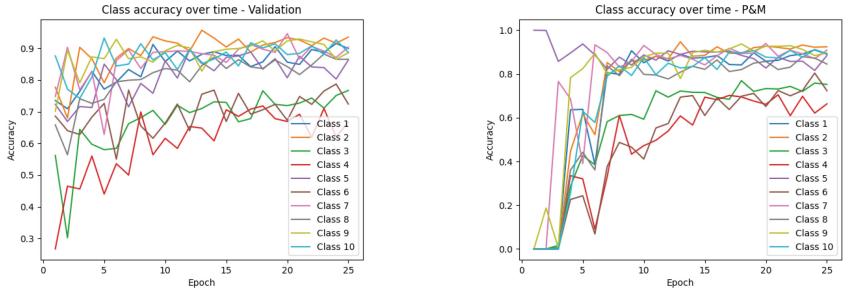
## 4.2 Heavy Pruning of the Network

As small-magnitude pruning achieved good results, the rate of pruned filters was increased. During this trial, about 68% of the filters were removed from layers 1 to 6. Such statistics were achieved by setting the  $SR$  for layers 1 - 6 to 0.8, which means that 80% of the filters will be affected, and setting compression factors as follows: for layers 1 and 2,  $CF = 5$ ; for 3 and 4,  $CF = 7$ ; for 5 and 6,  $CF = 10$ . In this experiment, we start from a randomly initialized network and run the algorithm after each training epoch. The process is continued for 25 epochs.



**Fig. 2.** Comparisons of the Accuracies and Losses for trial described in Sect. 4.2

The plots in Fig. 2 show very interesting behavior, in which after around 15 epochs, the accuracy and loss of the after-pruning probe seem to exactly follow the validation probe. That demonstrates that the algorithm works and can yield a valuable result in a time significantly shorter than that of the state-of-the-art methods currently available. Additionally, we can see that compression is constant since the training process is stabilized and could potentially be prolonged or tuned to achieve even better model fitting. Examining Fig. 3b and

**a** Metrics collected while validating**b** Metrics collected after pruning

**Fig. 3.** Comparisons of the Accuracies with separation to classes for trial described in Sect. 4.2. The metrics were collected while validating and after pruning.

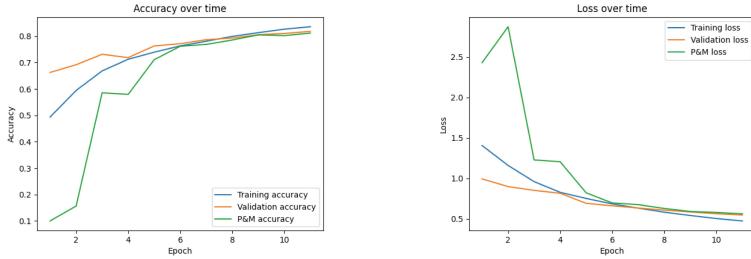
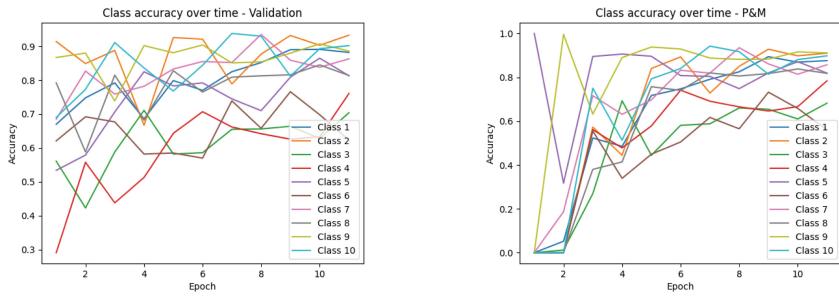
comparing with Fig. 3a, in more detail, we can also see that for some classes, it is enough to repeat training for just around six epochs to compress the knowledge gained. The results of a trial show that the method has great potential. It is done while training and does not require the retraining phase. This makes it possible to achieve acceptable results in prediction accuracy in a much shorter time frame than currently available methods, allowing model compression and a cost-efficient training procedure.

According to the presented method of calculating FLOPs after pruning is executed, for the used network, it was possible to achieve the following ratios of the new numbers of FLOPs to the original: 0.36, 0.1296, 0.1152, 0.1024, 0.0896, 0.0786 for layers of 1 to 6, respectively. The results were compared with the state-of-the-art algorithms, with the difference that the latter achieved the result much faster than the referenced methods. The total number of FLOPs for the entire network measured with the pytorch profiler has dropped to 26.7% of the original value. This observation shows that the proposed pruning method may be considered an advancement or alternative to the state-of-the-art methods.

### 4.3 Further Increasing the Hyperparameters

As the previous trial was successful, another was conducted to test whether it is possible to use even higher magnitude running. The selection rate is again 80% for all layers considered. The compression factor is set to 5 for layers 1 and 2, 7 for layers 3 and 4, 10 for layer 5, and 12 for layer 6.

The results are presented in Figs. 4 and 5. In Fig. 5b, we can clearly observe that for different classes, the algorithm is even beneficial in terms of accuracy. Additionally, the results collected from this experiment suggest that, thanks to the method used, the network adapted very quickly to the pruning, and it might have been finished by finally removing all of the selected filters instead of zeroing them. It took only a couple of epochs to reach that state. It can be identified by

**a** Accuracy through the epochs.**b** Loss through the epochs.**Fig. 4.** Comparisons of the Accuracies and Losses for the trial described in 4.3**a** Accuracy with separation to classes computed for validation phase**b** Accuracy with separation to classes computed after pruning phase**Fig. 5.** Comparisons of the Accuracies with separation to classes for the trial described in 4.3, where metrics were collected while validating and after pruning.

observing that all the selected filters are assigned to a single cluster because there are no significant differences between them. The observation also demonstrates that the algorithm allows the model to catch up with the original accuracy and loss within about 6 epochs. The training procedure then remains stable, and if continued, it would allow for an even better fit.

The filter reduction is 64% for the first two layers, 68% for the next two layers, 72% for the fifth layer, and 73% for the last one. That also means a value of the new to original FLOP ratio equals 0.36, 0.1296, 0.1152, 0.1024, 0.0896, and 0.0756 for layers from 1 to 6. The total profiler ratio of the remaining FLOPs to the initial ones is calculated to be 26.5%.

## 5 Extensive Testing of the Method

The efficiency of the method was extensively tested on multiple datasets. In this article, we present test results for the CIFAR-10 [9], the MNIST [1], Fashion MNIST [20], and USPS [8], datasets. The main comparisons with the methods

presented in [2, 2, 4, 10, 11, 14, 16, 22] were presented for the CIFAR-10 dataset and the VGG16 architecture. For the main criterion, we have selected the accuracy achieved, the accuracy drop, the number of epochs to achieve the results, and the drop in needed resources.

The referenced papers are not always clear about the exact procedure, as an example [2] does not explicitly present the number of training and retraining epochs; however, it presents a comparison of times based on 200 training epochs. In such a case, our method has the significant advantage of achieving even better accuracy in far fewer epochs.

Experiments with the CIFAR-10 dataset were conducted with 30 epochs of pretraining the model and retraining for 120 epochs, which gives 150 epochs of training in total. The configuration of the hyperparameters allowed the drop in 86.82% of FLOPs. Depending on the exact training policy, it was possible to achieve a gain in accuracy or a slight loss within the acceptable range. The results of several trials are presented in Table 1. All of the results achieved confirm that our introduced method is stable and allows for rapid model recovery, saving time and computational resources while achieving very good compression of the models. Table 2 shows the comparisons of the results achieved with other tools currently available. It shows clearly that the proposed method advances the state-of-the-art by allowing similar and better effects to be achieved with less resources than other methods.

**Table 1.** Example results of using the algorithm on model for CIFAR-10 classification with hyperparameters allowing for over 86% drop in number of FLOPs

Base accuracy	Accuracy after pruning	Change
87.45%	86.57%	-0.88 pp
86.12%	87.27%	+1.15 pp
87.61%	87.14%	-0.47 pp
86.72%	87.03%	+0.31 pp
85.76%	85.94%	+0.18 pp

Moreover, to prove that the algorithm is applicable for the general case, it was tested on the other data sets mentioned above: USPS [8], MNIST [1], and Fashion MNIST [20]. It was also possible to achieve a significant decrease in FLOPs performed for a single inference by 86.82% while maintaining accuracy within an acceptable range from the original accuracy of the model. The results are presented in Table 3. To achieve the results, the same procedure was used as in the CIFAR-10 case. It is important to note that despite the same procedure, the models gained interesting performance even before the training was stopped. For MNIST, in the 55th epoch, the model achieved an accuracy of 99.42%, and for Fashion MNIST 93.4% in the 120th epoch. In the case of USPS, the results were achieved within 60 epochs.

**Table 2.** The introduced algorithm compared to the other currently available methods on CIFAR-10 and VGG16, where dash means either no data in original publication or that metric does not apply.

Method	Base accuracy	Pruned accuracy	Change [pp]	Training epochs	Retraining epochs	Decrease in FLOPs
Li [10]	93.25%	93.4%	+0.15	-	40	34.2%
AA-E [14]	92.92%	92.72%	-0.2	160	100	79.69%
AFP-E [14]	92.92%	92.94%	+0.02	160	100	79.69%
AA-F [14]	92.92%	92.44%	+0.48	160	100	81.39%
AFP-F [14]	92.92%	92.87%	-0.05	160	100	81.39%
FPGM from scratch SA [4]	93.58%	93.54%	-0.04	-	-	34.2%
FPGM from scratch [4]	93.58%	93.23%	-0.35	-	-	35.9%
FPGM [4]	93.58%	80.38%	-13.2	-	0	-
FPGM [4]	93.58%	93.24%	-0.34	-	40	-
FPGM [4]	93.58%	94.00%	+0.42	-	160	-
SWP[16]	93.25%	93.65%	+0.4	160	-	71.16%
APSSF-2 [2]	84.28%	84.26%	-0.02	-	-	95.38%
APSSF-4 [2]	83.26%	83.23%	-0.03	-	-	97.72%
HRank [11]	93.96%	93.43%	-0.53	-	30 for each layer	53.5%
HRank [11]	93.96%	92.34%	-1.62	-	30 for each layer	65.3%
HRank[11]	93.96%	91.23%	-2.73	-	30 for each layer	76.5%
Weight-based [22]	93.83%	93.86%	+0.03	-	-	50.5%
Weight-based [22]	93.83%	93.48%	-0.53	-	-	50.5%
Our	86.12%	87.27%	+1.15	30	120	86.82%
Our	87.45%	86.57%	-0.88	30	120	86.82%

**Table 3.** Results of the introduced algorithm tested on the other datasets.

Dataset	Base accuracy	Pruned accuracy
MNIST	99.7%	99.43%
Fashion MNIST	93.39%	93.42%
USPS	97.61%	96.91%

## 6 Conclusions and Further Work

This paper introduced a novel approach to the optimization of Convolutional Neural Networks (CNNs) through a unique combination of pruning and compression techniques aimed at preserving knowledge while significantly reducing computational requirements and other resources. Our method departs from conventional pruning strategies that introduce sparsity and often require specialized

hardware. Instead, it focuses on the complete removal of less informative filters, followed by a sophisticated compression process that merges these filters to retain the network’s critical information. This approach allows the network to maintain its structural integrity and perform efficiently on standard hardware.

The proposed three-step algorithm—comprising filter selection based on entropy metrics, grouping of similar filters, and merging these groups—demonstrates significant advantages. By evaluating the informational content of the filters and strategically compressing them, our method ensures a minimal loss of accuracy. This is supported by extensive experiments on the CIFAR-10, MNIST, and Fashion MNIST datasets using the modified and original VGG16 architectures, where our approach achieved a reduction of up to 86.82% in the number of computational operations. In particular, the network retained up to 99% of its original accuracy after optimization.

These results underscore the potential of our technique to effectively reduce CNN size and computational load without sacrificing performance, making it highly applicable in environments where computational resources are limited. The ability to operate efficiently on general-purpose CPUs and GPUs, without the need for specialized sparsity-aware hardware, enhances its practical relevance.

Our contributions include not only the development of an innovative pruning and compression algorithm, but also the demonstration of its efficacy through rigorous empirical evaluations. This work advances state-of-the-art neural network optimization by providing a robust and scalable solution that bridges the gap between high-performance and resource-constrained deployment scenarios.

Future research directions will focus on extending this approach to other neural network architectures and larger and more complex data sets to further validate its generalizability and effectiveness. Additionally, exploring automated techniques for fine-tuning the hyperparameters of our method could potentially yield further improvements in efficiency and performance.

## References

1. Deng, L.: The mNIST database of handwritten digit images for machine learning research. *IEEE Sig. Process. Mag.* **29**(6), 141–142 (2012)
2. Geng, L., Niu, B.: APSSF: adaptive CNN pruning based on structural similarity of filters. *Int. J. Comput. Intell. Syst.* **17**(1), 1–27 (2024)
3. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint [arXiv:1808.06866](https://arxiv.org/abs/1808.06866) (2018)
4. 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, pp. 4340–4349 (2019)
5. He, Y., Xiao, L.: Structured pruning for deep convolutional neural networks: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* (2023)
6. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1389–1397 (2017)

7. Huang, Z., Shao, W., Wang, X., Lin, L., Luo, P.: Rethinking the pruning criteria for convolutional neural network. In: Advances in Neural Information Processing Systems, vol. 34, pp. 16305–16318 (2021)
8. Hull, J.J.: A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. **16**(5), 550–554 (1994). <https://doi.org/10.1109/34.291440>
9. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
10. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets (2017). <https://arxiv.org/abs/1608.08710>
11. Lin, M., et al.: Hrank: filter pruning using high-rank feature map (2020). <https://arxiv.org/abs/2002.10179>
12. Luo, J.H., Wu, J., Lin, W.: Thinet: a filter level pruning method for deep neural network compression. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5058–5066 (2017)
13. Mahmud, M., Kaiser, M.S., McGinnity, T.M., Hussain, A.: Deep learning in mining biological data. Cogn. Comput. **13**(1), 1–33 (2021)
14. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data, **29**, 2063–2079 (2018)
15. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. IEEE Trans. Neural Netw. Learn. Syst. **29**(6), 2063–2079 (2018)
16. Meng, F., et al.: Pruning filter in filter. In: Advances in Neural Information Processing Systems, vol. 33, pp. 17629–17640 (2020)
17. Meng, F., et al.: Filter grafting for deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6599–6607 (2020)
18. Skrzynski, J., Horzyk, A.: Efficient Pruning and Compression Techniques for Convolutional Neural Networks to Preserve Knowledge and Optimize Performance - Implementation. <https://github.com/j-skrzynski/Efficient-Pruning-and-Compression-Techniques-for-CNNs>
19. Wang, Z., Li, C., Wang, X.: Convolutional neural network pruning with structural redundancy reduction. CoRR abs/2104.03438 (2021). <https://arxiv.org/abs/2104.03438>
20. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint [arXiv:1708.07747](https://arxiv.org/abs/1708.07747) (2017)
21. Yang, T.J., Chen, Y.H., Sze, V.: Designing energy-efficient convolutional neural networks using energy-aware pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5687–5695 (2017)
22. Zheng, Y., Sun, P., Ren, Q., Xu, W., Zhu, D.: A novel and efficient model pruning method for deep convolutional neural networks by evaluating the direct and indirect effects of filters. Neurocomputing **569**, 127124 (2024) <https://doi.org/10.1016/j.neucom.2023.127124>, <https://www.sciencedirect.com/science/article/pii/S092523122301247X>



# Enhancing Convnets with Pruning and Symmetry-Based Filter Augmentation

Igor Ratajczyk<sup>(✉)</sup> and Adrian Horzyk<sup>(✉)</sup>

AGH University of Krakow, Mickiewicza Av. 30, 30-059 Krakow, Poland  
[ratajczyk@student.agh.edu.pl](mailto:ratajczyk@student.agh.edu.pl), [horzyk@agh.edu.pl](mailto:horzyk@agh.edu.pl)

**Abstract.** Contemporary neural networks usually use rigid or transferred architectures that are not adapted during the training process, which often causes problems with underfitting and overfitting or produces unnecessary large architectures that require complex and long-lasting regularization or implementation of attention blocks. The architecture optimization for a given dataset proceeds in an empirical process, where a few architectures of different hyperparameters are trained and compared to choose the one with the highest performance. When working with convnets, we have a priori set of some filters in different layers to allow the network to learn to represent the most frequent training data patterns to minimize underfitting and overfitting. Establishing a good enough network architecture is still a challenge for developers. In this paper, we propose a new method for augmenting filters during the training process to remove poorly developed or very similar filters and add new filters that better reproduce the frequent patterns occurring in the training data. The presented approaches remove unnecessary bias and harmful inferences produced by low-quality filters developed automatically during casual training. Moreover, this method can also reduce the network size, utilize filters more efficiently, reduce computational costs, accelerate the training process, and achieve better generalization in the same number of epochs as the models that do not use the presented approach.

**Keywords:** CNN structure reduction · filter augmentations · filter pruning · filter grafting · network size reduction · architecture adaptation

## 1 Introduction

In recent years, edge devices using convnets have rapidly developed, requiring a significant reduction in calculations and power consumption while maintaining

---

We gratefully acknowledge the funding support by program “Excellence initiative—research university” for the AGH University in Krakow as well as the ARTIQ project: UMO-2021/01/2/ST6/00004 and ARTIQ/0004/2021.

model performance. Moreover, many contemporary models and solutions consume too much energy and need too many resources for developers to use them freely and cost-effectively.

Denil et al. [6] have shown that neural networks can be over-parameterized, having redundant connections. Among others, pruning and quantization methods have been developed to address overparameterization in neural networks. Pruning and quantization are not exclusive, as has been shown by Wang et al. [33] who used it simultaneously. Another approach is Neural Architecture Search, which generally utilizes Reinforcement Learning [18] or Evolutionary Algorithms [24]. The main drawback of this approach is that it requires learning multiple architectures, which requires many computational resources.

The Lottery Ticket Hypothesis [9, 10] suggests that sparse subnetworks can achieve similar or better accuracy than the original network. This has led to various techniques that aim to identify these subnetworks and prune the original networks accordingly. The detailed study of simple scalar patterns in winning Lottery Ticket Hypothesis has been described by Zhou et al. [35]. However, most heuristics do not use the training-data-driven information available during the training process, even if it could increase the convergence rate of the network and speed up the training.

Grafting is a technique that complements pruning methods, where existing filters are supplemented by adding new ones, often by replacing the pruned filters. The developed grafting methods [19, 29] can barely increase the number of filters because they are usually based on the replacement of pruned filters.

After conducting a comprehensive analysis of filters in pre-trained models often used for transfer learning [15], it has been discovered that some filters exhibit a certain degree of symmetry to other filters. This finding suggests that incorporating this symmetry into models could improve their performance. It could also lead to new methods that take advantage of the symmetries to enhance the efficiency of pre-trained models. Despite vast research in machine learning, this particular aspect has not been thoroughly investigated and used in existing models.

Standard filters are typically initialized randomly. When adding new filters to any neural network layer, the problem of initializing the filters in the next layer arises. The initialization of these filters should consider the information about the augmentation used to generate these new filters. It is challenging because the next layer's filters depend on the previous layer's filters, so they should be sensibly initialized. Despite the importance of this initialization process, it has not been thoroughly investigated in the literature. In this paper, we provide research results in this area to further improve the performance of convnets, simultaneously decreasing their size.

We aim to address the following issues:

- utilizing knowledge available at the training stage of a model to accelerate the learning process of convnets,
- temporal increase of the number of filters to verify whether grafted filters are useful,

- weight initialization in the layer following the augmented one to increase convergence and reduce the disruptive effect of increasing the filters for the subsequent layers.

Our proposed solution consists of:

- filter augmentation by grafting, i.e., by populating filters via pre-defined symmetries to increase the final performance of a model,
- checking whether augmented filters represent features present in the training dataset,
- conjugated augmentations for the layer that follows the augmented one.

The main contribution of this paper is to show that we can obtain better performance of models for test datasets, reducing the overfitting and the number of parameters (weights) by almost 50%.

The second section provides an overview of related state-of-the-art (SOTA) methods for pruning and grafting. The third section discusses the symmetries in the convolutional layers of models dedicated to transfer learning. The fourth section addresses a predicate for pruning and grafting, followed by introducing the idea of conjugated augmentations. The fifth section covers the experimental results of the proposed approach and their comparisons. The last section provides conclusions.

## 2 Related Works

Pruning methods are used to reduce the size of neural networks and computational and memory requirements without compromising their accuracy by eliminating redundant filters, neurons, or connections that have little impact on inference results. Several pruning methods have been proposed in the literature, including weight pruning, filter pruning, and neuron pruning [21]. Weight pruning involves removing individual weights that have little impact on the model's performance. Filter (resp. neuron) pruning involves removing entire filters (resp. neurons) that have a minor influence on the model's performance. These methods can also be combined to achieve better results than using them separately. Considering the moderate pruning rate, it has been shown that pruning can sometimes increase the accuracy of the model [3].

Another approach to improve the performance of convnets by autobalanced filter pruning is an iterative approach that pre-trains the network in an innovative autobalanced way to enable efficient inference [7]. In [23], the filter restoration approach involves restoring or reactivating unimportant or redundant filters in the neural network to improve its performance. Structured pruning in convolutional layers at the level of feature maps and kernels has also been proposed to prune with regularity within the network in [21].

Convnets pruning especially addresses convolutional layers that contribute the most to model computational complexity. Pruning methods are divided by

the structure size they aim to prune. Unstructured pruning involves removing connections (weights) between neurons. One of the most popular unstructured pruning methods is magnitude-based pruning, which removes connections based on their magnitude. This method has proven efficient with the order-of-magnitude number of parameters reduction [8, 12]. Although unstructured pruning manages to reduce overfitting, it does not reduce the computational complexity of a model.

Another method is structured pruning, which involves removing entire neurons or filters from the network [21]. The main advantage over unstructured pruning is that it can efficiently reduce the computational complexity and inference time of pruned models. The results achieved by Li [17] show that structured pruning can effectively mitigate the FLOP of a model by 30%.

Pruning methods may aim to prune a fraction of the total number of weights in the entire network (global pruning) or a fraction of weights in every layer (local pruning). Global pruning has been studied by Tanaka et al. [28], resulting in a thorough investigation of a problem called layer collapse and a proposal of a method to completely avoid it. Pruning methods differ in the heuristics on which the pruning decision is made. These heuristics can be divided into model-based or data-based. One of the most popular model-based heuristics is based on the magnitude of weights for both unstructured [12] and structured pruning [17]. It is based on the assumption that the magnitude of weight of the connections or neurons determines their usefulness. Modern approaches to this heuristic utilize Selective Weight Decay [31].

Data-based pruning heuristics have emerged to address the issue of pre-trained models containing filters that may be inappropriate for a given task. One of the most popular data-driven heuristics is based on a gradient of parameters, as investigated by Molchanov et al. [20]. This approach calculates only the most essential coefficients to deal with the disadvantage of large datasets regarding memory resources and inference time. However, data-based methods may be problematic in large data sets because of their ample memory resources and a long inference time. Some approaches deal with this disadvantage by simplifying the problem. In the Molchanov approach [20], only the most essential coefficients are calculated. Not all developed methods require access to information available only on learning, and pruning can be performed on the already-trained models.

The opposite idea of grafting layers has also been explored. Grafting layers involve replacing less meaningful filters with better ones using an entropy-based criterion, which has been proven to increase the performance of a model by Meng et al. [19]. Thang et al. [29] extended the idea of grafting by simultaneously learning two networks and combining their results. In this approach, the number of filters remains constant, and the less meaningful filters are changed by noising them with better ones in terms of entropy. Filter grafting is a promising approach to improve the performance of convnets by assigning new weights to inactive filters to reactivate them [4]. This one-stage learning method maintains the same number of layers and filters within each layer, resulting in a higher representation

capability of the grafted network. None of those above-mentioned approaches increases the number of filters - even temporarily.

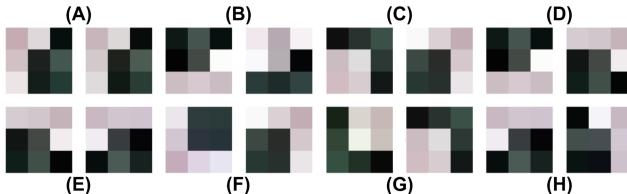
### 3 Analysis of Filter's Symmetry in Convnets

Convolutional Neural Networks (CNNs or convnets) are widely used in Computer Vision problems for pattern recognition. In convolutional layers, the basic patterns are recognized in the first layers of the CNN, while more complex patterns are found in the subsequent layers. Patterns are said to be invariant in shift, rotation, and scale. Due to convolution properties, the same patterns can be found invariantly in shifts. One of the solutions to this problem is to compute the kernels in the log-polar coordinate space, where rotations and scalings in the Cartesian coordinate system are equivalent to shifts. Su and Wen have studied this approach [27]. Another approach proposed by Worrall et al. [34] is to use complex circular harmonics to achieve 360° rotation equivariance.

#### 3.1 Filter Augmentation

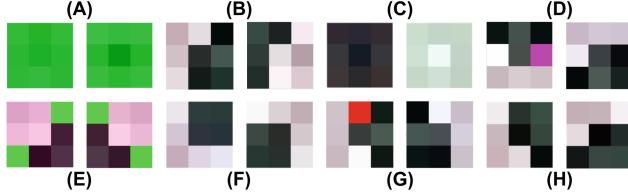
Symmetries can be found in the convolutional layers of modern CNN backbones, which are used for transfer learning. Figure 1 compares the exemplary filters and their symmetries found in the first convolutional layer of the VGG16 model [26]. Figure 2 depicts the chosen symmetry-related filters in the VGG19 model [26].

Each of these models was imported from TensorFlow [2] with ImageNet [5] weights. The symmetry score is measured by (2) with a score of at least 0.95.



**Fig. 1.** Exemplary filters and their symmetrical pair of the first convolutional layer of the VGG16 trained on the ImageNet dataset from TensorFlow.

In Figs. 1 and 2, pairs (A) represent nearly identical filters, that is, they are transformed through  $Id : f \rightarrow f$  with some additional noise, and pairs (B) and (C) are examples of negated filters  $\mathcal{A}_N : f \rightarrow -f$ . Such symmetries are most common in VGG16. Pair (D) shows a vertical flip  $\mathcal{A}_V$ , (E) presents a horizontal flip  $\mathcal{A}_H$ , (F) and (G) show 180° rotations  $\mathcal{A}_P$ . Filters (H) represent 90° rotation  $\mathcal{A}_R$ . There is no order in filters, meaning that shuffling them (with the proper shuffling of channels in the next layer) does not influence the network output; therefore, we do not specify whether the rotation is counter- or clockwise. No shown symmetry has a similarity score (2) equal to 1; all of them are only



**Fig. 2.** Exemplary filters and their symmetrical pair of the first convolutional layer of the VGG19 trained on the ImageNet dataset from TensorFlow.

approximated. The most common basic symmetries in CNN layers are horizontal and vertical flips along the negation of weights.

Let  $f^{[l]}$  denotes filter (1) that is a  $W^{[l]} \times H^{[l]} \times C^{[l]} \times N^{[l]}$  tensor of weights in the  $l$ th CNN layer, and  $f_{wh}^{[l]}[c, n](t)$  represents a single weight at width  $w$ , height  $h$ , in channel  $c$  of filter  $n$  at iteration  $t$ :

$$f^{[l]} \in \mathbb{R}^{W^{[l]} \times H^{[l]} \times C^{[l]} \times N^{[l]}} \quad (1)$$

**Table 1.** Number of pairs of similar filters in VGG16/VGG19/MobileNet via particular symmetries.

Thresholds					
Symmetry	0.80	0.90	0.95	0.97	0.99
Identity	73/74/15	21/23/3	5/4/3	1/2/3	1/1/1
Negation	108/100/2	47/48/1	23/20/0	12/13/0	4/5/0
H flip	78/81/16	35/31/3	9/6/3	3/3/2	0/0/1
V flip	88/85/15	35/37/3	14/15/3	7/5/3	0/1/0
90° rotation	96/92/18	35/39/4	16/15/3	8/10/3	1/1/0
180° rotation	87/83/15	31/35/4	12/12/3	5/5/2	0/0/0

Table 1 presents several pairs of filters whose similarity measure  $s$  according to (2) is greater than the specified similarity thresholds  $s$ .  $F_p^{[l]}$  is a  $L_2$  norm of the  $p$ th filter of the  $l$ th layer. Due to normalization, the similarity score (2) is bounded in the interval  $s \in [-1, 1]$ . Significantly lower values for a MobileNet are caused by a lower number of possible pairs since the first layer of the MobileNet consists of 32 filters instead of 64; therefore, possible pairs are reduced by a factor of 4 (as the number of pairs scales with the number of filters squared).

$$s_{mn}^{[l]} = \sum_{w=0}^{W^{[l]}} \sum_{h=0}^{H^{[l]}} \sum_{c=0}^{C^{[l]}} \frac{f_{wh}^{[l]}[c, m] \cdot f_{wh}^{[l]}[c, n]}{F_n^{[l]} \cdot F_m^{[l]}} \quad (2)$$

We can notice that filters are grouped into classes where every pair of filters can be obtained from others via augmentation or its composition with a certain threshold, for example  $Id$ ,  $\mathcal{A}_N$ ,  $\mathcal{A}_V$ ,  $\mathcal{A}_H$ ,  $\mathcal{A}_P$ ,  $\mathcal{A}_R$ . Figure 1 presents two classes of them: pairs (A), (B), (D), (E), and (H) represent one of them and (C), (F), and (G) the second. For these, symmetry is measured as (2), and the threshold  $s$  equals 0.95.

Contrary to data augmentation [16, 25], filter augmentation does not require data to be invariant to applied symmetries. We assume that some - possibly drop or latent - features are symmetrical, e.g., OCR letters, in general, do not possess horizontal flip symmetry, but when CNN learned how to recognize “ $>$ ” composition, we suppose it may also be helpful to acknowledge the “ $<$ ” sign. Filter augmentation is not exclusive to data augmentation; they may proceed simultaneously.

## 4 Symmetry-Based Grafting with Pruning

The proposed approach aims to improve the performance of convnets by supplementing pruning algorithms with an additional augmentation of filters in convolutional layers, called filter grafting. The problem with current filter initialization methods, such as Xavier [11] or He [14], is that they randomly sample weights from certain distributions, and it is a matter of luck that the filters represent important patterns of features at the beginning of the training process. Some filter weights may be initialized by too small or too many negative numbers and do not win in competitions at max-pooling layers or never achieve positive values for ReLu activation functions, which makes them represent no pattern except noise. This can hurt the performance of convnets. Some other filters can develop representations of similar or almost identical patterns, limiting the training capacity and negatively impacting the constructed convnet performance. Unsuitable developed filters and the too-small number of filters limit the network’s ability to represent essential training data patterns, thus making the network either blind to some input patterns or incorrectly working in some cases, significantly reducing its performance. Therefore, the often-used transfer learning, which begins the training process from the already developed filters, replaces the initial random initialization of weights, which usually accelerates convergence. However, the success of utilizing transfer learning is based on the well-developed filters in the transferred models.

In this paper, we analyze the problems with filter training and propose a new method to use the capacity of the convnets more efficiently and substantially reduce their size, achieving the same or better performance. We claim that entirely random initialization of initial weights can lead to the reduction of the training capacity of the network, so this process should be controlled smartly using the techniques presented in this paper.

## 4.1 Pruning

The predicate or heuristic for deciding whether to prune a given filter is made by comparing the  $L_2$  norm of the filters  $F$  with a dynamically calculated threshold. As the norm between convolutional layers may differ significantly, we propose establishing a threshold by measuring the  $q$  quantile of the  $L_2$  norm within a layer. This approach in the literature is known as local pruning. Also, since we pruned the entire filters, it may be classified as structured pruning.

We propose a heuristics  $P$  for pruning to be defined as (3). Filters to satisfy this equation are pruned.

$$P(n|q) : F_n^{[l]} < \text{quantile}_q(\{F_m^{[l]}\}_{m=0}^{N^{[l]}}) \quad (3)$$

Pruning the number of the convolutional layer  $l$  affects the shape, i.e. several channels  $C^{[l+1]}$  of the next layer  $l + 1$ . After flattening, the output of the last convolutional layer is usually treated as input to the first dense layer; therefore, the last convolutional layer is not pruned itself, but only proper channels are pruned, not the filters.

## 4.2 Grafting via Filter Augmentation

During the pruning stage, the grafting stage is performed. It consists of filter augmentation according to the pre-declared symmetries - e.g., negation  $\mathcal{A}_N$ , horizontal flip  $\mathcal{A}_H$ , rotation  $\mathcal{A}_R$ , etc. Augmentation of certain filters does not replace the source one, but rather appends the filter to a total collection while the source filter remains unchanged.

Similarly to pruning, in filter grafting, layer  $l$  affects the shape of the next layer  $l + 1$ , but the grafting creates new weights. They must be initialized to enhance the inference and stability of the learning process. We will discuss this problem later.

After multiplying the filter, we rank them according to importance - measured by their norm  $F$ . After that, they are sorted. First, all base filters are sorted in decreasing order with respect to  $F$ , and only then are the augmented filters, which are also sorted in decreasing order with respect to  $F$  of the source filters, appended.

It may turn out that we received pairs of similar filters - the measure of similarity is defined by (2) - in the case where the similarity between them is greater than the given threshold  $s$ , the one with lower value of  $F$ . The filter selection process was inspired by the Non-Maximal Suppression (NMS) algorithm [22].

We introduce the parameter  $\alpha \geq 0$  to limit the number of filters after the grafting stage. If  $N^{[l]}$  was the number of filters before grafting, then after this process, the number of filters will not exceed  $(1+\alpha) \cdot N^{[l]}$ . In the case of  $\alpha = 0$ , the grafting stage removes similar filters and replaces them with their augmentations. To maintain stability of the training process, the filters that have been created in this iteration have to be multiplied by a small number  $\epsilon$ . Such an initialization allows one to reject the filter immediately at the first pruning after the grafting

stage if its norm has not been increased more than the norm of the other filters in the layer.

The proposed  $\epsilon$ -initialization helps to check the performance of a newly created filter through augmentation. If the augmented filter does not represent any particular pattern present in the dataset, its activation is lower than other filters; thus, its change is minor after a few iterations (training epochs). Alternating grafting and pruning stages will eliminate these unsuccessfully augmented filters. After some iteration, they may be augmented again after the source filter has changed.

The grafting algorithm is described in the pseudocode of Algorithm 1. The similarity score is measured by (2). The implementation of conjugated augmentations is covered in the following subsection.

### 4.3 Conjugated Augmentation

We introduce a new concept of conjugated augmentations  $\mathcal{A}^\dagger$  - by which we mean augmentation applied to the kernels in layer  $l + 1$  while channels in layer  $l$  were augmented via  $\mathcal{A}$ .

We performed our research using VGG16 of Tensorflow on Imagenet weights. First, we look for pairs in layer  $l$ :  $n$ th and  $m$ th filters that are augmented to each other through augmentations defined in Table 2 with similarity measure  $s_{mn}^{[l]}$  (2) greater than 0.95.

Then we compare whether there exists (conjugated) augmentation such that channels  $n$  and  $m$  (respectively  $\mathcal{A}^\dagger(f^{[l+1]}[n, \eta])$  and  $f^{[l+1]}[m, \eta]$  for all  $w \in W^{[l]}, h \in H^{[l]}, \eta \in N^{[l]}$ ) (5) tend to have a similarity score (2) greater than given threshold  $s$ . The results collected for the exemplary filters in the first layer of VGG16 on ImageNet are presented in Fig. 3 where the histograms of  $s_{mn}^{[l+1]}$  of the filters and their proposed augmentations for conjugations in the second layer are shown. Each histogram shows the distribution density, i.e., the area under the plot sums up to 1.

We conclude that only the negation  $\mathcal{A}_N$  introduces non-trivial conjugated augmentation (other than identity) while  $\mathcal{A}_V$ ,  $\mathcal{A}_{R+}$ , and  $\mathcal{A}_{R-}$  usually require  $\mathcal{A}^\dagger = Id$ . The equality  $\mathcal{A}^\dagger = Id$  suggests that a given feature of the previous layer represented by the  $n$ th filter and its augmentation: the  $m$ th filter (4) is a feature that is invariant to augmentation  $\mathcal{A}$ . Our proposed augmentations and their conjugated pairs are presented in Table 2. Although for  $\mathcal{A}_V$ , it is possible to set  $\mathcal{A}_V^\dagger = \mathcal{A}_V$ , we propose  $\mathcal{A}_V^\dagger = Id$  as this conjugation, which seems to be more popular.

$$\forall(w, h, c) \in W^{[l]} \times H^{[l]} \times C^{[l]} f_{wh}^{[l]}[c, m] = \mathcal{A}(f_{wh}^{[l]}[c, n]) \quad (4)$$

$$\forall(w, h, \eta) \in W^{[l]} \times H^{[l]} \times N^{[l]} f_{wh}^{[l+1]}[m, \eta] = \mathcal{A}^\dagger(f_{wh}^{[l+1]}[n, \eta]) \quad (5)$$

The augmentation in layer  $l$  requires the proper expansion and initialization of the weights of the  $l + 1$  layer. The newly created weights in layer  $l + 1$ , arising from the augmented filter according to (5) are initialized by coping and

**Algorithm 1.** Algorithm for layer grafting

---

**Parameters:** $s$  - similarity threshold $\alpha$  - maximal number of output filters regulizer**Inputs:** $f^{[l]}$  - convolutional layer tensor $b^{[l]}$  - bias $l_0$  - reference to the convolutional layer. $b_0$  - reference to bias $I$  - a collection of indices of preserved filters from the previous layer $A$  - list of conjugated augmentations for channels**Outputs:** $l_p$  - pruned convolutional layer tensor $b_p$  - pruned bias $J$  - a collection of indices of preserved filters $A$  - list of conjugated augmentations for the next layer

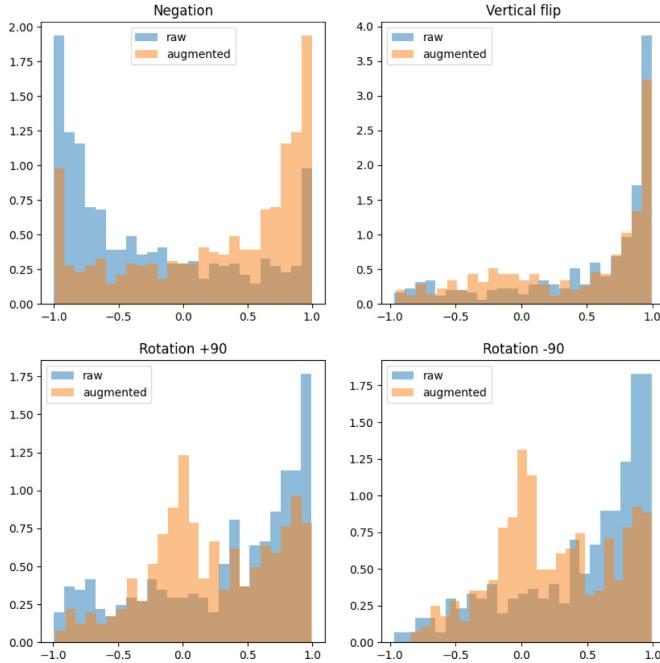
```

1: proceed conjugated augmentation and pruning for layers based on  $I$  and  $A$ 
2:  $K :=$  indices of  $f^{[l]}$  sorted with respect to  $\mu^{[l]}$  decreasing
3:  $f^* :=$  augmented tensor  $f^{[l]}$ 
4:  $f_F := f^{[l]}$  concatenated with  $f^*$ 
5:  $K^* := K$  concatenated with indices source filters of  $f^*$ 
6: active := array of length of  $I_F$  initialized with true
7:  $B :=$  empty list for the indices of filters to be preserved
8: for ( $i := 0, i < K^*.length, i++$ ) do
9:   if active[i] then
10:     $B.add(i)$ 
11:    if  $B.length \geq (1 + \alpha) \cdot N^{[l]}$  then
12:      break
13:    end if
14:    for ( $j := i+1, j < K^*.length, j++$ ) do
15:       $s_{ij}^{[l]} :=$  similarity score of  $i$ th and  $j$ th filter
16:      if active[j] and  $s_{ij}^{[l]} > s$  then
17:        active[j] := false
18:      end if
19:    end for
20:  end if
21: end for
22: update filters tensor and bias according to  $B$ 
23: generate conjugated augmentations for channels in the next layer
24:  $J := B$ 

```

---

augmenting by conjugated augmentation  $\mathcal{A}^\dagger$  the weights of the  $n$ th filter as shown in (5).



**Fig. 3.** Chosen augmentations and pairs of their proposed conjugations ( $Id$  and the best chosen augmentation) and their histograms.

**Table 2.** Chosen Augmentations and their proposed conjugations

Augmentation $\mathcal{A}$	Form	Conjugated
		Augmentation $\mathcal{A}^\dagger$
Negation	$\mathcal{A}_N : f \rightarrow -f$	$\mathcal{A}_N$
V flip	$\mathcal{A}_V : f_{wh} \rightarrow f_{(-w)h}$	$Id$
+90° rotation	$\mathcal{A}_{R+} : f_{wh} \rightarrow f_{h(-w)}$	$Id$
-90° rotation	$\mathcal{A}_{R-} : f_{wh} \rightarrow f_{(-h)w}$	$Id$

## 5 Experimental Results and Comparisons

All numerical tests were performed using NVIDIA RTX A6000 with 36471 MB memory. Computing capability: 8.6. Loaded cuDNN version 8600. Operating system: EndeavourOS, programming language: Python 3.10.9 [32], libraries: NumPy [13] version 1.23.5, TensorFlow version 2.12.0, TensorFlow Datasets [1] version 4.9.2. Besides one hot encoding, no data preprocessing was performed. Flowers [30] datasets were used. As all datasets are well-balanced, validation accuracy is used as performance measurement (Table 3).

**Table 3.** Comparison model size and performance across different architectures on Flowers Dataset. Reference models have been marked with \*.

Model	Pruning $Q$ (5/15/25)	Grafting $\alpha$ (10/20)	No of Param	Val Acc
VGG16*	-	-	1.5E7	68%
VGG16	0.1 / 0.1 / 0.1	0.1 / 0.2	1.4E7	74%
VGG16	0.5/0.2/0.2	0.1 / 0.2	<b>3.0E6</b>	<b>78%</b>
VGG19*	-	-	2.1E7	74%
VGG19	0.5/0.2/0.2	0.1/0.2	<b>3.2E6</b>	75%
VGG19	0.2/0.2/0.05	0.05/0.05	7.2E6	<b>76%</b>

Column Pruning Q should represent the quantiles used in the pruning process, e.g., quantiles 0.1/0.2/0.3 at epochs (5/15/25) should be interpreted as:  $q(5) = 0.1$ ,  $q(15) = 0.2$ , and  $q(25) = 0.3$ . A similar interpretation should be applied for the grafted column. Depending on the optimization target, the algorithm parameters should be adjusted accordingly. Higher quantile values result in a greater reduction in the size of the output model.

## 6 Conclusions

In this paper, we have demonstrated that filter pruning is possible and gives the same or better performance and results, indicating that we can effectively reduce filters without losing the quality and high performance of the solution. We have:

- investigated symmetries in filters in the VGG16, VGG19, and MobileNet networks and how they affect the following convolutional layer,
- introduced the idea of filter augmentation via symmetry and conjugated augmentation as a method for channel initialization in the layer following the augmented one.

In summary, the results presented show that a better performance of a model by 10% for a test dataset can be obtained by implementing the proposed method. We can also reduce the number of weights by nearly 80% simultaneously.

## References

1. TensorFlow Datasets, a collection of ready-to-use datasets. <https://www.tensorflow.org/datasets>
2. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). Software available from tensorflow.org
3. Blalock, D., Gonzalez Ortiz, J.J., Frankle, J., Guttag, J.: What is the state of neural network pruning? Proc. Mach. Learn. Syst. **2**, 129–146 (2020)
4. Cheng, H., et al.: Filter grafting for deep neural networks: reason, method, and cultivation. arXiv preprint [arXiv:2004.12311](https://arxiv.org/abs/2004.12311) (2020)

5. 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)
6. Denil, M., Shakibi, B., Dinh, L., Ranzato, M.A., de Freitas, N.: Predicting parameters in deep learning. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc. (2013)
7. Ding, X., Ding, G., Han, J., Tang, S.: Auto-balanced filter pruning for efficient convolutional neural networks. In: AAAI. AAAI Press, Palo Alto, California, USA (2018)
8. Fernandes, F.E., Jr., Yen, G.G.: Pruning deep convolutional neural networks architectures with evolution strategy. Inf. Sci. **552**, 29–47 (2021)
9. Frankle, J., Carbin, M.: The lottery ticket hypothesis: finding sparse, trainable neural networks. arXiv preprint [arXiv:1803.03635](https://arxiv.org/abs/1803.03635) (2018)
10. Frankle, J., Carbin, M.: The lottery ticket hypothesis: finding sparse, trainable neural networks. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. OpenReview.net (2019)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, vol. 9, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR (2010)
12. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Cortes, C., Lawrence, N.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28. Curran Associates, Inc. (2015)
13. Harris, C.R., et al.: Array programming with NumPy. Nature **585**(7825), 357–362 (2020)
14. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034 (2015)
15. Heaton, J., Goodfellow, I., Bengio, Y., Courville, A.: Deep learning, 800 pp. The MIT Press (2016). ISBN 0262035618. Program. Evol. Mach. **19**(1–2), 305–307 (2018)
16. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580 (2012). [arxiv:1207.0580](https://arxiv.org/abs/1207.0580)
17. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv, [arxiv:1608.08710](https://arxiv.org/abs/1608.08710) (2016)
18. Mahmud, M., Kaiser, M.S., Hussain, A., Vassanelli, S.: Applications of deep learning and reinforcement learning to biological data. IEEE Trans. Neural Netw. Learn. Syst. **29**(6), 2063–2079 (2018)
19. Meng, F., et al.: Filter grafting for deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6599–6607 (2020)
20. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning (2019)
21. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint [arXiv:1611.06440](https://arxiv.org/abs/1611.06440) (2016)

22. Neubeck, A., Van Gool, L.: Efficient non-maximum suppression. In: 18th International Conference on Pattern Recognition (ICPR'06), vol. 3, pp. 850–855 (2006)
23. Prakash, A., Storer, J., Florencio, D., Zhang, C.: REPR: improved training of convolutional filters. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10658–10667 (2019)
24. Shafiq, S., Ahmed, S., Kaiser, M.S., Mahmud, M., Hossain, M.S., Andersson, K.: Comprehensive analysis of nature-inspired algorithms for Parkinson's disease diagnosis. *IEEE Access* **11**, 1629–1653 (2022)
25. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Seventh International Conference on Document Analysis and Recognition, Proceedings, pp. 958–963 (2003)
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint [arXiv:1409.1556](https://arxiv.org/abs/1409.1556) (2014)
27. Su, B., Wen, J.-R.: Log-polar space convolution for convolutional neural networks. arXiv preprint [arXiv:2107.11943](https://arxiv.org/abs/2107.11943) (2021)
28. Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 6377–6389. Curran Associates, Inc. (2020)
29. Tang, Z., Li, Z., Yang, J., Qi, F.: P GGD: a joint-way model optimization strategy based on filter pruning and filter grafting for tea leaves classification. *Neural Process. Lett.* **54**, 06 (2022)
30. T.T. Team: Flowers, January 2019
31. Tessier, H., Gripon, V., Léonardon, M., Hannagan, A.T., Bertrand, D.: Rethinking weight decay for efficient neural network pruning. *J. Imaging* **8**, 64 (2022)
32. Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)
33. Wang, Y., Lu, Y., Blankevoort, T.: Differentiable joint pruning and quantization for hardware efficiency. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12374, pp. 259–277. Springer, Cham (2020). [https://doi.org/10.1007/978-3-030-58526-6\\_16](https://doi.org/10.1007/978-3-030-58526-6_16)
34. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Harmonic networks: deep translation and rotation equivariance. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5028–5037 (2017)
35. Zhou, H., Lan, J., Liu, R., Yosinski, J.: Deconstructing lottery tickets: zeros, signs, and the supermask. In: Advances in Neural Information Processing Systems, vol. 32 (2019)



# Improved Approximation Algorithms for the Cumulative Vehicle Routing Problem

Jingyang Zhao and Mingyu Xiao

University of Electronic Science and Technology of China, Chengdu, China  
[myxiao@uestc.edu.cn](mailto:myxiao@uestc.edu.cn)

**Abstract.** The Cumulative Vehicle Routing Problem (Cu-VRP) extends the classic Vehicle Routing Problem by incorporating fuel consumption considerations, which are crucial in transportation and logistics. In Cu-VRP with Stochastic Demands (Cu-VRPSD), the demand of each customer is unknown until the vehicle visits it. In this paper, we propose a randomized 2.5-approximation algorithm for splittable deliveries and a randomized 3.5-approximation algorithm for unsplittable deliveries, improving the best-known approximation ratios of 3.25 and 6, respectively. For Cu-VRP, since it is a special case of Cu-VRPSD, our algorithms also improve the best-known approximation ratios of 3.186 and 4, respectively. We apply our algorithms on the benchmark sets and the experimental results show that the quality of our computed solutions is much closer to optimality than the provable approximation ratio.

**Keywords:** Vehicle Routing · Cumulative Cost · Stochastic Demands · Approximation Algorithms

## 1 Introduction

In transportation and logistics, fuel consumption is a significant concern, which could represent as much as 60% of a vehicle's operational costs [27]. In the classic Vehicle Routing Problem (VRP) [9], we are given an edge-weighted complete graph  $G = (V = \{v_0, v_1, \dots, v_n\}, E)$ , where  $v_0$  denotes the depot and the other  $n$  vertices in  $V$  denote  $n$  customers, and the objective is to determine an *itinerary* for a vehicle with a capacity of  $Q \in \mathbb{R}_{>0}$  starting from (and ending at) the depot, fulfilling every customer's demand while minimizing the total traveling distance. However, an important factor that determines the fuel consumption is the total weight of the vehicle [13, 25, 30], i.e., the sum of the weight of the empty vehicle and the weight of the goods being carried by the vehicle. To address this, Kara *et al.* [21, 22] introduced the Cumulative Vehicle Routing Problem (Cu-VRP), which aims to find an itinerary such that the total *cumulative cost* as defined below is minimized.

Let  $a \in \mathbb{R}_{\geq 0}$  represent the cost of moving an empty vehicle per unit distance, and let  $b \in \mathbb{R}_{\geq 0}$  denote the cost of transporting one unit demand of goods per

unit distance. The cumulative cost of moving the vehicle one unit distance with  $d$  demand of goods is defined as  $a+b\cdot d$ . An itinerary to Cu-VRP is a set of tours  $\mathcal{T}$ , where each tour  $T = v_0v_{i_1}\dots v_{i_T}v_0 \in \mathcal{T}$  is a directed cycle containing the depot and some customers. We let  $\sum_{e \in E(T)} w(e) := w(v_0, v_{i_1}) + \dots + w(v_{i_{T-1}}, v_{i_T}) + w(v_{i_T}, v_0)$  denote the sum of the lengths of the edges on  $T$ , and let  $x_{eT}$  denote the carried demand of goods by the vehicle when traveling along  $e$  on  $T$ . The cumulative cost  $Cu(T)$  of  $T$  is defined as

$$Cu(T) := a \cdot \sum_{e \in E(T)} w(e) + b \cdot \sum_{e \in E(T)} x_{eT} \cdot w(e).$$

The cumulative cost of the itinerary  $\mathcal{T}$  is defined to be  $Cu(\mathcal{T}) := \sum_{T \in \mathcal{T}} Cu(T)$ . When  $a > 0$  and  $b = 0$ , Cu-VRP reduces to VRP.

For example, consider a tour  $T = v_0v_1v_2v_0$ , where we let  $e_i = v_{i-1}v_i \bmod 3$  and  $w(e_i) = 1$  for  $i \in \{1, 2, 3\}$ . Assume that the vehicle picks up 8 demand of goods at the depot and delivers 2 demand each to  $v_1$  and  $v_2$  along the tour  $T$ , respectively. We can get  $x_{e_1T} = 8$ ,  $x_{e_2T} = 6$ ,  $x_{e_3T} = 4$ , and thus  $Cu(T) = a \cdot 3 + b \cdot (8 + 6 + 4) = 3a + 18b$ . Note that the item  $b \cdot x_{e_3T} \cdot w(e_3)$ , where  $e_3$  is the last edge of  $T$ , represents the additional carrying cost incurred by the vehicle for goods that are picked up but not delivered during the tour  $T$ .

Cu-VRP has been studied extensively [10, 13, 16]. It is a simplified model for fuel consumption with a linear objective function for VRP [30]. Gaur *et al.* [16] proposed a heuristic algorithm using the column generation method. Wang *et al.* [29] generalized Cu-VRP into the model with multi-depots. Cu-VRP with time windows was investigated in [10], and Cu-VRP with a limitation on the number of used tours was studied in [12, 24]. A recent survey of Cu-VRP can be found in [8].

Due to uncertainty, the demand of each customer  $v_i$  may be represented by an independent random variable  $\chi_i \in [0, Q]$ , and the exact demand is known only when the vehicle visits  $v_i$ . When these conditions are added to VRP (resp., Cu-VRP), this problem is known as VRP (resp., Cu-VRP) with Stochastic Demands (VRPSD (resp., Cu-VRPSD)) [3, 14]. This model has natural applications in logistic where the demands of customers are unknown before routing the vehicle, and early surveys can be found in [4, 17]. Similarly, Cu-VRPSD reduces to VRPSD when  $a > 0$  and  $b = 0$ . In Cu-VRPSD [14], the goal is to devise a policy for constructing a feasible itinerary  $\mathcal{T}$  such that the expected cumulative cost of  $\mathcal{T}$ , denoted as  $\mathbb{E}[Cu(\mathcal{T})]$ , is minimized. Here,  $\mathbb{E}[Cu(\mathcal{T})]$  represents the expectation of the cumulative cost of  $\mathcal{T}$  over all possible sets of customers' demands. Cu-VRPSD is an interesting problem because we do not know the demands of customers in advance and we must carefully manage the vehicle's load, as any unnecessary carrying of goods can increase the cumulative cost of the tours.

For VRP and its variants, there are two common settings: *splittable* and *unsplittable*. In the *splittable* case, each customer can be satisfied by using multiple tours, while in the *unsplittable* case, each customer must be satisfied with one tour. Note that if a tour passes through a customer without delivering any goods, this tour is not considered to have been used to satisfy the customer.

## 1.1 Related Works

We mainly focus on approximation algorithms, which can generate solutions with a theoretical guarantee in polynomial time. For Cu-VRP, an algorithm is called a  $\rho$ -approximation algorithm if it can output a solution  $\mathcal{T}$  with a cumulative cost of  $Cu(\mathcal{T}) \leq \rho \cdot OPT$  in polynomial time, where  $OPT$  is the cumulative cost of the optimal solution and  $\rho$  is called the approximation ratio. For Cu-VRPSD, since the demands of customers are random, the definition slightly changes as follows: an algorithm is called a randomized  $\rho$ -approximation algorithm if it can use a policy to output a solution  $\mathcal{T}$  with an expected cumulative cost of  $\mathbb{E}[Cu(\mathcal{T})] \leq \rho \cdot \mathbb{E}[Cu(\mathcal{T}^*)]$  in polynomial time, where  $\mathcal{T}^*$  is the minimum expected cumulative cost solution obtained by the optimal policy.

We will use  $\alpha$  to denote the current approximation ratio of the metric Traveling Salesman Problem (TSP). Since the Christofides-Serdyukov algorithm [6, 28] is a 1.5-approximation algorithm for metric TSP, and the approximation ratio has been improved to  $1.5 - 10^{-36}$  by Karlin *et al.* [23], we have  $\alpha \approx 1.5$ . Next, we give a brief review of the literature on VRP(SD) and Cu-VRP(SD).

**VRP(SD).** For VRP, Haimovich and Kan [19] proposed an  $(\alpha + 1)$ -approximation algorithm for the splittable case, and Altinkemer and Gavish [1] proposed an  $(\alpha + 2)$ -approximation algorithm for the unsplittable case. Recently, Blauth *et al.* [5] improved the ratio to  $\alpha + 1 - \varepsilon$  for the splittable case, and Frigstad *et al.* [11] improved the ratio to  $\alpha + 1 + \ln 2 - \varepsilon'$  for the unsplittable case, where  $\varepsilon$  and  $\varepsilon'$  are small positive constants related to  $\alpha$ . For VRPSD, Bertsimas [3] gave the first randomized  $(\alpha + 1 + o(1))$ -approximation algorithm for the splittable case and a randomized  $(\alpha + Q)$ -approximation algorithm for the unsplittable case, and Gupta *et al.* [18] improved the approximation ratios to  $\alpha + 1$  and  $\alpha + 2$ , respectively.

**Cu-VRP(SD).** For Cu-VRP, Gaur *et al.* [13] proposed a  $(1 + \frac{4\alpha}{\sqrt{4\alpha^2 + 16\alpha - 2\alpha}})$ -approximation algorithm for the splittable case, and a  $(1 + \frac{4\alpha}{\sqrt{4\alpha^2 + 24\alpha + 4} - (2\alpha + 2)})$ -approximation algorithm for the unsplittable case. For Cu-VRPSD, Gaur *et al.* [14] proposed a  $2(1 + \alpha)$ -approximation algorithm for the splittable case, and a 7-approximation algorithm for the unsplittable case, and later, they [15] further improved these ratios to  $\max\{1 + \frac{3}{2}\alpha, 3\}$  and  $\max\{2 + \frac{3}{2}\alpha, 6\}$ , respectively. Since  $\alpha \approx 1.5$ , the current-best approximation ratios for splittable Cu-VRP and unsplittable Cu-VRP are 3.186 and 4, respectively [13], while the current-best approximation ratios for splittable Cu-VRPSD and unsplittable Cu-VRPSD are 3.25 and 6, respectively [15].

## 1.2 Our Results

We design improved approximation algorithms for Cu-VRPSD and Cu-VRP.

For Cu-VRPSD, we propose a randomized  $(\alpha + 1)$ -approximation algorithm for the splittable case, and a randomized  $(\alpha + 2)$ -approximation algorithm for the unsplittable case. These significantly improve upon the best-known randomized  $\max\{1.5\alpha + 1, 3\}$ -approximation algorithm and  $\max\{1.5\alpha + 2, 6\}$ -approximation

algorithm in [15], respectively. Note that our results even match the best-known approximation ratios for VRPSD in [18].

For Cu-VRP, since it is a special case of Cu-VRPSD, we immediately obtain a randomized  $(\alpha + 1)$ -approximation algorithm for the splittable case, and a randomized  $(\alpha + 2)$ -approximation algorithm for the unsplittable case. These improve upon the best-known approximation ratios of  $1 + \frac{4\alpha}{\sqrt{4\alpha^2 + 16\alpha - 2\alpha}}$  and  $1 + \frac{4\alpha}{\sqrt{4\alpha^2 + 24\alpha + 4} - (2\alpha + 2)}$  in [13], respectively. Moreover, we will show that for Cu-VRP, the algorithms can be derandomized in polynomial time while preserving the approximation ratios.

A summary of our approximation ratios under  $\alpha = \frac{3}{2}$  can be found in Table 1.

Finally, we apply our algorithms to the benchmark sets in [16], compare our results with previous results, and compute our gaps to optimality using the provided lower bounds. Experimental results show that the quality of our computed solutions is much closer to optimality than the provable approximation ratio suggests.

**Table 1.** A summary of the previous approximation ratios and our approximation ratios for Cu-VRPSD and Cu-VRP under  $\alpha = 1.5$

Problem	Previous Result		Our Result	
	Splittable	Unsplittable	Splittable	Unsplittable
Cu-VRPSD	3.25 [15]	6 [15]	<b>2.5</b>	<b>3.5</b>
Cu-VRP	3.186 [13]	4 [13]	<b>2.5</b>	<b>3.5</b>

Due to limited space, the proofs of lemmas and theorems marked with “\*” were omitted and they can be found in the full version of this paper.

## 2 Notations

In Cu-VRP or Cu-VRPSD, we use  $G = (V = \{v_0, v_1, \dots, v_n\}, E)$  to denote the input complete graph. There is a non-negative cost function  $w : E \rightarrow \mathbb{R}_{\geq 0}$  on the edges, where  $w(u, v)$  denotes the length of edge  $uv \in E$ . We assume that  $w$  is a *metric* function, i.e., it is symmetric and satisfies the triangle inequality. Each customer  $v$  has a required non-negative demand  $d_v \leq Q$ , where  $Q \in \mathbb{R}_{>0}$  is the capacity of the vehicle, and we let  $l_v := w(v_0, v)$ . For the sake of presentation, we use  $V' := V \setminus \{v_0\}$  to denote the customer set, and let  $l_i := w(v_0, v_i)$ ,  $d_i := d_{v_i}$ , and  $[i] := \{1, 2, \dots, n\}$ . In Cu-VRP,  $d_i$  is given, and we assume that  $d_i > 0$  since we can simply ignore  $v_i$  if  $d_i = 0$ . In Cu-VRPSD,  $d_i$  is given by an independent random variable  $\chi_i \in [0, Q]$ , its value becomes known only when the vehicle visits  $v_i$ , and we assume that the random variable  $\chi_i$  is not identically zero, as in such a case,  $v_i$  can also be ignored. Consequently, any feasible policy must visit every customer at least once, as shown in [18].

For any random variable  $\theta$ , we use  $\theta \sim U[l, r]$  to indicate that  $\theta$  is uniformly distributed over the interval  $[l, r)$ , where  $l < r$ .

A *tour*  $T$ , denoted by  $v_0v_{i_1}\dots v_{i_T}v_0$ , is a sequence of vertices  $v_0, v_{i_1}, \dots, v_{i_T}, v_0$ , where only the first and the last vertices are the same, and each consecutive pair of vertices is connected by an edge. Note that  $T$  can be regarded as a directed simple cycle, and it always contains the depot  $v_0$ . We use  $E(T)$  to denote the set of edges on  $T$ , and  $V'(T)$  to denote the set of customers on  $T$ . We assume that the vehicle carries  $x_{eT}$  demand of goods when traveling along  $e \in E(T)$ . As defined before, the cumulative cost of  $T$  is  $Cu(T) = a \cdot \sum_{e \in E(T)} w(e) + b \cdot \sum_{e \in E(T)} x_{eT} \cdot w(e)$ , where  $w(T) := \sum_{e \in E(T)} w(e)$  is said to be the weight of  $T$ . A *TSP tour* is a tour that visits all customers and the depot exactly once. We use  $\tau$  to denote the weight of the minimum weight TSP tour.

Cu-VRPSD can be described as follows.

**Definition 1 (Cu-VRPSD).** Given a complete graph  $G = (V = \{v_0, v_1, \dots, v_n\}, E)$ , a metric function  $w$ , a vehicle capacity  $Q \in \mathbb{R}_{>0}$ , random demand variables  $\chi_i$  for customers  $v_i$ , and two parameters  $a \in \mathbb{R}_{\geq 0}$  and  $b \in \mathbb{R}_{\geq 0}$ , we need to design a policy to find a feasible itinerary  $\mathcal{T}$  with minimized  $\mathbb{E}[Cu(\mathcal{T})]$  such that

- the vehicle carries at most  $Q$  demand of goods on each tour  $T \in \mathcal{T}$ ,
- the vehicle delivers goods to customers only in  $V'(T)$  on each tour  $T \in \mathcal{T}$ ,
- the sum of the delivered demand over all tours for each  $v_i \in V'$  equals  $\chi_i$ .

As mentioned, in splittable Cu-VRPSD, each customer is allowed to be satisfied by using multiple tours, and in unsplittable Cu-VRPSD, each customer must be satisfied by using only one tour. Under the assumption that the demand variable  $\chi_i$  is not identically zero, we are not required to know its distribution.

In Cu-VRP, we have  $\chi_i = d_i$  for each  $v_i \in V'$ , and  $d_i$  is known in advance.

To analyze our algorithms, we will use the following known lower bound.

**Lemma 1 ([15]).** For splittable or unsplittable Cu-VRPSD, we have  $\mathbb{E}[Cu(\mathcal{T}^*)] \geq a \cdot \max\{\tau, \frac{2}{Q} \cdot \eta\} + b \cdot \eta$ , where  $\eta := \sum_{i \in [n]} \mathbb{E}[\chi_i] \cdot l_i$ .

When  $b = 0$ , the lower bound in Lemma 1 was used in analyzing approximation algorithms for VRPSD in [18]. When  $a = 0$ , one can easily get that Cu-VRPSD can be solved exactly in polynomial time. So, we assume  $a > 0$ .

### 3 Improved Algorithms for Cu-VRPSD

In this section, we will propose a randomized  $(\alpha + 1)$ -approximation algorithm for splittable Cu-VRPSD, and a randomized  $(\alpha + 2)$ -approximation algorithm for unsplittable Cu-VRPSD. We first consider the splittable case.

**Algorithm 1.** An algorithm for splittable Cu-VRPSD ( $\text{SPLIT}(Q')$ )

---

**Input:** An instance of splittable Cu-VRPSD, and a parameter  $Q' \leq Q$ .

**Output:** A feasible solution to Cu-VRPSD.

```

1: Obtain an  $\alpha$ -approximate TSP tour  $T^*$  using an  $\alpha$ -approximation algorithm for
   metric TSP, orient  $T^*$  in either clockwise or counterclockwise direction, and denote
    $T^* = v_0v_1v_2\dots v_nv_0$  by renumbering the customers following the direction.
2: Load the vehicle with  $L := \theta$  demand of goods, where  $\theta \sim U[0, Q')$ .
3:  $i := 1$ .
4: while  $i \leq n$  do
5:   Go to customer  $v_i$ .
6:   if  $d_i \leq L$  then
7:     Deliver  $d_i$  demand of goods to  $v_i$ , and update  $L := L - d_i$ .
8:      $i := i + 1$ .
9:   else
10:    Deliver  $L$  demand of goods to  $v_i$ , and update  $d_i := d_i - L$ .
11:    Return to the depot, load the vehicle with  $Q'$  demand of goods, and update
         $L := Q'$ .
12:   end if
13: end while
14: Go to the depot.

```

---

### 3.1 The Splittable Case

Similar to the previous algorithm in [15], we propose a tour-partition algorithm for splittable Cu-VRP, denoted as  $\text{SPLIT}(Q')$ , where  $Q' \leq Q$  is the maximum allowed demand for goods to be carried during the vehicle's travel.

$\text{SPLIT}(Q')$  first computes an  $\alpha$ -approximate TSP tour  $T^*$ , which can be oriented in either clockwise or counterclockwise direction. Then,  $\text{SPLIT}(Q')$  ensures that the vehicle satisfies the customers in the order they appear on the TSP tour. The strategy adopted is greedy: the vehicle returns to the depot to reload  $Q'$  demand of goods whenever its current load is less than the demand of the serving customer. Note that the previous algorithm in [15] orients  $T^*$  by choosing one of the two possible orientations uniformly at random, and its analysis heavily relies on this property. However, we will show that this randomness is not necessary by using a new analysis method. Consequently, in  $\text{SPLIT}(Q')$ , we choose one of the two directions arbitrarily. The details of  $\text{SPLIT}(Q')$  can be found in Algorithm 1.

Our improved approximation algorithm for splittable Cu-VRPSD, denoted as Algorithm 1, is shown in Algorithm 2, where we consider two cases to call  $\text{SPLIT}(Q')$ .

**The Analysis.** Note that  $\text{SPLIT}(Q')$  carries  $\theta \sim U[0, Q')$  demand of goods initially. To analyze the expected cumulative cost of its solution  $\mathcal{T}$ , we first analyze the expected cumulative cost of  $\mathcal{T}$  conditioned on the demand realization such that  $\chi_i = d_i$  for each  $i \in [n]$ , denoted by  $\mathbb{E}[Cu(\mathcal{T}) \mid \chi_i = d_i]$ , and then obtain the expected cumulative cost of  $\mathcal{T}$ , i.e.,  $\mathbb{E}[Cu(\mathcal{T})] = \mathbb{E}[\mathbb{E}[Cu(\mathcal{T}) \mid \chi_i]]$ .

**Algorithm 2.** An improved algorithm for splittable Cu-VRPSD (ALG.1)**Input:** An instance of splittable Cu-VRPSD, and a vehicle capacity  $Q$ .**Output:** A feasible solution to Cu-VRP.

- 1: **if**  $2/\alpha \cdot a/b \geq Q$  **then**
- 2:   Obtain a solution  $\mathcal{T}$  by calling  $\text{SPLIT}(Q')$  with  $Q' = Q$ .
- 3: **else**
- 4:   Obtain a solution  $\mathcal{T}$  by calling  $\text{SPLIT}(Q')$  with  $Q' = 2/\alpha \cdot a/b$ .
- 5: **end if**
- 6: Return  $\mathcal{T}$ .

Assume that in  $\text{SPLIT}(Q')$  the vehicle carries  $L_{i-1}$  demand of goods when traveling along the edge  $v_{i-1}v_{i \bmod (n+1)}$  of the TSP tour  $T^*$ , i.e., during the travel after satisfying  $v_{i-1}$  and before satisfying  $v_{i \bmod (n+1)}$ .

**Lemma 2 (\*).** *For any  $i \in [n+1]$ , it holds  $L_{i-1} = \theta + \lceil \frac{\sum_{j=1}^{i-1} d_j - \theta}{Q'} \rceil \cdot Q' - \sum_{j=1}^{i-1} d_j$ . Moreover, we have  $L_{i-1} \sim U[0, Q']$  conditioned on the demand realization.*

**Lemma 3.** *In the solution of  $\text{SPLIT}(Q')$ , the expected cumulative cost conditioned on the demand realization during the vehicle's travel from  $v_{i-1}$  to  $v_i$  is  $a \cdot w(v_{i-1}, v_i) + b \cdot \frac{Q'}{2} \cdot w(v_{i-1}, v_i)$ .*

*Proof.* By Lemma 2, the vehicle carries  $L_{i-1}$  demand of goods during the vehicle's travel from  $v_{i-1}$  to  $v_i$ , where  $L_{i-1} \sim U[0, Q']$ . Hence, we have  $\mathbb{E}[L_{i-1} | \chi_i = d_i] = \int_0^{Q'} \frac{x}{Q'} dx = \frac{Q'}{2}$ . Then, the expected cumulative cost of the vehicle's travel from  $v_{i-1}$  to  $v_i$  conditioned on the demand realization is  $a \cdot w(v_{i-1}, v_i) + b \cdot \mathbb{E}[L_{i-1} | \chi_i = d_i] \cdot w(v_{i-1}, v_i) = a \cdot w(v_{i-1}, v_i) + b \cdot \frac{Q'}{2} \cdot w(v_{i-1}, v_i)$ .  $\square$

By Step 11 in  $\text{SPLIT}(Q')$ , if the vehicle visits  $v_i$  carrying  $L_{i-1} < d_i$  demand of goods, it will first deliver  $L_{i-1}$  demand of goods for  $v_i$ , and then proceed to the depot to reload  $Q'$  demand of goods to deliver the remaining required demand for  $v_i$ . We refer to this process as an *additional visit to  $v_0$* . Note that if  $Q' = Q$  the vehicle can satisfy  $v_i$  using at most one additional visit to  $v_0$ , as after one additional visit to  $v_0$ , the vehicle carries  $Q$  demand of goods and in our setting we have  $d_i \leq Q$ .

**Lemma 4.** *Conditioned on the demand realization, in the solution of  $\text{SPLIT}(Q')$  with  $Q' = Q$ , when delivering goods for  $v_i$ , the vehicle incurs an additional visit to  $v_0$  with a probability of  $\frac{d_i}{Q}$ , and the expected cumulative cost of the additional visit is  $a \cdot \frac{2}{Q} \cdot d_i \cdot l_i + b \cdot d_i \cdot l_i$ .*

*Proof.* By Lemma 2, the vehicle carries  $L_{i-1}$  demand of goods from  $v_{i-1}$  to  $v_i$ , where  $L_{i-1} \sim U[0, Q]$ . Hence, the vehicle incurs an additional visit to  $v_0$  with a probability of  $\Pr[L_{i-1} = x < d_i] = \int_0^{d_i} \frac{1}{Q} dx = \frac{d_i}{Q}$ .

As mentioned, the vehicle can satisfy  $v_i$  using at most one additional visit to  $v_0$  since  $Q' = Q$ . If the vehicle incurs an additional visit to  $v_0$ , by  $\text{SPLIT}(Q')$ , the

vehicle will carry 0 demand of goods from  $v_i$  to  $v_0$  and  $Q$  demand of goods from  $v_0$  to  $v_i$ . So, the cumulative cost of this additional visit will be  $a \cdot 2 \cdot l_i + b \cdot Q \cdot l_i$ . Since  $L_{i-1} \sim U[0, Q]$ , the expected cumulative cost of this additional visit conditioned on the demand realization is  $\int_0^{d_i} \frac{a \cdot 2 \cdot l_i + b \cdot Q \cdot l_i}{Q} dx = a \cdot \frac{2}{Q} \cdot d_i \cdot l_i + b \cdot d_i \cdot l_i$ .  $\square$

**Theorem 1.** For Cu-VRPSD with any  $Q' \in (0, Q]$ , SPLIT( $Q'$ ) generates a solution  $\mathcal{T}$  with an expected cumulative cost of  $\mathbb{E}[Cu(\mathcal{T})] \leq a \cdot (\alpha \cdot \tau + \frac{2}{Q'} \cdot \eta) + b \cdot \frac{Q'}{2} \cdot (\alpha \cdot \tau + \frac{2}{Q'} \cdot \eta)$ .

*Proof.* Recall that  $\mathbb{E}[Cu(\mathcal{T})] = \mathbb{E}[\mathbb{E}[Cu(\mathcal{T}) \mid \chi_i]]$  and  $\eta = \sum_{i \in [n]} \mathbb{E}[\chi_i] \cdot l_i$ . It is sufficient to prove  $\mathbb{E}[Cu(\mathcal{T}) \mid \chi_i = d_i] \leq a \cdot (\alpha \cdot \tau + \frac{2}{Q'} \cdot \sum_{i \in [n]} d_i \cdot l_i) + b \cdot \frac{Q'}{2} \cdot (\alpha \cdot \tau + \frac{2}{Q'} \cdot \sum_{i \in [n]} d_i \cdot l_i)$  for any  $Q' \in (0, Q]$ . Next, we analyze the expected cumulative cost of  $\mathcal{T}$  conditioned on the demand realization.

Firstly, we consider the case that  $Q' = Q$ .

**Case 1:**  $Q' = Q$ . By Lemma 3, the total expected cumulative cost of the vehicle for traveling the edges in the TSP tour  $T^*$  is  $\sum_{e \in E(T^*)} (a \cdot w(e) + b \cdot \frac{Q}{2} \cdot w(e)) = a \cdot w(T^*) + b \cdot \frac{Q}{2} \cdot w(T^*) \leq a \cdot \alpha \cdot \tau + b \cdot \frac{Q}{2} \cdot \alpha \cdot \tau$ , where the inequality follows from  $w(T^*) \leq \alpha \cdot \tau$  since  $T^*$  is an  $\alpha$ -approximate TSP tour.

By Lemma 4, the total expected cumulative cost of the additional visits to  $v_0$  by customers in  $V'$  is  $\sum_{v_i \in V'} (a \cdot \frac{2}{Q} \cdot d_i \cdot l_i + b \cdot d_i \cdot l_i) = \sum_{i \in [n]} (a \cdot \frac{2}{Q} \cdot d_i \cdot l_i + b \cdot d_i \cdot l_i)$ .

Therefore, SPLIT( $Q$ ) generates a solution  $\mathcal{T}$  with an expected cumulative cost of  $\mathbb{E}[Cu(\mathcal{T}) \mid \chi_i = d_i] \leq a \cdot (\alpha \cdot \tau + \frac{2}{Q} \cdot \sum_{i \in [n]} d_i \cdot l_i) + b \cdot \frac{Q}{2} \cdot (\alpha \cdot \tau + \frac{2}{Q} \cdot \sum_{i \in [n]} d_i \cdot l_i)$ .

**Case 2:**  $Q' < Q$ . Note that in Lemma 4 where  $Q' = Q$  we used the property that the demand of each customer is at most  $Q'$ , which may not hold for  $Q' < Q$ . Hence, we cannot directly extend the above result to the case that  $Q' < Q$ . However, we can implicitly split each customer  $v_i$  with  $d_i > Q'$  into a set of customers at the same place, denoted as  $V_i$ , with each having a demand of at most  $Q'$  and all having a total demand of  $d_i$ . Hence,  $l_i = l_v$  for any  $v \in V_i$  and  $d_i \cdot l_i = \sum_{v \in V_i} d_v \cdot l_v$ . Denote the original instance and the new instance by  $I$  and  $I'$ , respectively. The TSP tour in  $I'$ , denoted as  $v_0 p_1 \dots p_n v_0$  where  $p_i$  can be any permutation of customers in  $V_i$ , has the same cost as the TSP tour  $T^* = v_0 v_1 \dots v_n v_0$  in  $I$ . It is easy to see that based on the greedy strategy SPLIT( $Q'$ ) generates two equivalent solutions for  $I$  and  $I'$  with the same cumulative cost. Since the demand of each customer in  $I'$  is at most  $Q'$ , by the previous analysis of the case that  $Q' = Q$  and the proofs of Lemmas 3 and 4, the solution  $\mathcal{T}$  has an expected cumulative cost of  $\mathbb{E}[Cu(\mathcal{T}) \mid \chi_i = d_i] \leq a \cdot (\alpha \cdot \tau + \frac{2}{Q'} \cdot \sum_{i \in [n]} d_i \cdot l_i) + b \cdot \frac{Q'}{2} \cdot (\alpha \cdot \tau + \frac{2}{Q'} \cdot \sum_{i \in [n]} d_i \cdot l_i)$ , which finishes the proof.  $\square$

**Theorem 2.** For splittable Cu-VRPSD, Algorithm 1 is a randomized  $(\alpha + 1)$ -approximation algorithm.

*Proof.* According to Algorithm 1 in Algorithm 2, we have two following cases.

**Case 1:**  $2/\alpha \cdot a/b \geq Q$ . Algorithm 1 calls SPLIT( $Q$ ). By Theorem 1, it generates a solution  $\mathcal{T}$  with an expected cumulative cost of

$$\mathbb{E}[Cu(\mathcal{T})] \leq a \cdot (\alpha \cdot \tau + \frac{2}{Q} \cdot \eta) + b \cdot \frac{Q}{2} \cdot \left( \alpha \cdot \tau + \frac{2}{Q} \cdot \eta \right). \quad (1)$$

We further consider two cases.

**Case 1.1:**  $\tau \leq \frac{2}{Q} \cdot \eta$ . We can get that

$$\begin{aligned} \mathbb{E}[Cu(\mathcal{T})] &\leq a \cdot \left( \alpha \cdot \tau + \frac{2}{Q} \cdot \eta \right) + b \cdot \left( \frac{Q}{2} \cdot \alpha \cdot \tau + \eta \right) \\ &\leq a \cdot \left( \alpha \cdot \tau + \frac{2}{Q} \cdot \eta \right) + b \cdot (\alpha \cdot \eta + \eta) \\ &\leq (\alpha + 1) \cdot \mathbb{E}[Cu(\mathcal{T}^*)], \end{aligned}$$

where the first inequality follows from (1), the second from  $\tau \leq \frac{2}{Q} \cdot \eta$  by the definition of Case 1.1, and the last from Lemma 1.

**Case 1.2:**  $\tau \geq \frac{2}{Q} \cdot \eta$ . Let  $\delta := 1 - \frac{\frac{2}{Q} \cdot \eta}{\tau}$ . If  $\tau = 0$ , we can get  $\eta = 0$  and then  $\mathbb{E}[Cu(\mathcal{T})] \leq (\alpha + 1) \cdot \mathbb{E}[Cu(\mathcal{T}^*)] = 0$  by (1) and Lemma 1. If  $\tau > 0$ , we can get  $0 \leq \delta \leq 1$ . Then, we have

$$\begin{aligned} \mathbb{E}[Cu(\mathcal{T})] &\leq a \cdot \left( \alpha \cdot \tau + \frac{2}{Q} \cdot \eta \right) + b \cdot \left( \frac{Q}{2} \cdot \alpha \cdot \tau + \eta \right) \\ &= a \cdot \left( \alpha \cdot \tau + \frac{2}{Q} \cdot \eta \right) + b \cdot \left( \frac{Q}{2} \cdot \delta \cdot \alpha \cdot \tau + \frac{Q}{2} \cdot (1 - \delta) \cdot \alpha \cdot \tau + \eta \right) \\ &\leq a \cdot \left( \alpha \cdot \tau + \frac{2}{Q} \cdot \eta \right) + a \cdot \delta \cdot \tau + b \cdot \left( \frac{Q}{2} \cdot (1 - \delta) \cdot \alpha \cdot \tau + \eta \right) \\ &= a \cdot ((\alpha + \delta) \cdot \tau + (1 - \delta) \cdot \tau) + b \cdot \left( \frac{Q}{2} \cdot \frac{\frac{2}{Q} \cdot \eta}{\tau} \cdot \alpha \cdot \tau + \eta \right) \\ &= a \cdot (\alpha + 1) \cdot \tau + b \cdot (\alpha + 1) \cdot \eta \\ &\leq (\alpha + 1) \cdot \mathbb{E}[Cu(\mathcal{T}^*)], \end{aligned}$$

where the first inequality follows from (1), the second inequality from  $2/\alpha \cdot a/b \geq Q$  by the definition of Case 1, the second equality from  $\delta = 1 - \frac{\frac{2}{Q} \cdot \eta}{\tau}$ , and the last inequality from Lemma 1.

**Case 2:**  $2/\alpha \cdot a/b \leq Q$ . ALG.1 calls SPLIT( $2/\alpha \cdot a/b$ ). By Theorem 1, it generates a solution  $\mathcal{T}$  with an expected cumulative cost of

$$\begin{aligned} \mathbb{E}[Cu(\mathcal{T})] &\leq a \cdot \left( \alpha \cdot \tau + \frac{2}{2/\alpha \cdot a/b} \cdot \eta \right) + b \cdot \frac{2/\alpha \cdot a/b}{2} \cdot \left( \alpha \cdot \tau + \frac{2}{2/\alpha \cdot a/b} \cdot \eta \right) \\ &= a \cdot \alpha \cdot \tau + b \cdot \alpha \cdot \eta + a \cdot \tau + b \cdot \eta \\ &= a \cdot (\alpha + 1) \cdot \tau + b \cdot (\alpha + 1) \cdot \eta \\ &\leq (\alpha + 1) \cdot \mathbb{E}[Cu(\mathcal{T}^*)], \end{aligned}$$

**Algorithm 3.** An algorithm for unsplittable Cu-VRPSD (UNSPLIT( $Q'$ ))

**Input:** An instance of splittable Cu-VRPSD, and a Capacity  $Q'$ .

**Output:** A feasible solution to Cu-VRP.

```

1: Obtain an  $\alpha$ -approximate TSP tour  $T^*$ , as Step 1.
2: Load the vehicle with  $L := \theta$  demand of goods, where  $\theta \sim U[0, Q')$ .
3:  $i := 1$ .
4: while  $i \leq n$  do
5:   Go to customer  $v_i$ .
6:   if  $d_i \leq L$  then
7:     Deliver  $d_i$  demand of goods to  $v_i$ , and update  $L := L - d_i$ .
8:   else
9:     Return to the depot, load the vehicle with  $d_i - L$  demand of goods, go to
       customer  $v_i$ , and deliver  $d_i$  demand of goods to  $v_i$ .
10:    Return to the depot, load the vehicle with  $L + \lceil \frac{d_i - L}{Q'} \rceil \cdot Q' - d_i$  demand of
        goods, go to customer  $v_i$ , and update  $L := L + \lceil \frac{d_i - L}{Q'} \rceil \cdot Q' - d_i$ .
11:   end if
12:    $i := i + 1$ .
13: end while
14: Go to the depot.

```

where the first inequality follows from (1), and the last inequality from Lemma 1.

Therefore, Algorithm 1 is a randomized  $(\alpha + 1)$ -approximation algorithm.  $\square$

Since Cu-VRP is a special case of Cu-VRPSD where  $\Pr[\chi_i = d_i] = 1$ , we get

**Corollary 1.** *For splittable Cu-VRP, Algorithm 1 is a randomized  $(\alpha + 1)$ -approximation algorithm.*

Although Algorithm 1 is randomized for Cu-VRP, it can be derandomized as follows. Since Algorithm 1 generates a solution based on partitioning an  $\alpha$ -approximate TSP tour, by the above corollary, there exists a partition such that the correspond solution is an  $(\alpha+1)$ -approximate solution for Cu-VRP. Given the TSP tour, based on the dynamic programming approach, the optimal partition with the minimum cumulative cost can be found in  $O(n^2)$  [13]. Hence, to obtain a deterministic  $(\alpha+1)$ -approximation algorithm, we can use the dynamic program to find the optimal partition and then obtain the corresponding solution.

### 3.2 The Unsplittable Case

For the unsplittable case, the vehicle must satisfy every customer within a single tour. The tour-partition algorithm, denoted as UNSPLIT( $Q'$ ), is shown in Algorithm 3. The main idea is to modify SPLIT( $Q'$ ) whiling ensuring that the vehicle carries the same load  $L_{i-1}$  of goods (as defined in SPLIT( $Q'$ ) in Lemma 2) when traveling along each edge  $v_{i-1}v_i$  of the TSP tour. Our improved algorithm for unsplittable Cu-VRPSD, denoted as Algorithm 2, is shown in Algorithm 4.

**Algorithm 4.** An improved algorithm for unsplittable Cu-VRPSD (ALG.2)**Input:** An instance of splittable Cu-VRPSD, and a Capacity  $Q$ .**Output:** A feasible solution to Cu-VRP.

- 1: **if**  $4/\alpha \cdot a/b \geq Q$  **then**
- 2:     Obtain a solution  $\mathcal{T}$  by calling UNSPLIT( $Q$ ).
- 3: **else**
- 4:     Obtain a solution  $\mathcal{T}$  by calling UNSPLIT( $4/\alpha \cdot a/b$ ).
- 5: **end if**
- 6: Return  $\mathcal{T}$ .

**The Analysis.** Assume that in UNSPLIT( $Q'$ ) the vehicle carries  $L'_{i-1}$  demand of goods when traveling along the edge  $v_{i-1}v_{i \bmod (n+1)}$  of the TSP tour  $T^*$ .

**Lemma 5.** *For any  $i \in [n+1]$ , it holds  $L'_{i-1} = L_{i-1}$ , and  $L'_{i-1} \sim U[0, Q')$  conditioned on the demand realization.*

*Proof.* For any  $i \in [n+1]$ , since  $L_{i-1} \sim U[0, Q')$  conditioned on the demand realization by Lemma 2, it is sufficient to prove the equality  $L'_{i-1} = L_{i-1}$ .

Since  $L'_0 = L_0 = \theta$ , the equality holds for  $i = 1$ . Assume  $L'_{i-1} = L_{i-1}$  for some  $i \geq 1$ . According to SPLIT( $Q'$ ) and UNSPLIT( $Q'$ ), the latter algorithm performs differently only if the vehicle visits  $v_i$  with a load of  $L'_{i-1} = L_{i-1} < d_i$ . In SPLIT( $Q'$ ), since the vehicle is based on the greedy strategy, it will go to the depot to reload  $Q'$  demand of goods exactly  $\lceil \frac{d_i - L_{i-1}}{Q'} \rceil$  times, and hence the vehicle will carry  $L_i = L_{i-1} + \lceil \frac{d_i - L_{i-1}}{Q'} \rceil \cdot Q' - d_i$  demand of goods after satisfying  $v_i$ , which is the same as  $L'_i$  in UNSPLIT( $Q'$ ) by Step 10. Hence, it holds  $L'_i = L_i$ . By induction, the equality holds for any  $i \in [n+1]$ .  $\square$

We can continue the analysis in a manner similar to the splittable case. So, the proofs of the following lemmas and theorems were omitted.

**Lemma 6 (\*).** *In the solution of UNSPLIT( $Q'$ ), the expected cumulative cost conditioned on the demand realization during the vehicle's travel from  $v_{i-1}$  to  $v_i$  is  $a \cdot w(v_{i-1}, v_i) + b \cdot \frac{Q'}{2} \cdot w(v_{i-1}, v_i)$ .*

By Steps 9 and 10 in UNSPLIT( $Q'$ ), if the vehicle visits  $v_i$  carrying  $L'_{i-1} < d_i$  demand of goods, it will go to the depot without delivering any good to  $v_i$ , reload  $d_i$  demand of goods, go to satisfy  $v_i$ , return to the depot to reload, and then go to  $v_i$  again. This process involves two additional visits to  $v_0$ .

**Lemma 7 (\*).** *Conditioned on the demand realization, for each customer  $v_i$ , the vehicle incurs two additional visits to  $v_0$  with a probability of at most  $\frac{d_i}{Q'}$ , and the expected cumulative cost is at most  $a \cdot \frac{4}{Q'} \cdot d_i \cdot l_i + b \cdot 2 \cdot d_i \cdot l_i$ .*

**Theorem 3 (\*).** *For Cu-VRPSD with any  $Q' \in (0, Q]$ , UNSPLIT( $Q'$ ) generates a solution  $\mathcal{T}$  with an expected cumulative cost of  $\mathbb{E}[Cu(\mathcal{T})] \leq a \cdot (\alpha \cdot \tau + \frac{4}{Q'} \cdot \eta) + b \cdot \frac{Q'}{2} \cdot (\alpha \cdot \tau + \frac{4}{Q'} \cdot \eta)$ .*

**Theorem 4 (\*).** *For unsplittable Cu-VRPSD, Algorithm 2 is a randomized  $(\alpha + 2)$ -approximation algorithm.*

Similarly, we have the following corollary.

**Corollary 2.** *For unsplittable Cu-VRP, Algorithm 2 is a randomized  $(\alpha + 2)$ -approximation algorithm.*

Algorithm 2 can also be derandomized by using the dynamic programming method.

## 4 Experimental Results

For unsplittable Cu-VRP, Gaur *et al.* [16] proposed a heuristic algorithm using the column generation method, tested it on 89 instances, and also provided high-quality lower bounds for most of these instances by running a fixed timeout of 3 h per instance. For the sake of comparison, we conduct experiments for unsplittable Cu-VRPSD and Cu-VRP, compute our gaps using the lower bounds in [16], and compare our results with those in [16]. Note that we do not use the lower bound in Lemma 1 because, to our knowledge, it is typically applied only in the theoretical analysis of approximation algorithms and is significantly less accurate than the lower bound obtained by the column generation method.

**Instances.** The 89 instances used in [16] are all well-known benchmark sets of VRP and available on the website<sup>1</sup>. The first 74 instances from [2] include 27 instances in the  $A$ -set, 23 instances in the  $B$ -set, and 24 instances in the  $P$ -set. The remaining 15 instances, denoted as the  $E'$ -set, comprise 14 instances from the  $E$ -set in [7] and a unique instance from [26], where each customer has a unit demand. Although these instances were tested only for Cu-VRP, where the demands are given, they can still be used for Cu-VRPSD as long as we do not use the information of the demands in advance. In these instances,  $a$  and  $b$  were set to  $Q$  and 1, respectively, as the two terms in the objective function balance each other out [16]. Most of these instances contain fewer than 100 customers. The detailed information, including the vehicle capacity, number of customers, previous results, and lower bounds, can be found in [16]. Note that [16] also considered the trivial case  $a = 0$  and  $b = 1$ , which can be solved exactly as mentioned, and the special case  $a = 1$  and  $b = 0$ , which reduces to VRP. The results of our algorithms for the latter case can be found in the full version.

**Implementations.** We consider two TSP tours: the first is obtained using the 1.5-approximation algorithm [6, 28], and the second is obtained using the LKH heuristic algorithm [20], a well-known heuristic for TSP that often generates optimal TSP tours for small instances very quickly.

For Cu-VRPSD, we use Algorithm 2 to conduct experiments. In Algorithm 2, for the sake of analysis, we require that if the vehicle incurs two additional visits to  $v_0$  when serving  $v_i$ , it will go to  $v_0, v_i, v_0$ , and  $v_i$  again, respectively. However,

---

<sup>1</sup> <http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances>.

since the vehicle should go to  $v_{(i+1) \bmod (n+1)}$  next, in the experiments we ensure that the vehicle directly goes to  $v_{(i+1) \bmod (n+1)}$  without returning to  $v_i$  again. By the triangle inequality, this can slightly reduce some cumulative cost. Moreover, since Algorithm 2 is randomized, to evaluate its expected performance, we run it 20 times for each instance and take the average of the results as our final result.

For Cu-VRP, since the demands of customers are known, after obtaining an  $\alpha$ -approximate TSP tour in Algorithm 2, we use dynamic programming to compute the optimal partition of the TSP tour and then determine the result corresponding to this optimal partition. The algorithm is denoted as Algorithm 2+. As mentioned, Algorithm 2+ is the derandomized version of Algorithm 2 and can guarantee an approximation of at most  $\alpha + 1$ .

**Table 2.** A summary of the average improvement ratios and the average gaps, where “with APP” (resp., “with LKH”) means that the TSP tour is obtained by the 1.5-approximation algorithm (resp., the LKH algorithm)

Set	ALG.2 with APP		ALG.2 with LKH	
	Avg. Gap		Avg. Gap	
$A$ -set	64.76%		60.71%	
$B$ -set	76.48%		73.81%	
$P$ -set	58.67%		53.43%	
$E'$ -set	80.08%		71.60%	

(a) The results of ALG.2 with APP/LKH for Cu-VRPSD with  $a = Q$  and  $b = 1$

Set	ALG.2+ with APP		ALG.2+ with LKH	
	Avg. Improvement	Avg. Gap	Avg. Improvement	Avg. Gap
$A$ -set	12.45%	11.10%	13.75%	9.41%
$B$ -set	17.82%	11.82%	18.95%	10.29%
$P$ -set	8.71%	11.91%	11.32%	8.69%
$E'$ -set	8.49%	22.00%	13.97%	12.62%

(b) The results of ALG.2+ with APP/LKH for Cu-VRP with  $a = Q$  and  $b = 1$

**Results.** Due to the inherent differences between Cu-VRPSD and Cu-VRP, it is not meaningful to compare our results of Algorithm 2 for Cu-VRPSD with the previous results for Cu-VRP. Moreover, since Cu-VRPSD is harder than Cu-VRP, the optimal solution for Cu-VRPSD must be not better than the optimal solution for Cu-VRP. Hence, the lower bounds for Cu-VRP can still be used as lower bounds for Cu-VRPSD. Therefore, we compute the gaps using the previous lower bounds for both Algorithm 2 and Algorithm 2+, but compare the results with the previous results only for Algorithm 2+.

A summary of our results can be found in Table 2. The column “Set” lists the four previously described sets of instances. The column “Avg. Improvement”

for each  $X$ -set, with  $X \in \{A, B, P, E'\}$ , indicates the average improvement ratio of our results compared to the results in [16], where the improvement ratio for each instance in the  $X$ -set is calculated by  $\frac{\text{Previous Result} - \text{Our Result}}{\text{Previous Result}}$ . The column “Avg. Gap” is the average gap of our results compared to the lower bounds in [16], where the gap for each instance in the  $X$ -set is calculated by  $\frac{\text{Our Result} - \text{Lower Bound}}{\text{Lower Bound}}$ . Note that 7 out of 89 instances in [16] were not solved within 3 h, and thus do not have corresponding lower bounds or previous results. These instances are excluded from the calculation of average improvement ratios and gaps.

If we use the 1.5-approximation algorithm for metric TSP, we can see that: For Cu-VRPSD, the solution obtained by Algorithm 2 has an average gap of at most 80.08% for each  $X$ -set with  $X \in \{A, B, P, E'\}$ , implying that the approximation ratio of Algorithm 2 is at most 1.81 on average; For Cu-VRP, the solution obtained by Algorithm 2+ outperforms the previous algorithm in [16], and the average gap is at most 11.91% (resp., 22.00%) for the first three sets (resp., last set), achieving an average improvement ratio of at least 8.49%. If we use the LKH algorithm for metric TSP, the results can be further improved. Notably, the gap of Algorithm 2 reduces to at most 73.81% on average, and the improvement ratio of Algorithm 2+ achieves at least 11.32%. In conclusion, both Algorithm 2 and Algorithm 2+ generate solutions that are much closer to optimality than their provable approximation ratios suggest.

Our algorithms are implemented in C++ on a standard desktop computer with an AMD Ryzen 5 PRO 4650G with Radeon Graphics (3.70 GHz, 32.0 GB RAM). All 89 instances can be solved within 0.47 (resp., 13.82) seconds for Algorithm 2 and 0.48 (resp., 16.51) seconds for Algorithm 2+ if we use the 1.5-approximation (resp., LKH) algorithm for metric TSP. Note that the previous algorithm was run with a fixed time-out of 3 h per instance. The codes of our algorithms can be found in <http://github.com/JingyangZhao/Cu-VRP>.

**Acknowledgments.** The work is supported by the National Natural Science Foundation of China, under the grants 62372095, 62172077, and 62350710215.

## References

1. Altinkemer, K., Gavish, B.: Heuristics for unequal weight delivery problems with a fixed error guarantee. *Oper. Res. Lett.* **6**(4), 149–158 (1987)
2. Augerat, P., Naddef, D., Belenguer, J., Benavent, E., Corberan, A., Rinaldi, G.: Computational results with a branch and cut code for the capacitated vehicle routing problem (1995)
3. Bertsimas, D.J.: A vehicle routing problem with stochastic demand. *Oper. Res.* **40**(3), 574–585 (1992)
4. Bertsimas, D.J., Simchi-Levi, D.: A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Oper. Res.* **44**(2), 286–304 (1996)
5. Blauth, J., Traub, V., Vygen, J.: Improving the approximation ratio for capacitated vehicle routing. In: Mathematical Programming, pp. 1–47 (2022)

6. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report. Carnegie-Mellon University (1976)
7. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. *J. Oper. Res. Soc.* **20**, 309–318 (1969)
8. Corona-Gutiérrez, K., Nucamendi-Guillén, S., Lalla-Ruiz, E.: Vehicle routing with cumulative objectives: a state of the art and analysis. *Comput. Ind. Eng.* **169**, 108054 (2022)
9. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. *Manag. Sci.* **6**(1), 80–91 (1959)
10. Fernández Gil, A., Lalla-Ruiz, E., Gómez Sánchez, M., Castro, C.: The cumulative vehicle routing problem with time windows: models and algorithm. *Ann. Oper. Res.* 1–29 (2023)
11. Friggstad, Z., Mousavi, R., Rahgoshay, M., Salavatipour, M.R.: Improved approximations for capacitated vehicle routing with unsplittable client demands. In: IPCO 2022. LNCS, vol. 13265, pp. 251–261. Springer, Cham (2022)
12. Fukasawa, R., He, Q., Song, Y.: A branch-cut-and-price algorithm for the energy minimization vehicle routing problem. *Transp. Sci.* **50**(1), 23–34 (2016)
13. Gaur, D.R., Mudgal, A., Singh, R.R.: Routing vehicles to minimize fuel consumption. *Oper. Res. Lett.* **41**(6), 576–580 (2013)
14. Gaur, D.R., Mudgal, A., Singh, R.R.: Approximation algorithms for cumulative VRP with stochastic demands. In: Govindarajan, S., Maheshwari, A. (eds.) CAL-DAM 2016. LNCS, vol. 9602, pp. 176–189. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-29221-2\\_15](https://doi.org/10.1007/978-3-319-29221-2_15)
15. Gaur, D.R., Mudgal, A., Singh, R.R.: Improved approximation algorithms for cumulative VRP with stochastic demands. *Discret. Appl. Math.* **280**, 133–143 (2020)
16. Gaur, D.R., Singh, R.R.: A heuristic for cumulative vehicle routing using column generation. *Discret. Appl. Math.* **228**, 140–157 (2017)
17. Gendreau, M., Laporte, G., Séguin, R.: Stochastic vehicle routing. *Eur. J. Oper. Res.* **88**(1), 3–12 (1996)
18. Gupta, A., Nagarajan, V., Ravi, R.: Technical note - approximation algorithms for VRP with stochastic demands. *Oper. Res.* **60**(1), 123–127 (2012)
19. Haimovich, M., Kan, A.: Bounds and heuristics for capacitated routing problems. *Math. Oper. Res.* **10**(4), 527–542 (1985)
20. Helsgaun, K.: An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* **126**(1), 106–130 (2000)
21. Kara, İ., Kara, B.Y., Yetis, M.K.: Energy minimizing vehicle routing problem. In: Dress, A., Xu, Y., Zhu, B. (eds.) COCOA 2007. LNCS, vol. 4616, pp. 62–71. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-73556-4\\_9](https://doi.org/10.1007/978-3-540-73556-4_9)
22. Kara, İ., Kara, B.Y., Yetis, M.K.: Cumulative vehicle routing problems. IntechOpen Rijeka, HR (2008)
23. Karlin, A.R., Klein, N., Gharan, S.O.: A (slightly) improved approximation algorithm for metric TSP. In: STOC 2021, pp. 32–45. ACM (2021)
24. Mulati, M.H., Fukasawa, R., Miyazawa, F.K.: The arc-item-load and related formulations for the cumulative vehicle routing problem. *Discret. Optim.* **45**, 100710 (2022)
25. Newman, P., Alimoradian, B., Lyons, T.: Estimating fleet fuel consumption for vans and small trucks. *Transp. Sci.* **23**(1), 46–50 (1989)
26. Rinaldi, G., Yarrow, L.A.: Optimizing a 48-city traveling salesman problem: a case study in combinatorial problem solving. New York University, Graduate School of Business Administration (1985)

27. Sahin, B., Yilmaz, H., Ust, Y., Guneri, A.F., Gulsun, B.: An approach for analysing transportation costs and a case study. *Eur. J. Oper. Res.* **193**(1), 1–11 (2009)
28. Serdyukov, A.I.: Some extremal bypasses in graphs. *Upakovlyemye Sistemy* **17**, 76–79 (1978)
29. Wang, X., Choi, T.M., Liu, H., Yue, X.: A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios. *IEEE Trans. Syst. Man Cybern. Syst.* **48**(4), 545–556 (2016)
30. Xiao, Y., Zhao, Q., Kaku, I., Xu, Y.: Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Comput. Oper. Res.* **39**(7), 1419–1431 (2012)

# Author Index

## A

- Ali, U. A. Md. Ehsan 350  
Aurnob, Al Rafi 103

## B

- Bai, Yuze 119  
Bayram, Fatih 88  
Bei, Yijun 58  
Bharilya, Vibha 166

## C

- Chen, Fang 150  
Chen, Haopeng 330  
Chen, Jianhai 58  
Cheng, Keyang 73

## D

- Ding, Yu 150  
Doborjeh, Maryam 226  
Dwedar, Mohamed 88

## F

- Fan, Guoliang 366  
Feng, Zunlei 58  
Fu, Shaojing 1

## G

- Gao, Kewei 58  
Garnell, Saul 226

## H

- Hasegawa, Tatsuhito 301  
Hirose, Akira 286  
Horzyk, Adrian 381, 396  
Hou, Zeng-Guang 256  
Hu, Bin 256  
Huang, Chunyang 197

## I

- Islam, Md Rokon 103

## J

- Jesser, Alexander 88  
Jiang, Yuhan 241

## K

- Kameyama, Keisuke 350  
Konishi, Bungo 286  
Kumar, Neetesh 166

## L

- Li, Dapeng 366  
Li, Kangshun 241  
Li, Ping 42  
Li, Yizhou 119  
Li, Zhidong 150  
Liang, Bin 150  
Liang, Shuming 150  
Lin, Lin 366  
Lin, Sen 58  
Lin, Zihong 330  
Liu, De-Lin 256  
Liu, Erteng 58  
Liu, Lin 1  
Liu, Shi-Qi 256  
Lu, Xinyue 366  
Luo, Ji-Chang 256  
Luo, Yuchuan 1

## M

- Ma, Kehua 181  
Mahmud, Mufti 103  
Maniamfu, Pavodi 350  
Miah, Md Saef Ullah 103  
Moodley, Perusha 26

## N

- Natsuaki, Ryo 286

## P

- Paruchuri, Praveen 26

**Q**

Qiu, Zhenyu 1

**R**

Rahman, M. Mostafizur 103

Ratajczyk, Igor 396

**S**

Sakai, Ko 350

Sakai, Shunsuke 301

Shen, Junchen 42

Skrzyski, Jakub 381

Smith, Brian 226

Sun, Ming 211

Sun, Yingjie 119

Szyszka, Paul 226

**T**

Tanim, Sharia Arfin 103

Tao, Yucheng 330

Thamby, Dhillu 26

Tsuge, Shunsuke 301

Turkcan, Mehmet 226

**W**

Wan, Hao 73

Wang, Guiling 134

Wang, Jia-Xing 256

Wang, Kaixin 241

Wang, Nancy 134

Wang, Ruoyu 271

Wang, Tao 256

Wang, Yang 150

Wang, Ying 1

Wei, Aoao 181

Witbrock, Michael 16

Wu, Suping 181

**X**

Xiang, Zixue 211

Xiao, Mingyu 410

Xiao, Zhongwen 318

Xie, Xiao-Liang 256

**Y**

Yang, Lei 241

Yang, Shaohua 181

Yang, Yong 197

Yao, Jiawei 197

Yao, Junfeng 197

Yao, Lina 271

Yao, Wen 211

Young, Nathan 16

Yu, Jian 134

**Z**

Zhai, Zijie 42

Zhang, Ce 366

Zhang, Jie 42

Zhang, Kai 42

Zhang, Siqi 150

Zhang, Wenzhe 211

Zhang, Xitie 181

Zhang, Yuqi 134

Zhao, Jingyang 410

Zhao, Junfeng 181

Zhao, Pinjie 241

Zhao, Yanjie 318

Zheng, Chengping 119

Zheng, Xin 134

Zhou, Hao 73

Zhou, Lei 134

Zhou, Xiao-Hu 256

Zhou, Xing 58