

Hai Jin (Ed.)

Communications in Computer and Information Science

2452

Artificial Intelligence and Machine Learning

Second International Artificial Intelligence Conference, IAIC 2024
Jinyun, China, November 8–10, 2024
Revised Selected Papers, Part II

Part 2

 Springer



Communications in Computer and Information Science

2452

Series Editors

Gang Li , *School of Information Technology, Deakin University, Burwood, VIC, Australia*

Joaquim Filipe , *Polytechnic Institute of Setúbal, Setúbal, Portugal*

Zhiwei Xu, *Chinese Academy of Sciences, Beijing, China*

Rationale

The CCIS series is devoted to the publication of proceedings of computer science conferences. Its aim is to efficiently disseminate original research results in informatics in printed and electronic form. While the focus is on publication of peer-reviewed full papers presenting mature work, inclusion of reviewed short papers reporting on work in progress is welcome, too. Besides globally relevant meetings with internationally representative program committees guaranteeing a strict peer-reviewing and paper selection process, conferences run by societies or of high regional or national relevance are also considered for publication.

Topics

The topical scope of CCIS spans the entire spectrum of informatics ranging from foundational topics in the theory of computing to information and communications science and technology and a broad variety of interdisciplinary application fields.

Information for Volume Editors and Authors

Publication in CCIS is free of charge. No royalties are paid, however, we offer registered conference participants temporary free access to the online version of the conference proceedings on SpringerLink (<http://link.springer.com>) by means of an http referrer from the conference website and/or a number of complimentary printed copies, as specified in the official acceptance email of the event.

CCIS proceedings can be published in time for distribution at conferences or as post-proceedings, and delivered in the form of printed books and/or electronically as USBs and/or e-content licenses for accessing proceedings at SpringerLink. Furthermore, CCIS proceedings are included in the CCIS electronic book series hosted in the SpringerLink digital library at <http://link.springer.com/bookseries/7899>. Conferences publishing in CCIS are allowed to use Online Conference Service (OCS) for managing the whole proceedings lifecycle (from submission and reviewing to preparing for publication) free of charge.

Publication process

The language of publication is exclusively English. Authors publishing in CCIS have to sign the Springer CCIS copyright transfer form, however, they are free to use their material published in CCIS for substantially changed, more elaborate subsequent publications elsewhere. For the preparation of the camera-ready papers/files, authors have to strictly adhere to the Springer CCIS Authors' Instructions and are strongly encouraged to use the CCIS LaTeX style files or templates.

Abstracting/Indexing

CCIS is abstracted/indexed in DBLP, Google Scholar, EI-Compendex, Mathematical Reviews, SCImago, Scopus. CCIS volumes are also submitted for the inclusion in ISI Proceedings.

How to start

To start the evaluation of your proposal for inclusion in the CCIS series, please send an e-mail to ccis@springer.com.

Hai Jin
Editor

Artificial Intelligence and Machine Learning

Second International Artificial Intelligence Conference, IAIC 2024
Jinyun, China, November 8–10, 2024
Revised Selected Papers, Part II

Editor

Hai Jin

Huazhong University of Science and Technology
Wuhan, China

ISSN 1865-0929

ISSN 1865-0937 (electronic)

Communications in Computer and Information Science

ISBN 978-981-96-7190-8

ISBN 978-981-96-7191-5 (eBook)

<https://doi.org/10.1007/978-981-96-7191-5>

© 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

These conference proceedings are a collection of the papers accepted by IAIC 2024 — the 2024 International Artificial Intelligence Conference, held on November 8–10, 2024, in Jinyun, China.

The conference was organized by Zhejiang Sci-Tech University and Tech Science Press. IAIC 2024 provided a vibrant platform for the exchange of ideas and discussions on the latest advancements in artificial intelligence. The conference encompassed a wide range of topics, including machine learning, natural language processing, computer vision, robotics, AI ethics, and more.

The reviewing process for **IAIC 2024** was rigorous and meticulously organized. A total of **108 submissions** were received, and each manuscript underwent a **single-blind peer-review process**, with **two independent review reports** provided for every paper. Despite the high overall quality of the submissions, only **39 papers** were accepted for publication. This selective acceptance rate reflects the conference's commitment to academic excellence and the outstanding caliber of research showcased at IAIC 2024.

We extend our heartfelt gratitude to the authors for their excellent contributions, as well as to the reviewers for their dedication to ensuring a thorough evaluation process, which has been instrumental in maintaining the conference's high quality.

We would also like to express our appreciation to the organizers and sponsors, whose support and commitment made IAIC 2024 a success. Their dedication to advancing the field of artificial intelligence is truly commendable. The logos of our esteemed sponsors can be found on the following pages.

We hope this volume serves as a valuable resource for researchers, academics, and practitioners, fostering continued innovation and dialogue in the dynamic field of artificial intelligence.

IAIC 2024 Organizing Committee

December 2024

Organization

General Chair

Hai Jin	Huazhong University of Science and Technology, China
---------	---

Technical Program Committee Chair

Zhihui Zhan	Nankai University, China
-------------	--------------------------

Technical Program Committee Members

Sohaib Latif	University of Chenab, Pakistan
Wei Liang	Hunan University of Science and Technology, China
Muhammad Aamir	Huanggang Normal University, China
Ziaur Rahman	Huanggang Normal University, China
Zhonglin He	Huanggang Normal University, China
Jameel Ahmed Bhutto	Huanggang Normal University, China
Tariq Hussain	Zhejiang Gongshang University, China
Haseeb Ur Rahaman	University of Malakand, Pakistan
Muhammad Shahid Anwar	Gachon University, South Korea
Khurshed Aurangzeb	King Saud University, Saudi Arabia
Javed Ali Khan	University of Hertfordshire, UK
Hameedur Rahman	Air University, Pakistan
Yutao Tang	Guilin University of Electronic Technology, China
Qiong Wu	Jiangnan University, China
Zhiguo Qu	Nanjing University of Information Science and Technology, China
Le Sun	Nanjing University of Information Science and Technology, China
Chenxi Huang	Xiamen University, China
Yudong Zhang	University of Leicester, UK
Nitin Goyal	Central University of Haryana, India
Nonita Sharma	Indira Gandhi Delhi Technical University for Women, India

Xingsi Xue	Fujian University of Technology, China
Haoxi Zhang	Chengdu University of Information Technology, China
Xia Du	Xiamen University of Technology, China
Qi Zhang	City University of Macau, China
Jizhe Zhou	Sichuan University, China
Xin Liu	Lanzhou University, China
Saadaldeen Rashid Ahmed	Bayan University, Iraq
Gang Xu	North China University of Technology, China
Mohammed Amin Almaiah	University of Jordan, Jordan
Said A. Salloum	University of Salford, UK
Fang Liu	Hunan University, China
Zhiping Cai	National University of Defense Technology, China
Yizhang Wang	Yangzhou University, China
Le Zou	Hefei University, China
Weiwei Jiang	Beijing University of Posts and Telecommunications, China
Weixi Gu	China Academy of Industrial Internet, China
Miao He	Beijing Institute of Mathematical Sciences and Applications, China
Jian Su	Nanjing University of Information Science and Technology, China
Zhengguo Sheng	University of Sussex, UK
Siguang Chen	Nanjing University of Posts and Telecommunications, China
Wei Fang	Nanjing University of Information Science and Technology, China
Victor S. Sheng	Texas Tech University, USA

Organization Committee Members

Heming Jia	Sanming University, China
Yi Cai	South China University of Technology, China
Huiling Chen	Wenzhou University, China
Wei Fang	Jiangnan University, China
Jingrui Zhang	Xiamen University, China
Wentao Li	Southwest University, China
Deming Lei	Wuhan University of Technology, China
Keping Yu	Hosei University, Japan
Hongfeng Wang	Northeastern University, China

Shijun Liu	Shandong University, China
Xingsi Xue	Fujian University of Technology, China
Taiyong Li	Southwestern University of Finance and Economics, China
Zhi-Hui Zhan	Nankai University, China
Bin Ma	Shandong University of Technology, China
Rencan Nie	Yunnan University, China
Shiwei Cheng	Shanghai Jiao Tong University, China
Haidong Shao	Hunan University, China
Yin Zhang	University of Electronic Science and Technology of China, China
Yan Pei	University of Aizu, Japan
Changyan Yi	Nanjing University of Aeronautics and Astronautics, China
Chunwei Tian	Northwestern Polytechnical University, China
Zhiping Cai	National University of Defense Technology, China
Pei-Wei Tsai	Swinburne University of Technology, Australia
Farman Ali	Sungkyunkwan University, South Korea
Adam Slowik	Koszalin University of Technology, Poland
Nitin Goyal	Central University of Haryana, India
Cesar Maldonado Sanin	University of Newcastle, Australia
Zhenhua Yu	Xi'an University of Science and Technology, China
Xiaozhi Gao	University of Eastern Finland, Finland
Yue Zhang	Drexel University, USA
Nadeem Sarwar	Bahria University Lahore Campus, Pakistan
Saeed Hamood Alsamhi	University of Galway, Ireland
Alexander Mikroyannidis	Open University, UK
Ashutosh Kumar Dubey	Chitkara University, India
Eugenio Vocaturo	University of Calabria, Italy
Fabrizio Marozzo	University of Calabria, Italy
Jawad Ahmad	Edinburgh Napier University, UK
Marios Avgeris	Carleton University, Canada
Noor Zaman Jhanjhi	Taylor's University, Malaysia
Muhammad Tariq Sadiq	University of Essex, UK
Batyrkhan Omarov	Al-Farabi Kazakh National University, Kazakhstan
Anand Nayyar	Duy Tan University, Vietnam
Seifedine Kadry	Noroff University College, Norway
Essam H. Houssein	Minia University, Egypt
Ali Safaa Sadiq	Nottingham Trent University, UK

Sultan Daud Khan	National University of Technology, Pakistan
Jeng-Shyang Pan	Shandong University of Science and Technology, China
Roseline Oluwaseun Ogundokun	Landmark University, Nigeria
Mukesh Prasad	University of Technology Sydney, Australia
Qiong Wu	Jiangnan University, China
Xin Tan	East China Normal University, China
Songjie Wei	Nanjing University of Science and Technology, China
Gokhan Altan	Iskenderun Technical University, Turkey
Yu-Cheng Hu	Tunghai University, Taiwan
Zhi-Hui Zhan	Nankai University, China
Shahid Anwar	Gachon University, South Korea
Tianyuan Liu	Donghua University, China

Organizers

Co-organized by



Zhejiang Sci-tech University



Tech Science Press

Supporting Institutes



Contents – Part II

Artificial Intelligence in Network and Security Systems

Noise-Resilient Compressive Sensing via Orthogonality Constraints	3
<i>Yang Zheng, Zhen Zhang, Jie Yuan, Shuyi Feng, and Chenxin Lu</i>	
Joint Sensing of Computing Power and Network for Smart Grid: A Deep Matrix Completion Approach	18
<i>Yuxiang Qiu, Lei Zhang, Yuan Zhu, and Lan Gan</i>	
Rethinking RSSI-Based Key Extraction for UAVs and Ground Stations	31
<i>Yanbo Li and Jain Huang</i>	
Campus Card Security Depository System Based on State Secret Algorithm ...	43
<i>Yingzi Huo, Kefei Li, Meiguo Ke, Yanbing Wu, Yuxin Liu, and Ronglin Zhang</i>	
Network Traffic Classification via Deep Hashing Network	54
<i>Kunxiang Niu and Wen Luo</i>	
A Trust Management Integrated Bayesian Network Prediction in Vehicular Networks	66
<i>Hongbin Zhang, Zi Li, Dongmei Zhao, and Yuelin Liu</i>	
Research and Development of Distribution Line Defects and Violations Detection Method and Edge Computing System Based on Improved PPYOLOe	77
<i>Zaixin Chen, Peike Du, Tihua Yan, and Kaijun Wu</i>	
A Joint Optimization Algorithm for Computing Resource Allocation, UAV Trajectory, and Task Offloading in Remote Regional VANETs	93
<i>Geng Chen, Yiming Zhang, and Qingtian Zeng</i>	
LSTMR: Spatio-Temporal 3D Multiscale ResNet Model for Cellular Network Traffic Prediction	106
<i>Geng Chen, Weiming Wang, and Qingtian Zeng</i>	
An Efficient Name Lookup Approach Based on Character Indexing	121
<i>Dacheng He, Man Jiang, and Ke Xu</i>	

A Multi-channel Target Direction Speech Separation Model Based
on Efficient Convolutional Networks 132
Zaixin Chen, Yang Mao, Tihua Yan, and Biqing Li

Secure Cloud-Edge Collaborative Task Scheduling Framework Across
Data Centers 147
Ming Wan, Yuxiang Qiu, Yuan Zhu, and Lan Gan

Research on Security Sharing Mechanism of Cyber Threat Intelligence
Based on Consortium Blockchain 161
Fei Wang

Towards Automated Decoding of Vehicle CAN Data Using Deep Learning 181
Ankang Jiao, Wendi Li, Honglong Zhu, and Hao Han

Transfer Learning-Driven Surrogate-Assisted Differential Evolution
Algorithm with User Generated Content 201
Yumeng Peng, Lin Bao, Xiaoyan Sun, Jun Hao, Yu Wang, and Hang Yang

Author Index 217

Contents – Part I

Artificial Intelligence in Real-World Applications

An Adaptive Power Control Method for Power Communication Networks Based on Quantum Security	3
<i>Xingnan Li, Donghai Huang, Jiajia Fu, Zhongmiao Kang, and Jiangang Lu</i>	
From Machine Intelligence to Machine Ethics: An Analysis of the Practical Path of Artificial Morality	12
<i>Jingjing Zhao and Xingtong Liu</i>	
The Frequency Bounds for Travelling Salesman Problem Based on Frequency Quadrilaterals	23
<i>Yong Wang and WenFa Zheng</i>	
Speech Recognition Method Based on Improved Transformer	39
<i>Liqiang Wang, Dong Lv, Wenchen He, Haiyang Wang, Yifan Dong, and Mingshuang Sui</i>	
A Two-Stage Approach to Multi-Feature Acoustic Scene Classification	49
<i>Liqiang Wang, Yitao Li, Shan Ning, Haiyang Wang, Jie Chu, and Qirong Zhou</i>	
Star Extraction Based on Multi-scale Grey Morphology in Stray Light Backgrounds	61
<i>Shuhui Zhou, Yang Zheng, Chenxin Lu, Jinwei Cheng, and Wanyu Liu</i>	
Lightweight One-Stage Ship Detection Based on SAR Images for Embedded Platform	76
<i>Chenxin Lu, Ji Shen, Shuhui Zhou, Yang Zheng, and Jie Yuan</i>	
Multi-Modal False Information Identification Using BERT and ResNet	89
<i>Dong Lv, Yitao Li, Wenchen He, Zihan Wang, Jiawei Qin, and Liqiang Wang</i>	
Arithmetic Awareness Based Distributed Routing Mechanism for Power Dispatch Networks	99
<i>Zhifei Hao, Xiao Feng, Hang Zhang, Hanming Yan, and Guangwei Liu</i>	

A Dual-Mode Integrated Communication Method for Power Line Carrier and Wireless in Distribution Systems 109
Xiao Feng, Peiyao Zhang, Mingtao Cui, Yuhao Chang, Yong Sun, and Wenbin Liu

High-Performance Database Storage Technology for Power Smart Terminals 119
Lycho Huang, Xiao Feng, Jinguo Fang, Yuhao Chang, and Guangwei Liu

Quantum Key Distribution Strategy for Power Quantum Communication Networks Based on Trusted Relay Nodes 132
Donghai Huang, Xingnan Li, Jiajia Fu, Wenjuan Liang, and Zhongmiao Kang

Research on Network Traffic Anomaly Detection Method Integrating Graph Topology Optimization and Random Restart Walk Algorithm 142
Wen Luo, Haotian Li, Kunxiang Niu, and Longfei Yao

An Evaluation Method for Cutted Generated Video Based on Eigenvectors 158
Ji Yu, Haotian Gao, and Jundi Ding

Joint Optimization of Task Scheduling and Resource Allocation for Hierarchical Federated Learning 169
Nan Ding, Tianshu Huang, and Liutao Zhao

A Smart Grid Planning Data Decision Scoring Model Based on Random Forest 179
Lijun Xiao, Xiangwei Meng, Dezhi Han, and Dacheng He

Feature Importance on Axial Compressive Strength Prediction of Concrete-Filled Steel Tubular Columns Using Random Forest Algorithm 189
Bo Su, Zhenhong Wu, Dongqi Jiang, Gang Bi, Jiaxin Fan, and Kai Yan

AEDs Treatment and Cognition Analysis of BECTS Through EEG Features ... 201
Zhezha Zhao, Jiagui Chen, Dinghan Hu, Jiajia Fang, and Jiuwen Cao

Light Source Optimization Design of Intelligent Grading Equipment for Flue-Cured Tobacco 212
Chen Pengbiao, Hou Kaihu, Gong Peiyao, and Bai Fan

Multidimensional Attention Convolution Plugin for Ship Radiated Noise Classification 228
Zhongxiang Zheng and Peng Liu

Underwater Acoustic Target Detection Network Based on Comprehensive
Feature Extraction 239
 Denghui Sun, Kaiqiao Wang, Peng Liu, and Chun Zhang

Research on a Temperature Prediction Model for Meteorological Data
Integrating XGBoost and MLP 252
 Wenhua Xue, Yuxing Gao, Long Chen, and Enqi Lv

A Review of Research Advances in Image Segmentation of Skin Lesions 265
 Mengmeng He and Xiaoyan Zhang

Author Index 281

Artificial Intelligence in Network and Security Systems



Noise-Resilient Compressive Sensing via Orthogonality Constraints

Yang Zheng, Zhen Zhang, Jie Yuan, Shuyi Feng, and Chenxin Lu^(✉)

Shanghai Aerospace Electronic Technology Institute, Shanghai 201109, China
zhengyang_058@aliyun.com, lcxzzz3@163.com

Abstract. Deep neural networks have substantially improved reconstruction performance for compressive sensing (CS) in comparison to traditional optimization algorithms. However, their vulnerability to noise remains a critical challenge, often leading to significant degradation in performance. To overcome this challenge, we propose a universal algorithm that introduces orthogonality constraints on both the learned measurement matrix and the reconstruction neural network model in CS, with the objective of enhancing the resilience to noise for existing learning-based CS methods while preserving high reconstruction performance. To be concrete, we first develop an approximate restricted isometry property (RIP) to optimize the learned measurement matrix, thereby ensuring stable sampling in accordance with CS theory. We demonstrate that the approximate RIP is equivalent to imposing orthogonality constraints on the measurement matrix. Subsequently, we extend these orthogonality constraints to the reconstruction network to guarantee robust recovery with theoretical guarantees. Comprehensive experimental evaluations demonstrate that our approach significantly enhances robustness of learning-based CS methods, achieving competitive reconstruction performance when compared to current CS approaches.

Keywords: Compressed sensing · noise resilience · deep neural networks · restricted isometry property

1 Introduction

Compressive sensing (CS) theory ensures that a signal can be reliably reconstructed from far fewer measurements than those dictated by Nyquist sampling theory [1]. This approach is particularly efficient for signal sampling, as signals are inherently compressed during the sampling process, removing the necessity for high-speed processing or transmitting large amounts of data. Mathematically, an n -dimensional signal $x \in \mathbb{R}^n$ can be almost exactly recovered using $m \ll n$ randomized measurements $y = \Phi x = \Phi \Psi \alpha \in \mathbb{R}^m$ obtained by the linear projection $\Phi \in \mathbb{R}^{m \times n}$. Here, $\Psi \in \mathbb{R}^{n \times s}$ represents the transformation that induces a K -sparse representation vector $\alpha \in \mathbb{R}^s$ of x , and consequently, stable recovery of x from y is ensured when $\Phi \Psi$ satisfies restricted isometry property (RIP) [2].

The optimization problem for signal reconstruction in CS is usually formulated with a ℓ_1 -norm constraint on the sparse representation vector as follows:

$$\min_{\alpha \in \mathbb{R}^s} \|\alpha\|_1, \text{ s.t. } \|\Phi\Psi\alpha - y\|_2 \leq \epsilon, \quad (1)$$

where a small deviation ϵ represents the presence of noise in real-world observations y . The relaxation $\|\Phi\Psi\alpha - y\|_2 \leq \epsilon$ ensures that the reconstruction remains robust to noise, which is inevitable in practical applications. The majority of the noise can be mitigated by enforcing a strict sparsity constraint on the sparse representation vector such as $\|\alpha\|_1$, as the noise typically does not exhibit sparsity in the same domain.

The primary challenges in CS involve designing effective sampling matrices and developing high-accuracy reconstruction algorithms from linear measurements. For the first challenge, commonly used measurement matrices include the random Gaussian matrix [3], binary matrices with a 0/1 Bernoulli distribution [4, 5], and structured matrices [6, 7]. In terms of high-fidelity reconstruction, state-of-the-art algorithms such as convex optimization methods [8, 9], greedy algorithms [10], and iterative thresholding algorithms [11] have been developed. However, despite their theoretical guarantees of stable sampling and accurate reconstruction with overwhelming probability, traditional CS methods face limitations in real-world applications due to their high computational complexity and mediocre recovery quality [12].

Recently, deep neural networks (DNNs) have been integrated into CS to enhance sampling and reconstruction processes, leveraging their strong learning capabilities [12–25]. These DNN-based CS methods can either adaptively learn a projection from measurements to original images by minimizing the distortion between the initial scene and the target output [12–15, 22], or they can enable end-to-end training that jointly optimizes both sampling and reconstruction [16–21, 24, 25].

However, despite their impressive performance and accelerated reconstruction, DNN-based CS approaches are vulnerable to noise when models are trained on clean images. This vulnerability stems from two main factors: (1) The restricted isometry property (RIP), which ensures stable sampling in CS theory, is not guaranteed when using learnable measurement matrices in existing DNN-based CS methods [16–21, 24, 25]. (2) These CS reconstruction networks typically minimize the mean square error (MSE) between the model output and the ground truth to achieve high-quality recovery, rather than directly solving the reconstruction optimization problem as in Eq. (1), which limits their ability to inherit the robustness of traditional reconstruction algorithms.

To tackle this problem, we develop a universal algorithm to improve the robustness of existing DNN-based CS methods during the training phase. Specifically, by enforcing orthogonality in each layer of the DNN-based CS models, the models achieve greater noise resilience and improved reconstruction performance. We also provide a theoretical analysis linking orthogonality constraints to enhanced noise-resilient properties. Our main contributions are summarized as follows:

1. Inspired by the RIP that ensures stable sampling for CS, we derive a concise and effective optimization strategy to enhance sampling stability by constraining the orthogonality of the learnable measurement matrix.
2. We further apply orthogonality constraints to the reconstruction network and theoretically demonstrate its resilience to noise when each layer of the network is normalized to an orthonormal or semi-orthogonal matrix.
3. Extensive experiments verify that our algorithm enhances the robustness of DNN-based CS methods while maintaining competitive recovery quality compared to current techniques.

2 Related Work

2.1 Conventional CS Algorithms

Conventional CS algorithms typically employ ℓ_1 -norm constrained optimization to reconstruct x from measurements y , under the assumption that x is sparse:

$$\min_x \frac{1}{2} \|\Phi x - y\|_2^2 + \lambda \|\Psi x\|_1. \quad (2)$$

In this formulation, Ψ represents the sparsifying basis (or transformation matrix) that induces sparsity in x , and λ is the regularization parameter. The basis Ψ can be derived from analytical linear transforms, such as the discrete cosine transform (DCT) or wavelet transforms [26], or can be learned by incorporating prior knowledge, such as structural patterns [27] or union of subspaces [28].

Various approaches have been developed to solve the constrained optimization problem in Eq. (2). For instance, Zhang *et al.* [29] utilized collaborative sparsity priors to exploit the non-local self-similarity inherent in natural images. Metzler *et al.* [30] integrated the approximate message passing (AMP) framework with a well-established BM3D denoiser [31] for CS recovery. Additionally, Ma *et al.* [32] reformulated the AMP algorithm as a forward-backward splitting algorithm to reconstruct signals in compressive video sampling. Azghani *et al.* [33] proposed a multihypothesis reconstruction strategy utilizing the alternating direction method of multipliers (ADMM) [34] for compressive video sampling. Furthermore, Dinh and Jeon [35] developed an iterative weighted recovery method for block-based CS, which focuses on low-frequency signal components and constrains the solution space using simplified Landweber iterations [36].

The conventional CS algorithms achieve robust reconstruction from noisy measurements by fulfilling two fundamental requirements: (i) the use of hand-crafted measurement matrices, denoted as Φ , which are designed to satisfy the RIP of matrix A ; and (ii) the recovery of a sparse solution to Eq. (2) using traditional reconstruction algorithms that effectively attenuate noise. Despite their robustness, these methods are often limited by substantial computational complexity and mediocre reconstruction quality, typically requiring numerous iterations to solve Eq. (2) or its variants [12]. In this paper, we explore deep learning-based approaches to compressed sensing, with the objective of enhancing reconstruction efficiency and performance while ensuring effective noise suppression.

2.2 Deep Learning Based CS Methods

Recently, DNNs have been increasingly integrated into CS frameworks to enhance reconstruction efficiency and performance [12–25]. Mousavi *et al.* [13] designed a stacked denoising autoencoder that, in an unsupervised way, learns structured representations from training data for reconstruction purposes. Kulkarni *et al.* [12] and Mousavi *et al.* [18] designed novel DNN models for CS reconstruction, which learns a non-linear projection from measurements to original image blocks, subsequently stitching these blocks together to form the original image. Yao *et al.* [14] further improved reconstruction quality by leveraging residual learning techniques. Zhang *et al.* [15] presented a learning-based CS framework derived from the Iterative Shrinkage-Thresholding Algorithm (ISTA), successfully translating optimization methods into a DNN. Shi *et al.* [19, 21] pioneered the exploration of CNN-based CS models and later proposed a scalable CS model capable of reconstructing signals at various sampling ratios using a single model [20]. More recently, Shen *et al.* introduced transformers to facilitate CS reconstruction [24], and Zhang *et al.* proposed an adaptive multi-scale network for image CS [25].

Despite the promising performance of DNN-based CS methods, they often fall short of meeting two essential robustness criteria: (i) ensuring that the measurement matrix satisfies the RIP, and (ii) producing a sparse solution that is resilient to noise. While some approaches [12–15, 22] employ a random Gaussian matrix for sampling to fulfill criterion (i), the learned projection from measurements to the original image frequently contradicts criterion (ii). Conversely, end-to-end CS methods that utilize a learned measurement matrix [16–21, 24, 25] have demonstrated exceptional reconstruction performance. However, as depicted in Fig. 1, these methods exhibit instability in the presence of noise.

To overcome these challenges, this paper introduces a universal optimization algorithm for DNN-based CS methods, designed to ensure stable reconstruction. The proposed algorithm integrates the superior reconstruction performance of end-to-end CS approaches with the robustness of traditional CS techniques. In Sect. 3, we present a theoretical analysis of robustness in both sampling and reconstruction, while Sect. 4 provides empirical evidence of the algorithm’s universality and effectiveness across various scenarios.

3 Robust Compressive Sensing via Orthogonality Constraints

In this section, we propose an algorithm that optimize both sampling and reconstruction to enhance the robustness of DNN-based CS methods through orthogonality constraints. We first derive an optimization strategy based on RIP to enhance sampling stability by constraining the orthogonality of the learnable measurement matrix. We further apply orthogonality constraints to the reconstruction network and theoretically prove its resilience to noise in Theorem 1.

3.1 Stable Sampling via Orthogonality Constraints on Measurement Matrix

According to the definitions of restricted isometry constant and property [2], the restricted isometry constant (RIC) of a matrix $A \in \mathbb{R}^{m \times s}$ is the smallest $\delta_K \in (0, 1)$ satisfying that

$$(1 - \delta_K) \|\alpha\|_2^2 \leq \|A\alpha\|_2^2 \leq (1 + \delta_K) \|\alpha\|_2^2, \quad (3)$$

for all K -sparse vectors $\alpha \in \mathbb{R}^s$. When the matrix A satisfies the K -RIP with the RIC δ_K , the accurate signal reconstruction based on the noisy measurements can be guaranteed by finding the exact solution to the optimization problem (1). It is worth mentioning that RIP not only ensures reconstruction quality but also serves as a measure of noise suppression capability, indicating that the suppression ability is negatively correlated with δ_K . Furthermore, it is shown in [37] that given arbitrary Gaussian measurement matrix Φ and a transformation matrix Ψ , suppose that

$$C \cdot \mu^2(\Phi, \Psi) \cdot K \cdot \log n \leq m, \quad (4)$$

then A satisfies RIP with overwhelming probability, where C is some positive constant, $K = \|\alpha\|_0$ denote the sparsity, n is the dimension of original signal x , and $\mu(\Phi, \Psi)$ is the mutual coherence of Φ and Ψ defined by

$$\mu(\Phi, \Psi) = \sup\{|\langle \Phi_i, \Psi_j \rangle|\}. \quad (5)$$

Note that $\mu(\Phi, \Psi)$ is usually close to 0 in CS [37].

Subsequently, we demonstrate that RIP is hardly guaranteed in DNN based CS methods. It is worth emphasizing that DNN-based CS methods do not sufficiently exploit the sparsity of the original signal x via the transformation matrix Φ . Those end-to-end networks [16–20] learned the measurement matrix Φ instead of $A = \Phi\Psi$, which can be viewed as $A = \Phi$ and $\Psi = I$ with the identity matrix I . In this case, K corresponds to $\|x\|_0$ rather than $\|\alpha\|_0$, resulting in a larger value since there is no transformation matrix to induce sparsity in x . Since the weight distribution of DNNs is generally observed to follow a Gaussian distribution [38], the learned measurement matrix Φ is expected to comply with this pattern. Hence, we have $\mu(\Phi, \Psi) = \mu(\Phi, I) = |\phi|$, where ϕ is the element of Φ that satisfies Eq. (5) and $|\phi|$ follows the folded normal distribution, implying that $\mu(\Phi, I)$ is close to 0 with a high probability. According to Eq. (4), when comparing conventional CS methods with DNN-based CS methods, the values of $\mu(\Phi, \Psi)$ are both close to 0, and the constant C as well as the dimensions m and n are the same. However, K in DNN-based CS methods is significantly larger than in conventional CS methods, which implies that the matrix A is less likely to satisfy the RIP in DNN-based CS compared to conventional CS.

Therefore, we propose optimizing Φ based on Eq. (3) to ensure the RIP for DNN-based CS methods. As $A = \Phi$ and $\alpha = x$ in DNN-based CS, Φ can be optimized as:

$$\delta_K \triangleq \arg \min_{\Phi} \frac{|\|\Phi x\|_2^2 - \|x\|_2^2|}{\|x\|_2^2}, \quad (6)$$

Eq. (6) can be simplified by

$$\frac{|\|\Phi x\|_2^2 - \|x\|_2^2|}{\|x\|_2^2} = \frac{|\langle (\Phi^* \Phi - I)x, x \rangle|}{\|x\|_2^2} \leq \delta_K. \quad (7)$$

According to Cauchy-Schwarz inequality ($|\langle x, y \rangle| \leq \|x\|_2 \|y\|_2$) and sub-multiplicativity of p -norm ($\|MN\|_p \leq \|M\|_p \|N\|_p$), we have

$$|\langle (\Phi^* \Phi - I)x, x \rangle| \leq \|\Phi^* \Phi - I\|_2 \|x\|_2^2. \quad (8)$$

Considering Eq. (7) and (8), we define $\tilde{\delta}_K$ as follows:

$$\tilde{\delta}_K \triangleq \arg \min_{\Phi} \|\Phi^* \Phi - I\|_2. \quad (9)$$

Here, $\delta_K \leq \tilde{\delta}_K$, which implies that δ_K would satisfy Eq. (3), when $\tilde{\delta}_K$ does. In the network training phase, the measurement matrix Φ is iteratively updated by solving Eq. (9) with gradient descent steps. The iterative update step is defined in Eq. (10):

$$\Phi \leftarrow \Phi - 4\beta(\Phi^* \Phi - I)\Phi \quad (10)$$

where β is learning rate and $4(\Phi^* \Phi - I)\Phi$ is the gradient of $\|\Phi^* \Phi - I\|_2$. Compared with Eq. (6), we eliminate the terms related to the original image x to simplify the calculation and renew Φ by itself. We further evaluate the effectiveness of approximate RIP in Sect. 4.

3.2 Robust Reconstruction Based on DNNs with Orthogonality Constraints

Reconstruction neural networks are vulnerable to noisy CS measurements, as it is trained to minimize MSE rather than find the sparse solution to Eq. (1). To address this problem, we present Theorem 1 to guarantee the robustness of proposed DNN-based optimization via enforcing the orthogonality. Supposing that the network utilizes ReLU function and is constructed without bias terms, and the measurements are corrupted by Gaussian noise, we draw the conclusion that the network can suppress noise when its weights satisfy Theorem 1.

Theorem 1. Denote $\delta_1, \dots, \delta_{n_0}$ independent normal random variables with zero mean and variance σ^2 . Given arbitrary matrices $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$ for $k \in [1, m]$. If noise $\vec{\delta} = [\delta_1, \dots, \delta_{n_0}]^T$, we define the function $f_k(\vec{\delta}) = \text{ReLU}(W_k \vec{\delta})$ and $f_k \circ f_{k-1}(\vec{x}) = \text{ReLU}[W_k \cdot \text{ReLU}(W_{k-1} \vec{x})]$, where ReLU is the commonly used activation function in DNNs satisfying $\text{ReLU}(x) = \max(0, x)$. When W_k can be normalized to orthonormal or semi-orthogonal matrices, the noise is limited based on the concatenation function $f_m \circ f_{m-1} \cdots f_1$:

$$\mathbb{E}[\|f_m \circ f_{m-1} \cdots f_1(\vec{\delta})\|_2^2] \leq \mathbb{E}[\|\vec{\delta}\|_2^2], \quad (11)$$

which means that a deep neural network with its expression function $f_m \circ f_{m-1} \cdots f_1$ is noise-resilient. $\square$

Proof. Suppose $\mathcal{G}_m(\vec{x}) = f_m \circ f_{m-1} \cdots f_1(\vec{x})$. When the neural network converges to a stable state, that is $\mathbb{E}[\|\mathcal{G}_m(\vec{y}) - \Phi^+ \vec{y}\|_2^2] \leq \epsilon_1$ for a noiseless recovery, where $\vec{y}$ represents measurements and Φ^+ denotes the inverse transformation of sampling projection, thus $\Phi^+ \vec{y}$ is the original signal. For a noisy recovery, the ideal condition is that $\mathbb{E}[\|\mathcal{G}_m(\vec{y} + \vec{\delta}) - \Phi^+ \vec{y}\|_2^2] \leq \epsilon_1 + \mathbb{E}[\|\vec{\delta}\|_2^2]$, which means the noise $\vec{\delta}$ has been limited in network.

Note that $f_k(\vec{a} + \vec{b}) \leq f_k(\vec{a}) + f_k(\vec{b})$, as $\text{ReLU}(\vec{a} + \vec{b}) \leq \text{ReLU}(\vec{a}) + \text{ReLU}(\vec{b})$. Since $\mathbb{E}[\|\mathcal{G}_m(\vec{y} + \vec{\delta}) - \Phi^+ \vec{y}\|_2^2] \leq \mathbb{E}[\|\mathcal{G}_m(\vec{y}) - \Phi^+ \vec{y}\|_2^2] + \mathbb{E}[\|\mathcal{G}_m(\vec{\delta})\|_2^2] \leq \epsilon_1 + \mathbb{E}[\|\mathcal{G}_m(\vec{\delta})\|_2^2]$, we alternatively prove $\mathbb{E}[\|\mathcal{G}_m(\vec{\delta})\|_2^2] \leq \mathbb{E}[\|\vec{\delta}\|_2^2]$.

i) When $m = 1$, $\vec{Y} = W_1 \vec{\delta}$ and $\vec{Z} = \text{ReLU}(\vec{Y}) = \max(Y_i, 0) = [Z_1, \dots, Z_{n_1}]^T$ with $\vec{Z} = \mathcal{G}_1(\vec{\delta}) = f_1(\vec{\delta})$. Since $\vec{\delta} \sim \mathcal{N}(0, \Sigma_{\vec{\delta}})$ with $\Sigma_{\vec{\delta}} = \sigma^2 I$, we have $\vec{Y} \sim \mathcal{N}(0, \Sigma_{\vec{Y}})$ with $\Sigma_{\vec{Y}} = W_1 \Sigma_{\vec{\delta}} W_1^T$. The variance of the i -th row of $\vec{Y}$ is $\text{var}(Y_i) = (\eta_i \sigma)^2$ with the singular value η_i of W_1 for $i = 1, \dots, n_1$. Thus, the probability density function (pdf) $f_{Y_i}(\delta)$ of Y_i is $f_{Y_i}(\delta) = \exp(-\delta^2/2(\eta_i \sigma)^2)/\sqrt{2\pi}\eta_i \sigma$. Given $Z_i = \max(Y_i, 0)$, we can calculate its mean $\mathbb{E}(Z_i) = \eta_i \sigma / \sqrt{2\pi}$ and variance $\mathbb{E}(Z_i^2) = (\eta_i \sigma)^2/2$. Notice that

$$\mathbb{E}(\|\vec{Z}\|_2^2) = \mathbb{E}[\text{var}(\vec{Z})] = \mathbb{E}(\vec{Z}^T \Sigma_{\vec{Z}} \vec{Z}) = \mathbb{E}(\vec{Z}^T \Sigma_{\vec{Z}}^d \vec{Z} + \vec{Z}^T \Sigma_{\vec{Z}}^n \vec{Z}), \quad (12)$$

where $\Sigma_{\vec{Z}}^d$ is the diagonal of $\Sigma_{\vec{Z}}$ and $\Sigma_{\vec{Z}}^n$ is the off-diagonal part. We can obtain $\mathbb{E}(\vec{Z}^T \Sigma_{\vec{Z}}^d \vec{Z}) = \sum_{i=1}^{n_1} \mathbb{E}(Z_i^2) = \sum_{i=1}^{n_1} \frac{(\eta_i \sigma)^2}{2}$. For $\Sigma_{\vec{Z}}^n$, its (i, j) -th element is covariance $\text{cov}(Z_i, Z_j) = \mathbb{E}(Z_i Z_j) - \mathbb{E}(Z_i)\mathbb{E}(Z_j)$, $i \neq j$. To calculate $\mathbb{E}(Z_i Z_j)$, we first discuss the relation between $\vec{Y}$ and $\vec{Z}$. The joint pdf of Y_i and Y_j is

$$f_{Y_i, Y_j}(x, y) = \Gamma \cdot e^{-\frac{1}{2(1-\rho_{ij}^2)} \left(\frac{x^2}{(\eta_i \sigma)^2} - \frac{2\rho_{ij}xy}{\eta_i \eta_j \sigma^2} + \frac{y^2}{(\eta_j \sigma)^2} \right)}, \quad (13)$$

where $\Gamma = (2\pi\eta_i\eta_j\sigma^2\sqrt{1-\rho_{ij}^2})^{-1}$ and ρ_{ij} is the correlation coefficient between Y_i and Y_j . From Eq. (13), we have

$$\mathbb{E}(Z_i Z_j) = \Lambda \sigma^2 \cdot \int_0^\infty \int_0^\infty e^{-\frac{1}{2}(u^2 - 2\rho_{ij}uv + v^2)} uv du dv, \quad (14)$$

where $u = x(\eta_i \sigma \sqrt{1-\rho_{ij}^2})^{-1}$, $v = y(\eta_j \sigma \sqrt{1-\rho_{ij}^2})^{-1}$, $\Lambda = \frac{1}{2\pi}(\sqrt{1-\rho_{ij}^2})^3 \eta_i \eta_j \sigma^2$. Considering that $\mathbb{E}(Z_i Z_j) = M_{ij} \sigma^2$, we obtain for the (i, j) -th element of $\Sigma_{\vec{Z}}^n$

$$\text{cov}(Z_i, Z_j) = \mathbb{E}(Z_i Z_j) - \mathbb{E}(Z_i)\mathbb{E}(Z_j) = (M_{ij} - \frac{\eta_i \eta_j}{2\pi}) \sigma^2, \quad (15)$$

Since Eq. (15) has a complicated effect on Eq. (12), we suppose $\text{cov}(Z_i, Z_j) = 0$ by letting $M_{ij} = \frac{\eta_i \eta_j}{2\pi}$. Thus,

$$\left(\sqrt{1-\rho_{ij}^2} \right)^3 \int_0^\infty \int_0^\infty e^{-\frac{1}{2}(u^2 - 2\rho_{ij}uv + v^2)} uv du dv = 1, \quad (16)$$

Eq. (16) holds only when $\rho_{ij} = 0$ and $\text{cov}(Y_i, Y_j) = 0$. Since $\text{cov}(Y_i, Y_j) = \sigma^2 |\langle W_{1,i}, W_{1,j} \rangle| = 0, i \neq j$ for $\Sigma_{\vec{Y}} = W_1 \Sigma_{\vec{\delta}} W_1^T = \sigma^2 W_1 W_1^T$, we have for the i -th row $W_{1,i}$ of W_1 .

$$|\langle W_{1,i}, W_{1,j} \rangle| = 0, i \neq j, \quad (16)$$

Thus, we can rewrite $\mathbb{E}[\|\mathcal{G}_1(\vec{\delta})\|_2^2] \leq \mathbb{E}[\|\vec{\delta}\|_2^2]$ as

$$\mathbb{E}(\|\vec{Z}\|_2^2) = \mathbb{E}(\vec{Z}^T \Sigma_{\vec{Z}}^d \vec{Z}) = \sum_{i=1}^{n_1} \frac{(\eta_i \sigma)^2}{2} \leq \mathbb{E}(\|\vec{\delta}\|^2) = n_1 \sigma^2, \quad (17)$$

The above equation can be satisfied if $\eta_i \leq \sqrt{2}$. In conclusion, $|\langle W_{1,i}, W_{1,j} \rangle| = 0$ and $\eta_i \leq \sqrt{2}$ can ensure that the noise has been limited in network with $m = 1$ layer, which means that W_1 can be normalized as an orthogonal or semi-orthogonal matrix.

ii) When $m > 1$, we have for $\vec{Z} = \mathcal{G}_m(\vec{\delta})$,

$$\text{cov}(Z_i, Z_j) = (M_{ij} - \frac{\eta_i \eta_j}{2\pi})_m \cdots (M_{ij} - \frac{\eta_i \eta_j}{2\pi})_1 \sigma^2. \quad (18)$$

It means that each $W_k, k \in [1, m]$ needs to satisfy $|\langle W_{k,i}, W_{k,j} \rangle| = 0, i \neq j$ to force $\text{cov}(Z_i, Z_j) = 0$.

$$\mathbb{E}(\|\mathcal{G}_m(\vec{\delta})\|^2) = \mathbb{E}(\|\vec{Z}\|^2) = \sum_{i_m=1}^{n_m} \cdots \sum_{i_1=1}^{n_1} \frac{(\eta_{i_m} \cdots \eta_{i_1})^2}{2} \sigma^2 \leq \mathbb{E}(\|\vec{\delta}\|^2) = n_1 \sigma^2 \quad (18)$$

Eq. (18) holds when each W_k can be normalized to orthonormal or semi-orthogonal matrix. $\square$

Theorem 1 indicates that network weights can be updated following the iterative steps as Eq. (10), and we arrive at the same conclusion as Sect. 3.1 from different perspectives. Note that the convolution operation also conforms to Theorem 1, as it can be transformed into matrix multiplication, although this process can be time-consuming. To adapt the process for convolution layers, we generalize the orthogonality optimization with a direct modification following [39]. For a convolution kernel $C_k \in \mathbb{R}^{h_k \times w_k \times c_k \times o_k}$, where h_k, w_k, c_k , and o_k represent the height, width, input channel number, and output channel number respectively, we first reshape C_k into matrix $W_{C_k} \in \mathbb{R}^{h_k w_k c_k \times o_k}$ and then apply Eq. (10) to it. Although this optimization strategy does not strictly conform to the steps indicated in Theorem 1, it works empirically in practice. As each W_k approximates an orthonormal or semi-orthogonal matrix, its ability to suppress noise is enhanced, as shown in Theorem 1. The proposed algorithm for training the CS models is described in Algorithm 1. In Sect. 4, extensive experiments conducted on current CS methods with our algorithm provide further validation.

4 Experimental Results

We employ the proposed algorithm in six state-of-the-art network-based CS methods, which consist of recognized and cutting-edge methods, namely ReconNet [12], *Adp-Recon* [16], Dr2-Net [14], ISTA-Net⁺ [15], CSNet [21], MADUN [23], TransCS [24], and HGLISTA [22] for evaluation. ReconNet, Dr2-Net, ISTA-Net⁺, and HGLISTA sample images with a random Gaussian matrix, while *Adp-Recon*, the enhanced version of ReconNet, MADUN, TransCS, and CSNet utilize a learned measurement matrix. SCSNet [20], the enhanced version of CSNet, utilizes the same sampling architecture as CSNet and a parallel reconstruction network taking CSNet reconstruction as its branch. These branches are connected with shortcuts. For convenience, we conduct experiments on CSNet instead of SCSNet, as they show similar results.

To ensure a fair comparison, we utilize the same set of 91 images for training as in [12]. The experiments are carried out using TensorFlow on a workstation equipped with an Intel Xeon E5-2603 CPU and a GTX 1080Ti GPU. Consistent with standard practices in prior CS research, we generate 22,092 training pairs by randomly cropping image blocks (each 33×33 in size) from the 91 images. We use to implement all the methods for a range of measurement rates (MRs), i.e., 1%, 4%, 10%, 25%, and 50%, and train the network using Adam optimization with an initial learning rate of 0.0001 (2,000 epochs) and a batch size of 128. All the learning rates $\{\beta_k\}_{k=0}^m$ of the update in Eq. (10) are set to 0.0003, and we conduct the optimization every time 50 batches of images are fed, i.e., $V = 50$.

Algorithm 1. The proposed training algorithm with orthogonality constraints

Input: Optimization interval V , learning rates $\{\beta_k\}_{k=0}^m$, learned measurement matrix Φ , weights $\{W_k \vee C_k\}_{k=1}^m$ in reconstruction network model.

Output: Optimized CS model with orthogonality constraints.

```

1: for every  $V$  training batch do
2:   Optimize  $\Phi$  and  $\{W_k \vee C_k\}_{k=1}^m$  via backpropagation training
3:    $\Phi \leftarrow \Phi - 4\beta_0(\Phi^*\Phi - I)\Phi$ 
4:   for  $k = 1$  to  $m$  do
5:     if fully-connected layer  $W_k$  then
6:        $W_k \leftarrow W_k - 4\beta_k(W_k^*W_k - I)W_k$ 
7:     else if convolution layer  $C_k$  then
8:       Reshape  $C_k \in \mathbb{R}^{h_k \times w_k \times c_k \times o_k}$  into  $W_{C_k} \in \mathbb{R}^{h_k w_k c_k \times o_k}$ 
9:        $W_{C_k} \leftarrow W_{C_k} - 4\beta_k(W_{C_k}^*W_{C_k} - I)W_{C_k}$ 
10:      Reshape  $W_{C_k}$  back into  $C_k$ 
11:     end if
12:   end for
13: end for
14: Return  $\mathbf{x}_0$ 

```

Table 1. Average PSNR(dB)/SSIM for various methods on *Set 11* with MRs of 1%, 4%, 10%, 25%, and 50%. Std denotes the standard deviation of noise. The prefix ‘*Rob-*’ represents our algorithm. The best results is in bold.

Std	Methods	MR: 50%		MR: 25%		MR: 10%		MR: 4%		MR: 1%	
–	ReconNet [12]	25.15	0.7567	24.18	0.7109	22.21	0.6265	20.91	0.5767	18.07	0.4400
	<i>Adp</i> -Recon [16]	32.15	0.9255	29.48	0.8855	27.34	0.8284	24.14	0.7098	20.58	0.5122
	<i>Rob</i> -Recon	31.98	0.9231	30.01	0.8952	27.48	0.8342	23.97	0.6939	20.46	0.5129
	Dr2-Net [14]	29.82	0.8659	26.83	0.7872	23.39	0.6649	20.89	0.5518	17.84	0.4275
	<i>Rob</i> -Dr2	33.47	0.9368	29.56	0.8696	26.62	0.8002	23.84	0.6831	20.41	0.4970
	ISTA-Net ⁺ [15]	35.79	0.9541	30.76	0.8952	25.69	0.7738	21.56	0.6165	17.79	0.4456
	<i>Rob</i> -ISTA ⁺	36.30	0.9693	30.96	0.9192	26.24	0.8176	23.58	0.7005	20.02	0.5036
	CSNet [21]	36.95	0.9692	31.73	0.9214	27.44	0.8357	24.18	0.7123	20.58	0.5138
	<i>Rob</i> -CSNet	36.70	0.9684	31.73	0.9212	27.68	0.8401	24.33	0.7204	20.64	0.5216
	MADUN [23]	38.58	0.9738	32.88	0.9317	27.08	0.8269	21.33	0.6341	18.40	0.4586
	<i>Rob</i> -MADUN	38.66	0.9768	32.89	0.9323	27.21	0.8287	21.64	0.6472	19.44	0.4881
	TransCS [24]	38.74	0.9766	33.01	0.9399	28.48	0.8569	24.33	0.7241	20.77	0.5223
	<i>Rob</i> -TransCS	38.82	0.9778	33.19	0.9405	28.82	0.8687	24.64	0.7277	20.90	0.5287
	HGLISTA [22]	36.77	0.9628	31.48	0.9111	25.21	0.7676	21.92	0.6333	18.35	0.4557
	<i>Rob</i> -HGLISTA	40.23	0.9873	33.84	0.9725	27.87	0.8467	24.72	0.7603	20.76	0.5309
0.05	ReconNet [12]	24.82	0.7298	23.91	0.6959	22.00	0.6176	19.62	0.4879	17.40	0.3944
	<i>Adp</i> -Recon [16]	21.33	0.5452	22.57	0.5815	23.16	0.5805	22.67	0.6165	20.43	0.5037
	<i>Rob</i> -Recon	26.90	0.7631	27.39	0.7500	26.80	0.7938	23.84	0.6824	20.45	0.5123
	Dr2-Net [14]	23.05	0.5150	22.55	0.5071	21.61	0.5035	19.94	0.4607	17.30	0.3823
	<i>Rob</i> -Dr2	26.61	0.6588	26.28	0.6841	25.49	0.7053	23.46	0.6490	20.35	0.4925
	ISTA-Net ⁺ [15]	26.87	0.7187	28.61	0.8492	24.00	0.6951	20.80	0.5693	17.28	0.4127
	<i>Rob</i> -ISTA ⁺	33.16	0.9239	29.69	0.9077	26.17	0.8162	23.58	0.6999	20.01	0.5035
	CSNet [21]	15.59	0.2426	19.23	0.3729	22.82	0.5840	23.22	0.6491	20.50	0.5089
	<i>Rob</i> -CSNet	27.00	0.6810	28.11	0.7752	26.79	0.7914	24.18	0.7099	20.63	0.5208
	MADUN [23]	28.09	0.7931	24.37	0.7143	22.79	0.5299	20.33	0.4233	17.79	0.3989
	<i>Rob</i> -MADUN	32.03	0.9199	29.88	0.9153	27.12	0.8222	24.31	0.7021	19.34	0.4851
	TransCS [24]	28.55	0.8025	25.21	0.7223	22.93	0.5442	21.45	0.6690	19.44	0.4992
	<i>Rob</i> -TransCS	31.03	0.9168	29.79	0.9053	27.00	0.8187	24.02	0.7072	20.44	0.5151
	HGLISTA [22]	30.02	0.8723	28.97	0.8522	24.65	0.7115	21.33	0.5801	17.64	0.4166
	<i>Rob</i> -HGLISTA	35.01	0.9562	30.35	0.9167	26.81	0.8055	24.44	0.7137	20.34	0.5011
0.10	ReconNet [12]	23.83	0.6536	23.02	0.6401	21.15	0.5582	17.35	0.3368	16.36	0.3414
	<i>Adp</i> -Recon [16]	15.46	0.2302	16.52	0.3625	18.94	0.3849	19.96	0.4694	19.99	0.4793
	<i>Rob</i> -Recon	24.13	0.5987	23.69	0.5581	25.14	0.6897	23.54	0.6568	20.43	0.5101
	Dr2-Net [14]	17.92	0.3084	18.23	0.3199	18.42	0.3294	17.74	0.3212	16.36	0.3291
	<i>Rob</i> -Dr2	21.26	0.4337	22.44	0.4896	23.45	0.5745	22.68	0.5826	20.25	0.4853
	ISTA-Net ⁺ [15]	16.91	0.2627	21.90	0.5043	20.23	0.4040	18.88	0.4654	15.68	0.3186
	<i>Rob</i> -ISTA ⁺	30.91	0.9116	29.06	0.8988	26.12	0.8137	23.52	0.6982	19.97	0.5011
	CSNet [21]	9.899	0.1049	13.33	0.1852	18.04	0.3565	21.37	0.5328	20.33	0.4990
	<i>Rob</i> -CSNet	20.74	0.4182	23.58	0.5581	24.81	0.6674	23.71	0.6801	20.60	0.5181
	MADUN [23]	20.13	0.5007	20.01	0.5023	19.78	0.4787	19.22	0.4441	17.64	0.3986
	<i>Rob</i> -MADUN	28.11	0.8032	27.99	0.7988	26.48	0.7781	23.32	0.7072	18.77	0.4987
	TransCS [24]	22.32	0.5311	21.12	0.5188	21.01	0.4977	18.88	0.4732	16.78	0.4086
	<i>Rob</i> -TransCS	29.03	0.8192	28.55	0.8091	27.22	0.7979	23.36	0.7112	20.32	0.5210
	HGLISTA [22]	23.11	0.5431	22.07	0.5043	20.79	0.4842	19.64	0.4699	16.79	0.3486
	<i>Rob</i> -HGLISTA	31.03	0.9168	29.79	0.9053	27.00	0.8187	24.02	0.7072	20.44	0.5151

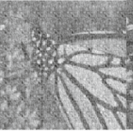
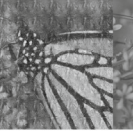
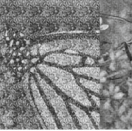
Ground Truth	Dr2-Net	Rob-Dr2	ISTA-Net ⁺	Rob-ISTA ⁺	CSNet	Rob-CSNet
						
PSNR SSIM	18.37 0.3911	22.55 0.5569	20.13 0.4665	28.20 0.9180	14.11 0.2511	24.24 0.6377
						
PSNR SSIM	18.14 0.2546	22.49 0.4840	19.78 0.3724	23.39 0.7810	17.25 0.2348	23.27 0.5718

Fig. 1. Reconstructions of ‘*Monarch*’ at MR of 25% and ‘*Cameraman*’ at MR of 10% in noisy condition with zero mean and standard deviation 0.10. The reconstruction results are reported in terms of PSNR (dB) and SSIM. Our methods with prefix ‘*Rob-*’ demonstrate the superior robustness to noise.

Note that we do not present results for various β_k and optimization intervals V , though they would affect the robustness and reconstruction performance of models. Similar to the benchmark methods, we also utilize the dataset *Set 11* for testing. The reconstruction results are reported in terms of the average Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) over the test images.

To demonstrate the robustness of our algorithm, we conduct experiments under different noise configurations. Table 1 summarizes the reconstruction performance with respect to five MRs under three noise levels, i.e., noiseless condition, Gaussian noise with zero mean and standard deviations (Stds) 0.05 and 0.10, respectively. The noises are added to each image block in the same manner, as shown in Fig. 1. Here, we use the prefix ‘*Rob-*’ to indicate the proposed algorithm.

Under noiseless conditions, our method significantly improves the PSNR and SSIM of ReconNet, Dr2-Net, ISTA-Net⁺, MADUN, TransCS, and HGLISTA by substituting the Gaussian sampling matrix with a learned measurement matrix. End-to-end network training provides superior performance compared to those networks that learn projections from measurements to the original image. Our method also achieves competitive recovery quality with *Adp-Recon* and CSNet, indicating that our algorithm does not degrade the performance of these end-to-end neural networks. When measurements are corrupted by noise with standard deviations (Stds) of 0.05 and 0.10, all baselines show evident performance degradation. Notably, the performance degradation is not consistently correlated with MRs; for instance, the performance of *Adp-Recon* and ISTA-Net⁺ does not improve with increasing MRs under Std 0.10 conditions. This fact implies the vulnerability and unpredictability of DNNs. In contrast to the baselines, our algorithm shows remarkable robustness against noise. Under noisy conditions, we achieve the best reconstruction performance. Figure 1 shows the reconstruc-

tions of ‘*Monarch*’ at an MR of 25% and ‘*Cameraman*’ at an MR of 10% under noisy conditions with Std 0.10, for further validation. Additionally, our algorithm tends to facilitate the end-to-end networks (*Adp-Recon* and CSNet) with a significant gain compared to the other benchmark methods. We observe that, compared to end-to-end CS networks, the random Gaussian matrix slightly improves robustness to noise at the cost of performance decay. Our algorithm benefits compressive sensing by enhancing both reconstruction performance and robustness.

5 Limitation

While the proposed method has demonstrated significant effectiveness in mitigating the impact of additive white Gaussian noise across the experiments conducted, it exhibits certain limitations when exposed to other forms of noise. Specifically, the algorithm’s performance may deteriorate when confronted with impulse noise, structured noise, or Poisson noise, which introduce unique challenges that fall outside the primary design of the algorithm.

Impulse noise, often characterized by abrupt and sporadic pixel value alterations, can result in severe image distortions that are not well-suited to the current framework of the method. The unpredictable nature of such noise necessitates specialized filtering techniques, which were not integrated into the proposed approach. Similarly, structured noise, which is typified by correlated disturbances that follow discernible patterns, presents additional complexities. The algorithm, optimized for handling uncorrelated noise, may struggle to detect and mitigate structured noise effectively, as it requires more advanced spatial or frequency domain processing strategies. Furthermore, Poisson noise, prevalent in photon-limited imaging scenarios, exhibits a fundamentally different statistical distribution compared to Gaussian noise. As a result, the proposed method’s assumption of additive noise properties may no longer hold, thereby diminishing its efficacy in these contexts.

Given these limitations, future research could focus on extending the algorithm’s robustness to address a broader spectrum of noise types. This could involve integrating adaptive or hybrid techniques tailored to the specific statistical and structural properties of impulse, structured, and Poisson noise. Additionally, further exploration into the balance between noise suppression and the preservation of critical image details is warranted, particularly in the presence of complex noise environments. Such enhancements could significantly broaden the applicability and effectiveness of the proposed method in real-world scenarios where multiple noise types coexist.

6 Conclusion

In this paper, we present a universal neural network optimization to improve the robustness of DNN-based CS methods. The proposed algorithm consists of two optimizations to guarantee stable sampling and robust reconstruction,

respectively. First, we introduce the approximate RIP to optimize the learned measurement matrix. Subsequently, we develop the upper bound of noise energy in Theorem 1 to ensure robustness in the reconstruction network. We provide a theoretical analysis for these optimizations. Extensive experimental evaluations show the robustness of our algorithm with no accuracy loss.

References

1. Donoho, D.L.: Compressed sensing. *IEEE Trans. Inf. Theory* **52**(4), 1289–1306 (2006)
2. Candes, E.J., Tao, T.: Decoding by linear programming. *IEEE Trans. Inf. Theory* **51**(12), 4203–4215 (2005)
3. Gan, L.: Block compressed sensing of natural images. In: 2007 15th International Conference on Digital Signal Processing, pp. 403–406. IEEE (2007)
4. Amini, A., Marvasti, F.: Deterministic construction of binary, bipolar, and ternary compressed sensing matrices. *IEEE Trans. Inf. Theory* **57**(4), 2360–2370 (2011)
5. Lu, W., Dai, T., Xia, S.T.: Binary matrices for compressed sensing. *IEEE Trans. Signal Process.* **66**(1), 77–85 (2017)
6. Dinh, K.Q., Shim, H.J., Jeon, B.: Measurement coding for compressive imaging using a structural measurement matrix. In: 2013 IEEE International Conference on Image Processing, pp. 10–13. IEEE (2013)
7. Gao, X., Zhang, J., Che, W., Fan, X., Zhao, D.: Block-based compressive sensing coding of natural images by local structural measurement matrix. In: 2015 Data Compressing Conference, pp. 133–142. IEEE (2015)
8. Wright, S.J., Nowak, R.D., Figueiredo, M.A.: Sparse reconstruction by separable approximation. *IEEE Trans. Signal Process.* **57**(7), 2479–2493 (2009)
9. Figueiredo, M.A., Nowak, R.D., Wright, S.J.: Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Signal Process.* **1**(4), 586–597 (2007)
10. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inf. Theory* **53**(12), 4655–4666 (2007)
11. Haupt, J., Nowak, R.: Signal reconstruction from noisy random projections. *IEEE Trans. Inf. Theory* **52**(9), 4036–4048 (2006)
12. Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., Ashok, A.: ReconNet: non-iterative reconstruction of images from compressively sensed measurements. In: IEEE Conference on Computing Vision Pattern Recognition, pp. 449–458 (2016)
13. Mousavi, A., Patel, A.B., Baraniuk, R.G.: A deep learning approach to structured signal recovery. In: Annual Allerton Conference on Communication Control Computing, pp. 1336–1343. IEEE (2015)
14. Yao, H., Dai, F., Zhang, S., Zhang, Y., Tian, Q., Xu, C.: Dr2-net: deep residual reconstruction network for image compressive sensing. *Neurocomputing* **359**, 483–493 (2019)
15. Zhang, J., Ghanem, B.: ISTA-net: interpretable optimization-inspired deep network for image compressive sensing. In: IEEE Conference on Computing Vision Pattern Recognition, pp. 1828–1837 (2018)
16. Xie, X., Wang, Y., Shi, G., Wang, C., Du, J., Han, X.: Adaptive Measurement Network for CS Image Reconstruction. In: Yang, J., et al. (eds.) CCCV 2017. CCIS, vol. 772, pp. 407–417. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-7302-1_34

17. Adler, A., Boubilil, D., Zibulevsky, M.: Block-based compressed sensing of images via deep learning. In: International Workshop Multimedia Signal Processing, pp. 1–6. IEEE (2017)
18. Mousavi, A., Baraniuk, R.G.: Learning to invert: signal recovery via deep convolutional networks. In: IEEE International Conference Acoustics, Speech Signal Processing, pp. 2272–2276. IEEE (2017)
19. Shi, W., Jiang, F., Zhang, S., Zhao, D.: Deep networks for compressed image sensing. In: IEEE International Conference Multimedia and Expo, pp. 877–882. IEEE (2017)
20. Shi, W., Jiang, F., Liu, S., Zhao, D.: Scalable convolutional neural network for image compressed sensing. In: IEEE Conference on Computing Vision Pattern Recognition (2019)
21. Shi, W., Jiang, F., Liu, S., Zhao, D.: Image compressed sensing using convolutional neural network. *IEEE Trans. Image Process.* **29**, 375–388 (2019)
22. Zheng, Z., Dai, W., Xue, D., Li, C., Zou, J., Xiong, H.: Hybrid ISTA: unfolding ISTA with convergence guarantees using free-form deep neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(3), 3226–3244 (2023)
23. Song, J., Chen, B., Zhang, J.: Memory-augmented deep unfolding network for compressive sensing. In: ACM International Conference on Multimedia, pp. 4249–4258 (2021)
24. Shen, M., Gan, H., Ning, C., Hua, Y., Zhang, T.: Transcs: a transformer-based hybrid architecture for image compressed sensing. *IEEE Trans. Image Process.* **31**, 6991–7005 (2022)
25. Zhang, K., Hua, Z., Li, Y., Chen, Y., Zhou, Y.: AMS-net: adaptive multi-scale network for image compressive sensing. *IEEE Trans. Multimedia* **25**, 5676–5689 (2022)
26. Mun, S., Fowler, J.E.: Block compressed sensing of images using directional transforms. In: IEEE International Conference on Image Processing, pp. 3021–3024. IEEE (2009)
27. He, L., Carin, L.: Exploiting structure in wavelet-based Bayesian compressive sensing. *IEEE Trans. Signal Process.* **57**(9), 3488–3497 (2009)
28. Lu, Y.M., Do, M.N.: A theory for sampling signals from a union of subspaces. *IEEE Trans. Signal Process.* **56**(6), 2334–2345 (2008)
29. Zhang, J., Zhao, D., Zhao, C., Xiong, R., Ma, S., Gao, W.: Compressed sensing recovery via collaborative sparsity. In: Data Compression Conference, pp. 287–296. IEEE (2012)
30. Metzler, C.A., Maleki, A., Baraniuk, R.G.: From denoising to compressed sensing. *IEEE Trans. Inf. Theory* **62**(9), 5117–5144 (2016)
31. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Trans. Image Process.* **16**(8), 2080–2095 (2007)
32. Ma, J., Plonka, G., Hussaini, M.Y.: Compressive video sampling with approximate message passing decoding. *IEEE Trans. Circuits Syst. Video Technol.* **22**(9), 1354–1364 (2012)
33. Azghani, M., Karimi, M., Marvasti, F.: Multihypothesis compressed video sensing technique. *IEEE Trans. Circuits Syst. Video Technol.* **26**(4), 627–635 (2015)
34. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. In: *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122 (2011)

35. Dinh, K.Q., Jeon, B.: Iterative weighted recovery for block-based compressive sensing of image/video at a low subrate. *IEEE Trans. Circuits Syst. Video Technol.* **27**(11), 2294–2308 (2016)
36. Eicke, B.: Iteration methods for convexly constrained ill-posed problems in Hilbert space. *Numer. Funct. Anal. Optim.* **13**(5–6), 413–429 (1992)
37. Duarte, M.F., Baraniuk, R.G.: Kronecker product matrices for compressive sensing. In: *IEEE International Conference on Acoustics, Speech Signal Processing*, pp. 3650–3653. IEEE (2010)
38. Ullrich, K., Meeds, E., Welling, M.: Soft weight-sharing for neural network compression. In: *International Conference on Learning Representation* (2017)
39. Xie, D., Xiong, J., Pu, S.: All you need is beyond a good INIT: exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. In: *IEEE Conference on Computing Vision Pattern Recognition*, pp. 6176–6185 (2017)



Joint Sensing of Computing Power and Network for Smart Grid: A Deep Matrix Completion Approach

Yuxiang Qiu^(✉), Lei Zhang, Yuan Zhu, and Lan Gan

Nari Information and Communication Technology Co. Ltd., Nanjing, China
{qiuyuxiang,zhanglei2,zhuyuan,ganlan}@sgepri.sgcc.com.cn

Abstract. As the scale of power grid business continues to expand, smart grids are rapidly emerging as an alternative to traditional power grids. In order to realize the efficient processing of power grid data, cloud computing and edge computing have been widely used in smart grid. However, how to realize the unified arrangement between edge nodes and cloud nodes also needs to be solved. To this end, Compute First Networking (CFN) was proposed, whose goal is to achieve optimal and efficient use of network and computing resources. However, how to efficiently obtain computing and network resource status in the smart grid is a major challenge, due to equipment damage, disconnection or even attack. Therefore, this work focuses on reconstructing complete computing and network resource information under incomplete status in smart grids and propose a resource sensing method based on deep matrix completion which can effectively obtain computing and network resource information. The experimental results evaluate the accuracy of our proposed method.

Keywords: Smart grids · Compute first networking · Computing power sensing · Network resource sensing · Deep matrix completion

1 Introduction

Power grid is the main stage of power system to supply power to users. It includes power generation, power transmission network, regional power distribution network and power consumption. It has the characteristics of wide geographical distribution, many kinds of equipment and variable operation mode. As power demand escalates and the scope of power grid operations expands, traditional grids are struggling to fulfill contemporary requirements. The concept of a smart grid, illustrated in Fig. 1, has emerged. This integrates cutting-edge networking and artificial intelligence technologies to develop an efficient and intelligent power distribution system [2,3,6,11,13]. To achieve effective management of grid data, cloud computing has become prevalent in smart grid applications [8]. Despite offering substantial computational resources, cloud computing falls short in addressing the need for real-time data processing within the grid due

to latency issues among cloud nodes [20]. Transmission delays are eased with the advent of edge computing. However, edge nodes have limited computing resources and cannot effectively handle compute-intensive tasks in smart grids. At the same time, how to realize the unified orchestration between edge nodes and cloud nodes also needs to be solved.

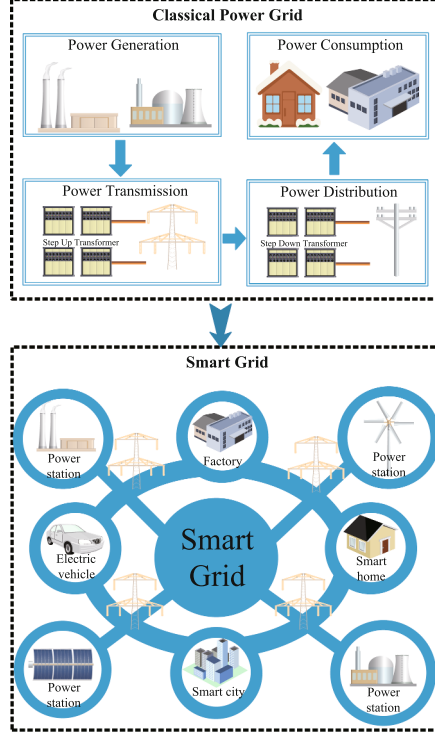


Fig. 1. Comparison of classical power grid and smart grid.

In order to solve the above problems, Compute First Networking (CFN) is proposed. The main goal of CFN is to realize the interconnection and scheduling of compute nodes across the network, so as to realize the optimization and efficient use of network resources and computing resources. In order to realize the unified scheduling of multi-dimensional resources, CFN first needs to obtain the corresponding resource status. However, the ongoing expansion of the current power grid and the increasing complexity of its structure lead to significant issues. Information and data within the grid can become highly fragmented or may be lost due to equipment failures [10]. Concurrently, the uneven distribution of computing nodes within the smart grid, coupled with high communication latencies, hampers the accurate acquisition of node resource information.

Therefore, how to obtain accurate and complete multi-dimensional resource information (computing and network resource information) in a complex smart grid is a major challenge. In this paper, we try to solve the above challenge. Based on this observation that there are certain temporal and spatial characteristics of device node resources in smart grid, for example, there are more power requests in office areas during workdays and more power requests in residential areas during breaks, an intuitive idea is to use these characteristics to infer the missing status information. Given the strengths of deep learning in feature extraction, we attempt to utilize deep matrix factorization to infer the absent state information. Consequently, we introduce a resource sensing methodology grounded in deep matrix factorization (DMF), capable of efficiently acquiring computational and network resource details within smart grids. Our contributions can be summarized as follows:

- We leverage deep networks to extract both temporal and spatial features from incomplete resource datasets, applying matrix factorization to accurately infer the missing computing and network resource information.
- We perform comprehensive evaluations using real-world network traffic and computational power datasets. The experimental outcomes demonstrate that our proposed algorithm achieves superior reconstruction accuracy.

2 Related Work

The smart grid facilitates bidirectional data flow between utility providers and consumers [1, 4]. Extensive research has been conducted on the development of both hardware and software components within the smart grid ecosystem. For instance, [17] has developed a load balancing algorithm utilizing reinforcement learning. In [15], a smart meter supporting edge computing is introduced to enhance overall system performance. Reference [16] gives the outline of the methodology for ensuring the interoperability of smart sensors within smart grids. A cyber-physical framework designed for real-time control and monitoring of smart grids is presented in [5]. An algorithm for state estimation specifically for edge computing nodes located at primary substations is described in [14]. An edge computing framework [12] aimed at supporting intelligent monitoring in smart grids is designed as discussed. Reference [19] proposes a new global pruning technique intended to compress and speed up models for efficient edge computing.

Despite these advancements, achieving unified orchestration between edge and cloud nodes, along with improving the overall utilization of computing resources, remains a challenge within smart grid environments. To address these issues, a novel architectural approach known as Compute First Networking (CFN) has been proposed to manage multi-dimensional resource allocation [9]. Nevertheless, further research into the application of CFN within smart grids is needed. This paper aims to fill this gap by exploring methods to acquire precise and comprehensive resource information within smart grids.

3 Preliminaries

In this section, we will define the problem that needs to be addressed. To clarify the issue and simplify its description, we begin by introducing relevant concepts and notations. Following this, we will formally outline the problem.

3.1 Definitions

To simplify the model analysis, we streamline the data upload process. We assume that each compute node in the smart grid consistently uploads its resource status information. The upload frequency is established beforehand by experts. The uploaded data is organized by node and upload time, with the resource status information stored in a matrix. We then model this issue as a Matrix Completion (MC) problem. To facilitate a better understanding of this paper, we first define some key concepts related to Matrix Completion.

Definition 1 (Ground Truth). *For a resource data matrix (whether related to network or computing), indexed by node number and upload time, we denote its ground truth matrix as $U_{m \times n}$. Each entry $U(i, j)$ represents the resource information uploaded by the i -th node at time j , where m, n denotes the upper and lower bounds of the matrix.*

Definition 2 (Indicator). *In the indicator matrix $\Omega_{m \times n}$, each entry $\Omega(i, j)$ indicates whether the corresponding entry in $U_{m \times n}$ is selected for upload. When $U(i, j)$ is selected, $\Omega(i, j) = 1$; Otherwise, $\Omega(i, j) = 0$. The matrix $\hat{\Omega}$ is used to represent the opposite status.*

Definition 3 (Upload). *An upload matrix $X_{m \times n}$ represents the actual uploaded data, where $\bullet$ signifies the Hadamard product, which refers to the element-wise multiplication:*

$$X = \Omega \bullet U,$$

Definition 4 (Complete Matrix). *Ψ is a missing value completion algorithm that seeks to reconstruct a complete matrix $\tilde{U}_{m \times n}$ from the upload matrix $X_{m \times n}$ by leveraging spatio-temporal characteristics, as expressed in the following equation:*

$$\Psi(X_{m \times n}) = \tilde{U} \approx U.$$

The reconstructed error, denoted by ε , quantifies the discrepancy between the reconstructed matrix $\tilde{U}$ and the ground-truth matrix U . The error can be expressed as $\varepsilon = \text{error}(\tilde{U}, U)$.

3.2 Problem Formulation

Based on the previous discussion, the research problem can be defined as follows: In the context of multidimensional resource sensing tasks within the smart grid, where each compute node is responsible for uploading only its own resource data, the objective is to determine how to utilize the collected incomplete data X along with the data reconstruction model Ψ , which is derived from spatio-temporal characteristics, to accurately reconstruct a complete resource dataset. Furthermore, it is essential to minimize the reconstruction error to the greatest extent possible:

$$\begin{aligned} \min \quad & \varepsilon = \text{error}(\tilde{U}, U) \\ \text{s.t.} \quad & \tilde{U} = \Psi(X_{m \times n}), \\ & X = \Omega \bullet U. \end{aligned} \tag{1}$$

4 Methodology and Performance Evaluations

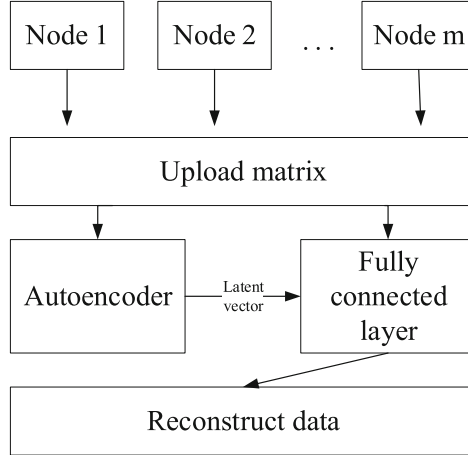


Fig. 2. The basic workflow of ADMF.

4.1 Methodology

The standard DMF approach [7], necessitates knowing the rank of the matrix for data inference; however, in practical scenarios, the exact rank is often unknown beforehand. Thus, our proposed algorithm consists of two main stages. First, Adaptive Deep Matrix Factorization (ADMF) utilizes an autoencoder to extract latent variables from the data matrix, thereby estimating the optimal rank of the

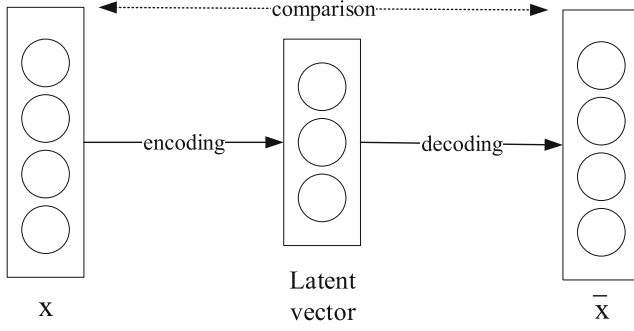


Fig. 3. The basic workflow of autoencoder.

matrix. Subsequently, ADMF employs DMF to reconstruct the complete data matrix from the latent variables and the estimated rank, as illustrated in Fig. 2.

An autoencoder is a basic neural network model designed to use backpropagation to ensure that the output closely matches the input. This process involves compressing the input into a set of latent features, followed by reconstructing the output from these features. The autoencoder consists of two primary components: an encoder and a decoder (as depicted in Fig. 3).

The encoder's task is to extract latent features $z \in \mathbb{R}^d$ (where d is the dimension of the latent feature space) from the input $x \in \mathbb{R}^N$, with $N \gg d$: $z = f(x) \Leftrightarrow \sigma^{(1)}(W_e x + b_e)$, in which $f(\cdot)$ denotes a nonlinear mapping.

On the contrary, the decoder transforms the latent feature vector z extracted by the encoder back into the reconstructed data $\hat{x} \in \mathbb{R}^N$: $\hat{x} = f^T(z) \Leftrightarrow \sigma^{(2)}(W_d z + b_d)$. $f^T(\cdot)$ is an approximation for the inverse function of $f(\cdot)$. $f^T(z)$ attempts to reverse the transformation applied by f to reconstruct the original input x as closely as possible.

However, the autoencoder cannot be directly applied to our problem because the uploaded data matrix is incomplete. Therefore, in this paper, we utilize its variant, denoising autoencoder. Data incompleteness is effectively avoided by introducing an indicator matrix, whose structure is shown in Fig. 4.

To reduce the model parameters, we assume $W_d = W_e^T$. The loss function can be equivalent to the following formula:

$$\min \frac{1}{m-1} \sum_{i=1}^{m-1} (\Omega_i \bullet \|x_i - \sigma^{(2)}(W_d^T \sigma^{(1)}(W_e x_i + b_e^{(1)}) + b_d^{(2)})\|^2) + \frac{\lambda_e}{2} \|W_e\|_F^2, \quad (2)$$

where $\sigma^{(1)}$ and $\sigma^{(2)}$ are the activation functions. W_e and W_d are the weight matrices. b_e and b_d are the vectors of bias terms of the encoder layer and decoder layer respectively. λ_e is an algorithm hyper-parameter.

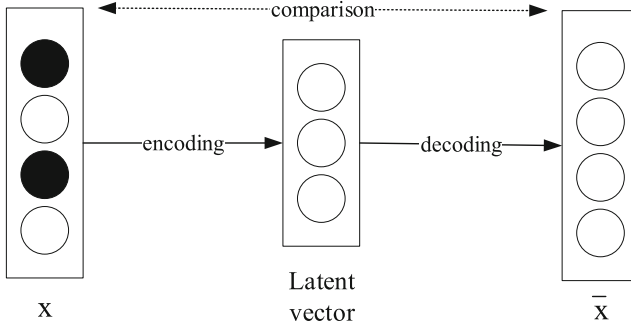


Fig. 4. The basic workflow of denoising autoencoder.

Considering that there are a number of heterogeneous devices in the environment of smart grids, we need to normalize the data with .

$$x = \frac{x - \min}{\max - \min}, \quad (3)$$

where x is the original data uploaded, $\max$ and $\min$ are the maximum and minimum values respectively. We then process it as input to the denoising autoencoder. Finally, we estimate the rank of the matrix r by using the dimension of the latent variable z , meaning $r = d$.

The deep matrix factorization (DMF) is a variant of the matrix factorization, the traditional matrix factorization usually assumes that the matrix is low-rank, that is, the matrix can be obtained by a linear transformation as follows,

$$X = PZ, \quad (4)$$

where P is a linear transformation. As a result, traditional matrix factorization can only extract linear features and cannot capture nonlinear features.

Therefore, the deep matrix factorization assumes that the matrix transformation is a nonlinear transformation.

$$X = g(Z), \quad (5)$$

where $g()$ is a nonlinear function. In order to capture nonlinear features fully, DMF adopts a fully connected neural network structure, as shown in Fig. 5.

Therefore, for a single hidden layer, the objective function of DMF is as follows,

$$\begin{aligned} \min_{z, W, b} \frac{1}{2n} \sum_{i=1}^n \|\Omega_i \bullet (x_i - g(Wz_i + b))\|^2 &+ \frac{\beta}{2n} \|Z\|_F^2 \\ &+ \frac{\lambda}{2} \|W\|_F^2 \end{aligned} \quad (6)$$

where $Z \in \mathbb{R}^{r \times n}$ is latent matrix which can be randomly initialized, z_i is the entry of Z , β, λ are hyperparameters which can be set in advance.

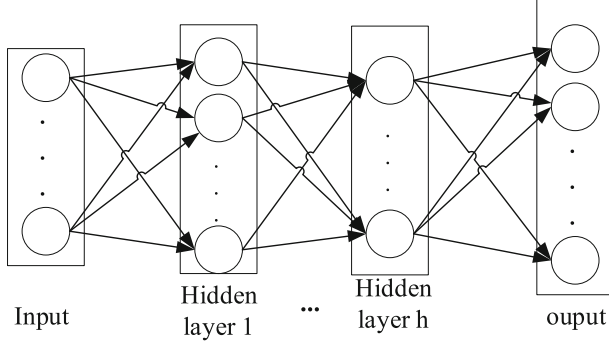


Fig. 5. The fully connected layer.

We denote the weight matrix as W and denote the vector of bias as b . For K hidden layers, the objective function of DMF is defined as:

$$L = \min_{Z, W, b} (L_1 + L_2), \quad (7)$$

where L_1 and L_2 are defined by the following equation,

$$L_1 = \frac{1}{2n} \sum_{i=1}^n \|\Omega_i \bullet (x_i - g^{(K+1)}(W^{(K+1)} \dots g^{(1)}(W^{(1)} z_i + b^{(1)}) \dots + b^{(K+1)}))\|^2,$$

$$L_2 = \frac{\beta}{2n} \|Z\|_F^2 + \frac{\lambda}{2} \sum_{j=1}^{K+1} \|W^{(j)}\|_F^2.$$

In summary, DMF can effectively capture nonlinear features through the use of fully connected layers, thereby enhancing the accuracy of the reconstructed data matrix. The autoencoder is utilized to extract the rank and latent variables from the incomplete multidimensional resource data matrix. Subsequently, DMF leverages these latent variables to extract nonlinear features and reconstruct the complete data matrix. The entire algorithmic workflow of ADMF is detailed in Algorithm 1.

4.2 Experimental Evaluation

Experiment Setup. We selected the real-world network traffic datasets¹ and the computing power data from 30 lab devices. For network traffic data, we randomly generate corresponding traffic data for each device. For computing

¹ <https://www.kaggle.com/c/web-traffic-time-series-forecasting/data/>.

Algorithm 1. The ADMF algorithm

Input: incomplete resource data $\tilde{\mathbf{X}}$;**Output:** Reconstructed complete data $\tilde{X}$;**The Autoencoder Phase**

- 1) Initialization parameters W_e, b_e, b_d ;
- 2) Normalize input by equation (3)
- 3) Train autoencoder by equation (2);
- 4) Get latent vector and estimation of matrix rank;

The DMF Recovery Phase

- 1) Initialization the latent matrix randomly from the autoencoder phase;
 - 2) Initialization fully connected layer network parameters;
 - 3) Reconstruct complete data matrix by equation (7);
-

power data, we randomly generate available computing resource information based on device performance.

Baselines: To evaluate the performance of our proposed algorithm, we compare it against state-of-the-art missing value inference algorithm LMaFit [18]. LmaFit is a widely-used alternating least-squares method for matrix completion. It determines the optimal matrix rank r for each signal map to ensure optimal performance, making it a robust choice for recovering missing values in data matrices. By carefully selecting the rank r , LmaFit aims to provide accurate reconstructions of the original matrix before any data was lost or became unavailable.

We compared the reconstruction errors at various missing rates (ranging from 10% to 90%). To assess the reconstruction errors, the Mean Average Error (MAE) and the precision are used as the evaluation metrics:

$$MAE = \frac{\|\hat{\Omega} \bullet (\tilde{X} - X)\|_F}{\|\hat{\Omega} \bullet (1 - M)\|_F},$$

and

$$Precision = 1 - \frac{\|\hat{\Omega} \bullet (\tilde{M} - M)\|_F}{\|X\|_F}.$$

Given that LmaFit has the capability to estimate the rank of the matrix, to ensure a fair and consistent comparison, we opt not to use the autoencoder to estimate the rank of the matrix but rather use the same rank as determined by LmaFit. For the parameters used in DMF, we adopt the best parameters identified in [7] (Fig. 7).

Experiment Result. For network data, as depicted in Fig. 6, it is evident that the Mean Absolute Error (MAE) of our proposed algorithm is consistently lower compared to LmaFit across different missing rates. This indicates that our algorithm performs better in reconstructing the missing values of network traffic data.

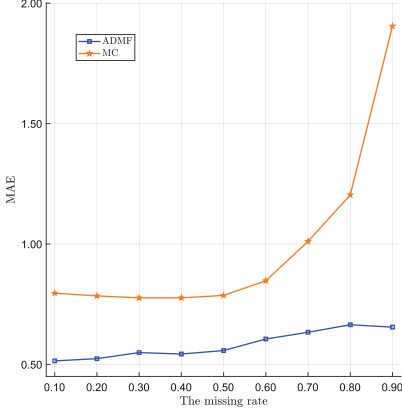


Fig. 6. The MAE of network data.

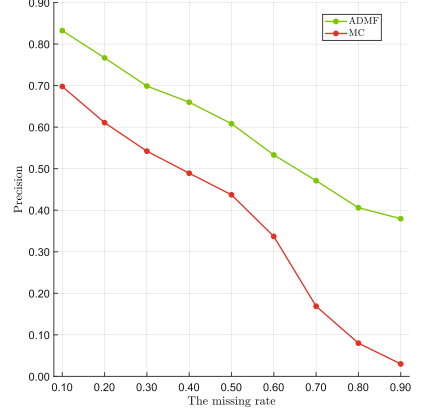


Fig. 7. The precision of network data.

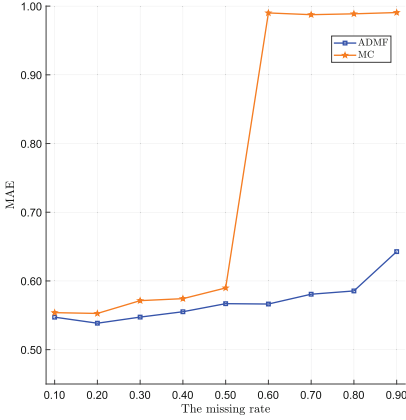


Fig. 8. The MAE of computing data.

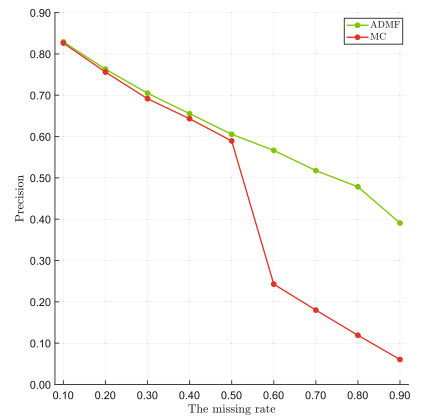


Fig. 9. The precision of computing data.

For computing data, Figs. 8 and 9 illustrate that our algorithm outperforms LmaFit. As the missing rate increases, the performance of LmaFit degrades more noticeably, particularly when the missing rate surpasses 0.5. In contrast, our algorithm demonstrates better robustness to increasing missing rates. This implies that even when more than half of the data is missing, our algorithm maintains a relatively stable performance.

For network-computing mixed data, Figs. 10 and 11 demonstrate that our algorithm consistently outperforms LmaFit. Our algorithm shows a lower error rate across different levels of missing data compared to LmaFit. As the missing rate increases, LmaFit's performance becomes unstable, exhibiting oscillations when the missing rate exceeds 0.5. In contrast, our algorithm maintains a more stable performance.

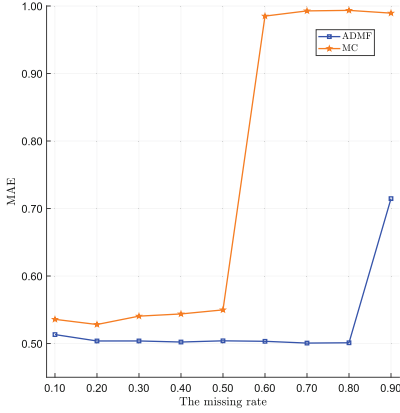


Fig. 10. The MAE of Network-computing data.

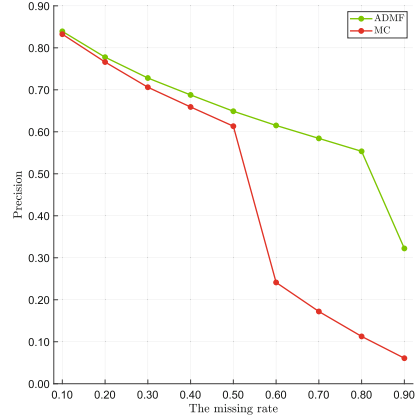


Fig. 11. The precision of Network-computing data.

In summary, our algorithm can effectively recover resource information in smart grid, compared with traditional missing value inference algorithm. At the same time, the experimental results show that our algorithm has strong robustness and can deal with extreme situations, such as the missing rate >0.5 .

5 Conclusion

We present a comprehensive approach for resource data recovery in smart grids, utilizing deep matrix decomposition to capture the temporal-spatial correlations inherent in resource data to facilitate the recovery of missing information. To efficiently and accurately determine the matrix rank required for the Deep Matrix Factorization (DMF), we apply a denoising autoencoder to compress the resource data and use the dimensionality of the latent variables as an optimal estimate for the matrix rank. Finally, we validate the effectiveness of our proposed algorithm using real-world network traffic and computing power data. The experimental results demonstrate that our algorithm is sufficiently robust to address the complexities encountered in smart grid scenarios.

Acknowledgment. This work was supported by the Key Technologies and Applications of Resource Collaborative Scheduling for Cloud-Edge-Network Integrated System Project of State Grid Corporation of China under Grant no. 5700-202318292A-1-1-ZN.

References

1. Albataineh, H., Nijim, M., Bollampall, D.: The design of a novel smart home control system using smart grid based on edge and cloud computing. In: Proceedings of the 8th International Conference on Smart Energy Grid Engineering, pp. 88–91 (2020). <https://doi.org/10.1109/SEGE49949.2020.9181961>

2. Almerab, M.A.: Smart grid services for energy utilities: insights on options and priorities and main application enabler in national grid SA. In: Proceedings of the 2022 Saudi Arabia Smart Grid, pp. 1–5 (2022). <https://doi.org/10.1109/SASG57022.2022.10199644>
3. Chen, H., et al.: Research on the transmission relationship of grid science and technology innovation indicators under the smart grid. In: Proceedings of the 2nd International Conference on Smart Grids and Energy Systems, pp. 19–22 (2023). <https://doi.org/10.1109/SGES59720.2023.10366922>
4. COLAK, I., BAYINDIR, R., SAGIROGLU, S.: The effects of the smart grid system on the national grids. In: Proceedings of the 8th International Conference on Smart Grid. pp. 122–126 (2020). <https://doi.org/10.1109/icSmartGrid49881.2020.9144891>
5. Dey, A., Chakraborty, S., Salapaka, M.V.: An end-to-end cyber-physical infrastructure for smart grid control and monitoring. In: Proceedings of the IEEE Power & Energy Society Innovative Smart Grid Technologies Conference, pp. 1–5 (2022). <https://doi.org/10.1109/ISGT50606.2022.9817539>
6. El-Dessouki, I., Saeed, N.: Smart grid integration into smart cities. In: Proceedings of the 2021 IEEE International Smart Cities Conference, pp. 1–4 (2021). <https://doi.org/10.1109/ISC253183.2021.9562769>
7. Fan, J., Cheng, J.: Matrix completion by deep matrix factorization. *Neural Netw.* **98**, 34–41 (2018)
8. Feng, G., Huang, Q., Deng, Z., Zou, H., Zhang, J.: Research on cloud security construction of power grid in smart era. In: Proceedings of the IEEE 2nd International Conference on Data Science and Computer Application, pp. 976–980 (2022). <https://doi.org/10.1109/ICDSCA56264.2022.9987863>
9. Gong, X., Bai, C., Ren, S., Wang, J., Wang, C.: A survey of compute first networking. In: Proceedings of the 23rd International Conference on Communication Technology, pp. 688–695 (2023). <https://doi.org/10.1109/ICCT59356.2023.10419572>
10. Haghshenas, S.H., Naeini, M.: Topology attack detection in smart grids through learning graph signal smoothness. In: Proceedings of the 2024 IEEE Green Technologies Conference, pp. 54–58 (2024). <https://doi.org/10.1109/GreenTech58819.2024.10520401>
11. Heck, G.C., Hexsel, R., Gomes, V.B., Iantorno, L., Junior, L.L., Santana, T.: Grid-city: a framework to share smart grids communication with smart city applications. In: Proceedings of the 2021 IEEE International Smart Cities Conference, pp. 1–4 (2021). <https://doi.org/10.1109/ISC253183.2021.9562794>
12. Huang, Y., Lu, Y., Wang, F., Fan, X., Liu, J., Leung, V.C.: An edge computing framework for real-time monitoring in smart grid. In: Proceedings of the IEEE International Conference on Industrial Internet, pp. 99–108 (2018). <https://doi.org/10.1109/ICII.2018.00019>
13. Routray, S.K., Gopal, D., Javali, A., Sahoo, A.: Narrowband IoT (nbiot) assisted smart grids. In: Proceedings of the International Conference on Artificial Intelligence and Smart Systems, pp. 1454–1458 (2021). <https://doi.org/10.1109/ICAIS50930.2021.9395891>
14. Ruuth, K., Mutanen, A.: Demonstrating state estimation for future smart grid. In: Proceedings of the 2023 IEEE PES Innovative Smart Grid Technologies Europe, pp. 1–6 (2023). <https://doi.org/10.1109/ISGTEUROPE56780.2023.10407731>
15. Sirojan, T., Lu, S., Phung, B.T., Ambikairajah, E.: Embedded edge computing for real-time smart meter data analytics. In: Proceedings of the International Conference on Smart Energy Systems and Technologies, pp. 1–5 (2019). <https://doi.org/10.1109/SEST.2019.8849012>

16. Song, E.Y., FitzPatrick, G.J., Lee, K.B., Griffor, E.: A methodology for modeling interoperability of smart sensors in smart grids. *IEEE Trans. Smart Grid* **13**(1), 555–563 (2022). <https://doi.org/10.1109/TSG.2021.3124490>
17. Thapa, R., Jiao, L., Oommen, B.J., Yazidi, A.: A learning automaton-based scheme for scheduling domestic shiftable loads in smart grids. *IEEE Access* **6**, 5348–5361 (2018). <https://doi.org/10.1109/ACCESS.2017.2788051>
18. Wen, Z.: Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Math. Program. Comput.* **4**(4), 333–361 (2012)
19. Yu, F., Cui, L., Wang, P., Han, C., Huang, R., Huang, X.: Easiedge: a novel global deep neural networks pruning method for efficient edge computing. *IEEE Internet Things J.* **8**(3), 1259–1271 (2021). <https://doi.org/10.1109/JIOT.2020.3034925>
20. Zhao, J., Xiang, M.M., Song, X., Tian, C., Yang, Y.: The application and threat of the internet of things in the smart grid. In: *Proceedings of the 2nd International Conference on Applied Machine Learning*, pp. 339–343 (2020). <https://doi.org/10.1109/ICAML51583.2020.00075>



Rethinking RSSI-Based Key Extraction for UAVs and Ground Stations

Yanbo Li^(✉)  and Jain Huang^{}

College of Computer Science and Technology, Nanjing University of Aeronautics and
Astronautics, Nanjing 211106, China
{lyanbo, ellison.huang}@nuaa.edu.cn

Abstract. Existing key extraction methods fail to work in dynamic unmanned aerial vehicles (UAVs) scenarios. This paper proposes a new key extraction framework, where a clock adjustment algorithm and a Canonical Correlation Analysis (CCA)-based sliding window smoothing method are employed to mitigate noise and enhance the reliability of the RSSI data. The Level Crossing Algorithm (LCA) is optimized for the dynamic and high-mobility UAV environment. A grid-search method is adopted to find the optimal parameters. Extensive experiments in indoor and outdoor scenarios, including real-world UAV flights, demonstrate the effectiveness of our method.

Keywords: Unmanned aerial vehicles · Key extraction · Received signal strength indicator · Canonical Correlation Analysis · Level crossing algorithm

1 Introduction

Unmanned aerial vehicles (UAVs) often involve the transmission of sensitive data, such as real-time videos, control commands, and telemetry information, which could have severe consequences if intercepted or tampered with. Hence, ensuring secure communication channels is essential to protect this data from unauthorized access and malicious threats when facing the inherent broadcast nature of wireless communication channels. Key establishment provides the foundation for secure communication, thus becoming an important issue.

Traditional security mechanisms, such as public key cryptography (PKC) and symmetric encryption, often rely on computational complexity to protect data. Recent research has shifted towards leveraging the physical layer characteristics of wireless channels to enhance communication security. This approach, rooted in information theory [4, 12], exploits the unique, reciprocal, and time-varying properties of wireless channels to generate shared cryptographic keys without the need for traditional key distribution mechanisms. While promising, existing methods have predominantly focused on static or low-mobility environments, such as indoor settings, low-speed scenarios, and vehicular networks. These methods, however, fall short when applied to the dynamic and high-mobility context of UAVs.

The primary limitations of existing works include: 1) **High mobility:** Current key extraction schemes are tailored for environments with minimal movement and stable signal conditions. In contrast, UAVs operate in highly dynamic environments with rapid changes in signal strength and frequent movement, making these schemes less effective. 2) **Synchronization Issues:** Many existing methods fail to account for the delays in packet transmission and reception caused by hardware and software limitations. These delays result in asynchronous sampling of the wireless channel, reducing the accuracy and reliability of the extracted keys.

To address these challenges, this paper proposes an improved key extraction method tailored specifically for UAV and ground station communication. The proposed approach addresses the challenges of dynamic and noisy environments by implementing a robust packet transmission and reception mechanism for near-simultaneous RSSI measurements. It introduces a Canonical Correlation Analysis (CCA)-based sliding window smoothing method to reduce noise, and utilizes the Level Crossing Algorithm (LCA) for optimized key extraction. Key extraction parameters are fine-tuned using a grid-search method, and a comprehensive information reconciliation protocol coupled with privacy amplification ensures the generation of high-entropy, secure keys. Extensive experimental validation in both indoor and outdoor scenarios demonstrates the effectiveness and robustness of the proposed method, making it a significant advancement in UAV communication security.

To the best of our knowledge, we are the first to consider key extraction in UAV environments using RSSI values. The main contributions of this work are listed as follows:

- We developed an optimized LCA framework for extracting cryptographic keys from RSSI in UAV-ground station communication, with an innovative clock synchronization scheme and a CCA-based sliding window smoothing method, coupled with BIC-based privacy amplification, to ensure the generation of high-entropy, secure keys.
- We implemented the proposed methods on real UAVs. Detailed experimental validation of the proposed method in realistic UAV scenarios, demonstrating its practicality and effectiveness in enhancing communication security.

2 Prerequisite

To better understand the problems, we describe some background. As shown in Fig. 1, when a network application injects a packet into a network interface, the packet is delivered from user space to kernel space and then reaches the network card. After the above latency C_{tx} , the network card waits for the channel to be idle and then transmits the packet in the air. If the transmission incurs errors due to wireless interference, the packet will be retransmitted and discarded after several failed attempts. Such a delay d_{tx} may vary and is unknown every time.

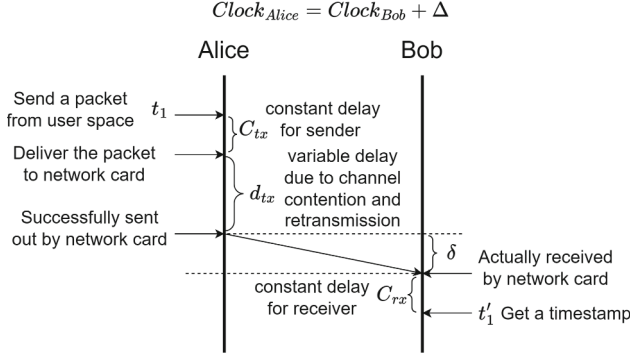


Fig. 1. Packet journal from sender to receiver and corresponding timestamps.

It is impossible to modify a commercial network card to record time right before the packet is actually being sent out in practice.

After a packet has been successfully sent out, the receiver will wait for the transmission delay δ (which is affected by the data rate and packet size) to capture the packet in his network card. There is also a gap C_{rx} between the reception time and the time when the packet is timestamped. In addition, there is a clock difference Δ between the sender and receiver. It is important to adjust Δ by either Alice or Bob before the key extraction. The entire packet traversal from the sender to receiver can be formulated in Eq.(1).

$$t_i + C_{tx} + d_{tx}^i + \delta = t'_i - C_{rx} + \Delta \quad (1)$$

3 System Design

The proposed system consists of two primary components: the UAV (Alice) and the ground station (Bob). Both entities are equipped with a wireless card capable of measuring the Received Signal Strength Indicator (RSSI) of transmitted packets. The overall architecture is designed to ensure synchronized RSSI measurement, efficient key extraction, and robust security against potential eavesdroppers. As shown in Fig. 2, the key components of our system are as follows:

3.1 Packet Transmission and Reception

To achieve near-simultaneous RSSI measurement, a UAV (Alice) and a ground station (Bob) are configured to continuously send and receive packets to each other in a fixed period T . Each packet is embedded with the sender's increasing sequence number i and the sender's tx timestamp t_i .

Both parties are equipped with wireless cards set to monitor mode, enabling them to capture RSSI values from received packets. Upon receiving a packet, each device records the RSSI value and the corresponding timestamp t'_i . This process generates two sets of data records: one for Alice and one for Bob.

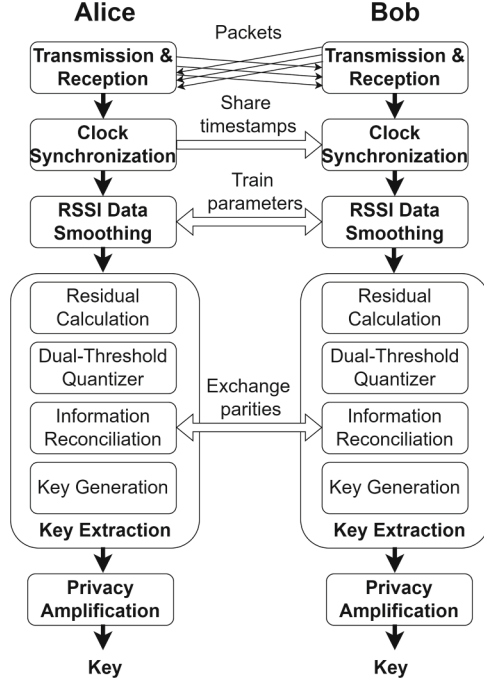


Fig. 2. Overall system architecture with a UAV and a ground station.

Note that it is crucial that Alice and Bob send packets independently without waiting for responses from each other. This independence ensures that the devices can continually measure the RSSI without synchronization delays caused by waiting for acknowledgments (ACKs).

3.2 Clock Synchronization

The next step is to synchronize Alice and Bob's clocks to accurately match the RSSI values recorded at different times with unknown delays introduced by hardware or software limitations. The process is illustrated in Fig. 3.

Since packet i and packet $i+1$ are sent in a fixed period T , $t_{i+1} - t_i = T$. With Eq. (1), we have $t'_{i+1} - t'_i$ equals to $T + (d_{tx}^{i+1} - d_{tx}^i)$. If the measured difference between t'_{i+1} and t'_i is actually close to T (e.g., t'_1 and t'_2 in the example), we assume $d_{tx}^{i+1} \approx d_{tx}^i \approx 0$; Otherwise, two values cannot match. Next, the receiver (Bob) picks up any pair of t'_p and t'_q such that $t'_p - t'_q \approx (p - q) * T$. Then, Δ is calculated by solving Eq. (2), where k is the number of selected timestamps (e.g., t'_1 , t'_2 , t'_5 and t'_7 in the example) and $c = C_{tx} + C_{rx} + \delta$, where can be measured offline [6]. Note that only one party (saying Bob) needs to calculate and adjust the clock difference.

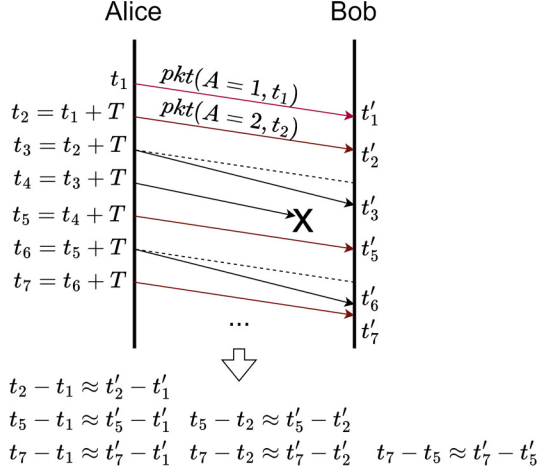


Fig. 3. Clock synchronization calculated on Bob's side.

$$\Delta = \frac{1}{k} \sum_{i=1}^k (t_i - t'_i - c) \quad (2)$$

Algorithm 1. Clock Adjustment and RSSI Matching

CLOCKADJUST(c) :

$\mathbf{S} \leftarrow \{\}$

Find any pair of pkts i and j s.t. $t'_i - t'_j \approx t_i - t_j$

if i **or** j **not** added in $\mathbf{S}$ **then**

$\mathbf{S} \leftarrow \mathbf{S} \cup \{t'_i - t_i\}$ or $\{t'_j - t_j\}$

end if

$\Delta \leftarrow \sum \mathbf{S} / |\mathbf{S}| - c$

RSSIMATCH(Δ) :

for all i and j s.t. $|t'_i - t'_j - \Delta| < 2\delta$ **do**

Add $(t'_i, RSSI_i)$ and $(t'_j, RSSI_j)$ for either Alice or Bob

end for

3.3 RSSI Matching and Smoothing

After adjusting the clock difference, both parties exchange the received timestamps (not RSSI values) and select RSSI values that have a close timestamp in both data sets. As shown in Fig. 4, we have $|t'_{B3} - t'_{A2}| < 2\delta$ and $|t'_{B4} - t'_{A5}| < 2\delta$ and only keep corresponding RSSI values after selection. The rationale behind this design is that the signals sent from the sender to the receiver and vice

versa often suffer the same attention or interference in a short period. Thus, both parties would observe highly matched values. The algorithm is illustrated in Algorithm 1.

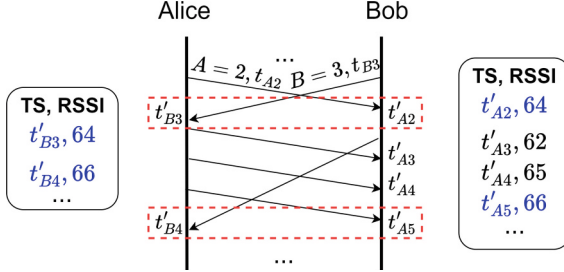


Fig. 4. Matching and Smoothing RSS samples recorded on both sides.

However, in the complex environments where UAVs operate, the noise can lead to mismatched RSSI values between the two communicating parties. A weighted sliding window smoothing method is adopted to mitigate the noise and ensure that the RSSI values are as similar as possible. Essentially, the data smoothing problem is about finding the optimal solution to reduce the noise. Traditional methods for data smoothing in key extraction primarily include the Equal Weight Method and Farrow Filter-based approach. However, they do not perform well in high-noise environments typical for UAV scenarios based on our field tests.

To address the limitations of traditional methods, we propose a Canonical Correlation Analysis (CCA)-based sliding window smoothing method. This method optimizes the traditional approach for high-noise data, achieving better entropy values. Initially, the first portion of RSSI values is exchanged between Alice and Bob to train the smoothing weights. After the weights are determined, RSSI values are no longer exchanged and used to extract keys only. Given the window size k , a weight sequence $a = (a_1, a_2, \dots, a_k)$ is set for Alice, and similarly for Bob, $b = (b_1, b_2, \dots, b_k)$. These weights must satisfy $\sum_{i=1}^k a_i = 1$ and $\sum_{i=1}^k b_i = 1$. The smoothed data for the i -th position is calculated as:

$$\begin{aligned} x'_i &= \sum_{j=1}^k a_j \text{RSSI}_{i+j-1}^A \\ y'_i &= \sum_{j=1}^k b_j \text{RSSI}_{i+j-1}^B \end{aligned} \quad (3)$$

Thus, the smoothed data sets are $X' = (x'_1, x'_2, \dots, x'_{n-k+1})$ and $Y' = (y'_1, y'_2, \dots, y'_{n-k+1})$. The unknown weight sequences a and b are resolved using the CCA framework, which aims to maximize the correlation $\rho(X', Y')$ by solving a $k \times k$ matrix equation. A small window size k is crucial to ensure that the distance-related information does not leak, which would otherwise compromise security. Through extensive experiments, it was found that a window size $k = 3$ is optimal for balancing noise reduction and security.

3.4 Improved LCA Key Extraction

UAV environments often have significant movement and rapid signal changes, so the RSSI data often contains slow variations that can obscure the true signal changes needed for key extraction. The smoothed data has filtered out slow-varying components related to the movement and distance between the UAV and the ground station. To obtain the residuals, which represent the rapid changes in RSSI values, we subtract the average of smoothed RSSI values from the original RSSI measurements. The residuals are then used as the primary data for the LCA process, ensuring that the key extraction focuses on the quick, more informative variations in the RSSI.

Next, we adopt a dual-threshold quantizer that converts the residual RSSI values into binary key bits. The quantizer operates by defining two thresholds based on the statistical properties of the RSSI data: the upper threshold $q+$ and the lower threshold $q-$. These thresholds are typically set using the mean μ_x and standard deviation σ_x of the RSSI values, with an additional scaling parameter α , saying $q+ = \mu_x + \alpha\sigma_x$ and $q- = \mu_x - \alpha\sigma_x$. Values falling between $q+$ and $q-$ are marked as ‘e’ (erasure) and are not used in the final key bit string.

For example, in Fig. 5, the result after quantizing the residuals is ‘00e1111ee1111111e0000000e110’. This method ensures that only the most significant changes in RSSI are considered, improving the consistency and reliability of the extracted key.

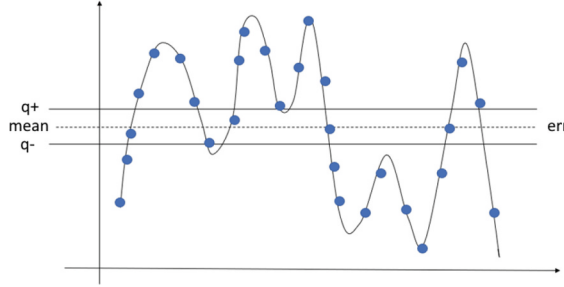


Fig. 5. Illustration of dual-threshold quantizer.

With a preliminary bit string that contains ‘0’s, ‘1’s, and ‘e’s, the next step is to process this bit string to extract the final key. The steps involved are:

- Erasure Removal: Remove all ‘e’ values from the bit string to ensure that only valid ‘0’s and ‘1’s remain.
- Bit Alignment: Align the bit strings from the UAV and the ground station to correct any mismatches due to slight timing differences.
- Information Reconciliation: Use an error correction protocol, such as the Cascade algorithm [14], to reconcile any remaining discrepancies between the bit strings of the two parties. This step often involves exchanging parity bits to ensure both parties end up with identical bit strings.

Privacy Amplification: To further enhance the security of the extracted key, a randomness extractor is applied. The implementation of the randomness extractor involves modeling dependencies using a Markov chain and employing the Bayesian Information Criterion (BIC) to determine the order of the Markov chain. This process ensures that the final key bits are as unpredictable as possible.

For each possible order of the Markov chain, we compute the likelihood of the observed sequence of key bits. This involves estimating the transition probabilities between states in the Markov chain. Then, we use the likelihood to calculate the BIC value for each model order. The model order that minimizes the BIC value is selected as the optimal order. By incorporating the BIC method, we can effectively model the dependencies in the extracted key bits and apply a randomness extractor to produce a highly secure key. The final key ensures a high level of security suitable for UAV-ground station communication.

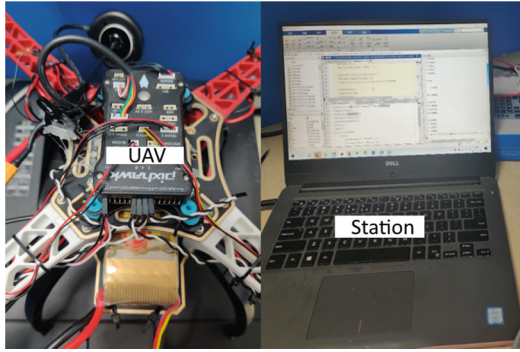


Fig. 6. Experiment devices including a UAV and a ground station.

4 Implementation and Evaluation

The implementation of our proposed key extraction method involves setting up an experimental environment (shown in Fig. 6). We used a UAV equipped with a Raspberry Pi 3B and a ground station with a Dell Intel i5 laptop. The Raspberry Pi was wired and attached to the bottom of the UAV.

To evaluate the performance of our key extraction method, we conducted a series of experiments in different environments: indoor lab, corridor, and outdoor UAV-ground station scenarios. To evaluate the performance of our key extraction model, several metrics, including Approximate Entropy Bit Rate (*abps*) and Secret Bit Rate (*sbps*), are defined to ensure that the model meets the necessary requirements.

Parameter $abps = \frac{E \cdot n_q - n_p}{t}$ serves as a comprehensive measure of the key generation process, incorporating the information leakage during reconciliation,

where E is the estimated Shannon entropy per encoded bit, n_q is the total number of encoded bits, n_p is the number of parity bits exchanged during information reconciliation, and t is the duration.

Parameter $sbps = \frac{n_e - n_p \cdot n_e}{n_q \cdot t}$ assesses the final rate of secure key bits after the information reconciliation and privacy amplification steps, where n_e is the total number of bits provided by the randomness extractor, n_p is the number of parity bits exchanged during information reconciliation, and n_q is the total number of encoded bits.

4.1 Indoor Experiments in Controlled Labs

As shown in Table 1, α determines the upper and lower thresholds of the dual-threshold quantizer, directly impacting the key extraction rate. The grid-search method was used to vary α from 0 to 0.5 in steps of 0.04. The results indicated that as α increases, the approximate entropy bit rate (abps) decreases. However, a higher α improves key consistency. For our experiments, α was set to 0.04 as a satisfactory value.

Table 1. Impact of Quantization Parameter α in Indoor Scenarios

α	Lab1	Lab2	Corridor1	Corridor2	Moving1	Moving2
0.00	1.19	1.01	1.74	0.99	1.19	1.01
0.04	1.45	1.00	1.18	1.08	1.45	1.00
0.08	1.34	0.86	1.45	0.94	1.34	0.65
0.12	1.26	0.69	1.29	0.86	1.26	0.69
0.16	1.16	0.67	1.12	0.64	1.16	0.67
0.20	1.05	0.57	1.10	0.57	1.05	0.57
0.24	0.83	0.56	0.81	0.35	0.83	0.56
0.28	0.78	0.41	0.69	0.00	0.78	0.41
0.32	0.71	0.41	0.64	0.00	0.71	0.41
0.36	0.68	0.27	0.43	0.00	0.68	0.27
0.40	0.58	0.25	0.38	0.00	0.58	0.25
0.44	0.50	0.23	0.34	0.00	0.50	0.23
0.48	0.46	0.16	0.28	0.00	0.28	0.12

As shown in Table 2, m is crucial in compressing the continuous bit strings into single bits during key extraction. The results showed that increasing m beyond 3 significantly reduces the approximate entropy bit rate (abps). Therefore, m was set to 2 for the subsequent experiments.

As shown in Table 3, the size of the sliding window s directly impacts the key extraction rate by affecting how the data is smoothed. The grid-search method indicated that as s increases, the *abps* gradually decreases. For the experiments, s was set to 20.

Table 2. Impact of Continuous Fragment Parameter m

m	Lab1	Lab2	Corridor1	Corridor2	Moving1	Moving2
2	1.33	0.82	1.03	1.03	1.33	0.82
3	0.94	0.51	0.89	0.42	0.94	0.51
4	0.64	0.35	0.41	0.00	0.64	0.35

Table 3. Impact of Sliding Window Size Parameter s

s	Lab1	Lab2	Corridor1	Corridor2	Moving1	Moving2
20	1.28	1.10	1.46	1.13	1.28	1.10
40	1.33	0.82	1.03	1.03	1.33	0.82
60	1.45	0.94	1.42	0.79	1.45	0.94
80	1.33	0.88	1.27	0.85	1.33	0.88
100	1.28	0.81	1.38	0.84	1.28	0.81

4.2 Outdoor Experiments in Field Tests

For the outdoor experiments, the UAV was flown in various patterns to simulate different operational scenarios. The parameters were optimized using the grid search method.

Due to the page limit, we only show the outdoor experiment results. The optimized parameters ($\alpha = 0.04$, $m = 2$, $s = 20$) provided the best performance in terms of key consistency and security. The *sbps* is around 4 7 bits per second, while the existing works such as LCA and ABSG did not work.

5 Related Work

Signal-Strength-Based Key Extraction. RSSI has become a popular source for key extraction due to its availability on most commercial devices. Key extraction methods based on RSSI take advantage of the received signal strength variations due to channel fading and randomness [1–3, 9, 10, 13]. Mathur’s level-crossing algorithm (LCA) [9] and Jana’s dynamic adaptive method (ABSG) [7] highlight the potential for high-consistency key generation despite environmental noise and equipment limitations. These methods primarily focus on static or low-mobility environments, such as indoor settings or low-speed scenarios, making them less effective for the dynamic and high-mobility context of UAV communications.

Other Properties in Key Extraction. Related studies also focused on leveraging other intrinsic properties of wireless channels to improve the efficiency and security of key extraction. Liu [8] addressed the issue of establishing key links when two devices are not within the same wireless local area network by exploiting the channel response from OFDM subcarriers. Patwari’s use of the

Karhunen-Loeve transform [11] and the ARUBE algorithm [5] have all contributed to higher key generation rates and better handling of noise and discrepancies in wireless channels. These methods ensure that even under dynamic and noisy conditions, secure keys can be reliably extracted.

Application in Vehicular Networks. Applying key extraction techniques to vehicular networks presents unique challenges due to the high mobility and dynamic environments. Although the work [14] proposed a scheme to allow two vehicles to extract a secret key from RSSI values in noisy, outdoor vehicular environments. However, this approach only works in the condition that the car is driven one after the other in the same direction on the same route at the same time. It is different from the scenario considered in this paper, where there is high mobility between UAVs and the ground station.

6 Conclusion

This paper presents a novel approach for key extraction in UAV-ground station communication by leveraging the Received Signal Strength Indicator (RSSI) of wireless channels. The proposed method addresses the inherent challenges of dynamic and noisy environments typical of UAV operations, providing a robust and efficient solution for secure communication.

References

1. Ali, S.T., Sivaraman, V., Ostry, D.: Eliminating reconciliation cost in secret key generation for body-worn health monitoring devices. *IEEE Trans. Mob. Comput.* **13**(12), 2763–2776 (2014). <https://doi.org/10.1109/TMC.2013.71>
2. Aono, T., Higuchi, K., Ohira, T., Komiyama, B., Sasaoka, H.: Wireless secret key generation exploiting reactance-domain scalar response of multipath fading channels. *IEEE Trans. Antennas Propag.* **53**(11), 3776–3784 (2005). <https://doi.org/10.1109/TAP.2005.858853>
3. Barsocchi, P., Oligeri, G., Soriente, C.: Shake: single hash key establishment for resource constrained devices. *Ad Hoc Netw.* **11**(1), 288–297 (2013)
4. Bennett, C.H., Brassard, G., Robert, J.M.: Privacy amplification by public discussion. *SIAM J. Comput.* **17**(2), 210–229 (1988). <https://doi.org/10.1137/0217014>
5. Croft, J., Patwari, N., Kaser, S.K.: Robust uncorrelated bit extraction methodologies for wireless sensors. In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IPSN '10, New York, NY, USA, pp. 70–81. Association for Computing Machinery (2010)
6. Han, H., Liu, Y., Shen, G., Zhang, Y., Li, Q., Tan, C.C.: Design, realization, and evaluation of Dozyap for power-efficient Wi-Fi tethering. *IEEE/ACM Trans. Networking* **22**(5), 1672–1685 (2014). <https://doi.org/10.1109/TNET.2013.2283636>
7. Jana, S., Premnath, S.N., Clark, M., Kaser, S.K., Patwari, N., Krishnamurthy, S.V.: On the effectiveness of secret key extraction from wireless signal strength in real environments. In: *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*. MobiCom '09, New York, NY, USA, pp. 321–332. Association for Computing Machinery (2009)

8. Liu, H., Wang, Y., Yang, J., Chen, Y.: Fast and practical secret key extraction by exploiting channel response. In: 2013 Proceedings IEEE INFOCOM, pp. 3048–3056 (2013). <https://doi.org/10.1109/INFOCOM.2013.6567117>
9. Mathur, S., Trappe, W., Mandayam, N., Ye, C., Reznik, A.: Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking. MobiCom '08, New York, NY, USA, pp. 128–139. Association for Computing Machinery (2008)
10. Paolo Barsocchi, Stefano Chessa, I.M.G.O.: A cyber-physical approach to secret key generation in smart environments. *J. Ambient Intell. Humanized Comput.* **4**, 1–16 (2011)
11. Patwari, N., Croft, J., Jana, S., Kasera, S.K.: High-rate uncorrelated bit extraction for shared secret key generation from channel measurements. *IEEE Trans. Mob. Comput.* **9**(1), 17–30 (2010). <https://doi.org/10.1109/TMC.2009.88>
12. Shannon, C.E.: Communication theory of secrecy systems. *The Bell Syst. Tech. J.* **28**(4), 656–715 (1949). <https://doi.org/10.1002/j.1538-7305.1949.tb00928.x>
13. Wei, Y., Zeng, K., Mohapatra, P.: Adaptive wireless channel probing for shared key generation based on PID controller. *IEEE Trans. Mob. Comput.* **12**(9), 1842–1852 (2013). <https://doi.org/10.1109/TMC.2012.144>
14. Zhu, X., Xu, F., Novak, E., Tan, C.C., Li, Q., Chen, G.: Using wireless link dynamics to extract a secret key in vehicular scenarios. *IEEE Trans. Mob. Comput.* **16**(7), 2065–2078 (2017). <https://doi.org/10.1109/TMC.2016.2557784>



Campus Card Security Depository System Based on State Secret Algorithm

Yingzi Huo^(✉), Kefei Li, Meiguo Ke, Yanbing Wu, Yuxin Liu,
and Ronglin Zhang

School of Computer Science and Engineering, Hunan University of Science and
Technology, Xiangtan 411100, China
yingzihuo02@mail.hnust.edu.cn

Abstract. With the continuous advancement of informationization, campus card management system has become an important part of the school's core infrastructure, which plays a vital role in campus life and management at various levels. However, the traditional campus card system has certain deficiencies in security, centralized management and trustworthiness. Aiming at these problems and challenges, this paper combines the state secret algorithm and blockchain technology to construct a campus card blockchain deposit system based on the state secret algorithm. The system is based on the federation chain architecture, adopts the state secret algorithm, proposes a double guarantee mechanism for data storage, and implements smart contracts and network optimization strategies, thus achieving significant improvement in the compliance regulation, security, transaction speed and efficiency of transaction data. After stress test verification, the system performs well in coping with the high-stress environment of blockchain, and is able to efficiently process a large number of transaction requests without obvious performance bottlenecks or serious response delays, which fully demonstrates its reliability and stability in practical applications.

Keywords: Blockchain · State Secret Algorithms · Data Security · Decentralization · Campus Card Management

1 Introduction

In the traditional economy, data transactions are fraught with challenges, including the incompleteness of data, which may result from inadequate regulation, inaccurate reporting, or the limitations of data collection methods [6]. Transaction data may be biased due to human error, data entry mistakes, or imprecise data collection techniques. The lack of standardization and traceability in transaction data, due to varying data collection standards across industries and regions, hinders comparative analysis and understanding [16]. Critically, transaction data often involves sensitive personal or institutional information. For instance, in the broader trend of economic development, campus cards have

become a carrier of students' financial information and a proof of identity in many schools. The loss of a campus card poses a significant risk of privacy leakage [20].

Incidents of information and financial loss due to lost campus cards are numerous [21]. In October 2020, a student from Panzhihua University had their campus card stolen and was fraudulently charged 70 yuan within ten minutes. In the winter of 2021, another student lost their campus card, resulting in the theft of 500 yuan of accumulated living expenses [11]. In 2022, a student's campus card was stolen, and the perpetrator used the student's identity to borrow over ten books from the library for resale. Each of these cases is related to the instability of campus card information, highlighting the urgent need for privacy protection measures to enhance the confidentiality and integrity of campus card data, safeguard students' informational property, and promote robust economic development [10].

Following the publication of "Bitcoin: A Peer-to-Peer Electronic Cash System" by Satoshi Nakamoto in 2008 [18], blockchain technology has surged in research interest. Blockchain can be categorized into public, consortium, and private chains based on node distribution. The advent of blockchain effectively addresses transaction data exchange issues, with consortium chains particularly suitable for electronic evidence storage, thereby further safeguarding against campus card risks [26]. As digital technology rapidly evolves, blockchain, a decentralized and secure distributed ledger technology, is gaining prominence [19]. Its applications in finance, healthcare, logistics, energy utilization, privacy management, cross-border payments, copyright protection, and government services have yielded significant results [3]. In the education sector, blockchain technology holds immense potential, especially in campus management, student identity verification, and academic record management, promising to revolutionize the educational system by enhancing the transparency, security, and trustworthiness of educational information [13].

In order to cope with the above challenges, this research develops a campus card blockchain depository system based on the state secret algorithm, which aims to enhance the security and reliability of campus card transactions. The main contributions of this paper are as follows:

- The system, based on a consortium chain architecture with defined node permissions and consensus mechanisms, enhances security, transaction speed, and cost efficiency, while employing domestic cryptographic algorithms SM2, SM3, and SM4 for robust data protection.
- The system implements a dual-data storage mechanism, with user basic information stored in a traditional MySQL database and transaction records on the blockchain, striking a balance between data security and system performance.
- Proposed smart contract and network optimization strategies refine the execution process and data storage, reducing gas consumption and improving transaction processing speed. The system's throughput and response capa-

bilities are notably enhanced through optimized network configurations and adjusted transaction parameters.

- The constructed system has been rigorously tested under blockchain stress conditions, demonstrating robust performance in handling a high volume of transaction requests without encountering significant performance bottlenecks or severe response delays.

The rest of the paper is structured as follows: the second section analyzes the current state of research related to campus card systems. The third part describes the specific architecture of the system in this paper. The fourth part tests the performance of the system. Finally, Part V summarizes the work of this paper.

2 Related Work

Currently, some schools have attempted to improve their campus card systems using blockchain technology to enhance information security and credibility [5]. However, most blockchain systems still use traditional encryption algorithms [22], which pose a risk of being attacked or cracked, especially with the enhancement of computing power and the development of cryptographic technology, some encryption algorithms may become less secure [17]. It mainly includes the following five questions:

1. Public blockchains suffer from long transaction delays and high costs due to a large number of nodes and decentralized consensus mechanisms [23]. The generation time of each block and the block size limit the system's throughput, causing new transactions to wait for inclusion in the next block, leading to delays [25]. This is unsuitable for immediate payment scenarios like retail transactions [9]. Traditional blockchain systems often use Proof of Work (PoW) algorithms [7], which are energy-intensive and not cost-effective for small or frequent transactions.
2. The public ledger of most traditional blockchain systems poses a risk of privacy leakage, as all transactions are transparent and traceable [30]. The vulnerability of nodes in a blockchain network, typically personal computers with lower performance and defense capabilities compared to dedicated servers [4], raises concerns about privacy and security in certain applications [27].
3. Conventional encryption algorithms may not meet the performance demands of high-frequency transactions, especially in blockchain environments requiring efficient processing of large volumes of transactions [1]. The transaction processing capacity of traditional blockchains like Bitcoin and Ethereum is limited, which restricts their application in high-frequency financial systems [12].
4. Traditional encryption algorithms are susceptible to various attacks, such as symmetric attacks and brute-force attacks, compromising their security and reliability, especially when handling sensitive information [15].

5. Storing all transaction information on the blockchain has its drawbacks, including increased storage costs, reduced performance, privacy issues, and data immutability [2]. The need for each node to store the entire blockchain's data can lead to scalability issues, while the verification and storage of all transactions can slow down the system [28].

To address these limitations, research has focused on introducing efficient consensus algorithms and data processing mechanisms to enhance transaction processing speed and throughput [6]. Optimizing blockchain architecture to reduce transaction confirmation times is crucial for handling high-frequency transactions. Efforts to efficiently integrate blockchain with advanced encryption algorithms aim to ensure security without compromising performance.

3 System Architecture

3.1 System Platform

FISCO BCOS is an open-source enterprise level blockchain platform developed and maintained by the Financial Blockchain Shenzhen Consortium [8]. According to system testing documentation, the platform's blockchain single chain performance can exceed 100000 TPS (using DMC algorithm), and the transaction confirmation time is seconds long, allowing the platform to maintain high efficiency when processing large amounts of transactions.

The national encryption algorithms SM2, SM3, and SM4 in the FISCO BCOS platform are developed based on domestic cryptographic standards, providing strong support for data encryption, data integrity verification, and message authentication. And the platform has designed universal modules and standardized protocols, which are designed to support different types of nodes and application scenarios, thereby reducing the integration cost and complexity between heterogeneous systems.

Smart contracts was proposed by computer scientist and encryption master Nick Szabo around 1994. Due to the fact that data cannot be changed once it is uploaded, the data generated by contract execution is transparent and traceable. Blockchain and smart contracts can perfectly complement each other's strengths and weaknesses. At the same time, smart contracts can automatically execute some pre-defined terms and rules based on these immutable trusted data on the chain. The transactions specified in the contract need to be executed frequently, and the contract will be executed without fail, making it difficult for malicious nodes to interfere. Smart contracts in blockchain are embedded as a scripting language into the blockchain network and provide rich functionality. Once a smart contract is deployed on the blockchain, it usually cannot be changed unless a specific upgradable design pattern is adopted. Once the triggering conditions are met, the smart contract will automatically execute and generate corresponding outputs according to the contract rules based on input instructions. The generated records will be put into transactions and ultimately stored in the blockchain.

Smart contracts in Ethereum run in a closed sandbox environment, also known as EVM (Ethereum Virtual Machine) virtual machine. Based on the trustworthy and decentralized characteristics of blockchain, smart contracts can execute efficiently and without interference. However, when running smart contracts in EVM, they are isolated from the surrounding environment and cannot access the network, file system, or other processes on the Ethereum platform. FISCO BCOS currently uses Ethereum's smart contract language Solidity, thus also adopting its abstraction of the executor interface.

3.2 Overall System Analysis

This system adopts front-end and back-end separation development, the back-end server receives the request from the front-end to complete the corresponding logic function, and then update the data. The data persistence is divided into two parts, one is stored in the traditional relational database, and the other part involving the amount of consumption is stored in the blockchain, which not only ensures the security of the consumption records, but also prevents the rapid growth of block data. This time, the back-end server is developed by the currently popular Spring Boot, the database is MySQL5.7, the blockchain storage is FISCO BCOS 2.0, and the encryption process is realized by using the state secret algorithm, and the front-end interface is developed by using Vue and element UI to complete the separation of front and back ends. In order to better facilitate the transplantation and adaptation of different front-end and back-end frameworks, this development strictly follows the standard development process, and at the same time encapsulates the corresponding functions, so as to increase the robustness of the program and secondary development.

FISCO BCOS is a domestic enterprise-led R&D, open-source, secure and controllable enterprise-level financial alliance chain bottom layer platform, built in collaboration with the Open Source Working Group of the Golden Chain Alliance, and officially open-sourced to the public in 2017 [14]. FISCO BCOS takes the actual needs of the alliance chain as the starting point, takes into account performance, security, operability, ease of use, and extensibility, and supports a variety of SDKs, which makes it convenient to coding use among back-end software developers [24].

The signature verification of the state-secret version of FISCO BCOS [29] is changed from the ECDSA signature algorithm of the standard version to the SM2 signature algorithm, the message digest algorithm is changed from the SHA-256 SHA-3 to the SM3 message digest algorithm, the drop disk encryption algorithm is changed from the AES-256 encryption algorithm to the SM4 encryption algorithm, and the certificate mode is changed from the OpenSSL certificate mode to the state-secret dual-certificate mode, and so on. The certificate mode has been changed from OpenSSL certificate mode to State Secret dual certificate mode, etc. At the same time, because the state secret version and the standard version of FISCO BCOS have more differences in data structure, for example, symmetric encryption and decryption is changed from AES (length of encryption secret key: 256 bits) to SM4 (length of symmetric key: 128

bits), and the length of the transaction is changed from 520bits (of which the identifier is 8bits, and the length of the signature is 512bits) to 1024bits (128 bytes, where the public key is 512bits and the signature length is 512bits), so it is necessary to use the script provided by FISCO BCOS to generate a new certificate corresponding to the state secret version. At the same time, the previous standard version of the smart contract should be recompiled into the state-secret version of the smart contract by replacing the ethereum-solidity compiler with the state-secret-solidity compiler.

3.3 System Architecture Design

The architecture of this system is shown in Fig. 1. In the application layer, user or administrator: the user interacts with the system through the client (e.g., campus card) to initiate information query and data operation requests; front-end control interface or API: the user's request is sent to the system through the front-end interface or API, which is used to process the user's query and data operation needs; back-end processing: it receives the request from the front-end and carries out the corresponding data processing. The processing includes data storage, querying and uploading to the blockchain; Database API: the back-end processing module interacts with the database through the database API to realize the data storage and query functions. In the interface layer, API function interfaces are provided to realize the fast invocation of blockchain functions. The interface layer is the bridge between the application layer and the protocol layer, and the blockchain functions are invoked through the standardized API. In the protocol layer, the blockchain network: uses the P2P protocol to realize communication between nodes, ensuring efficient message delivery and data consistency.

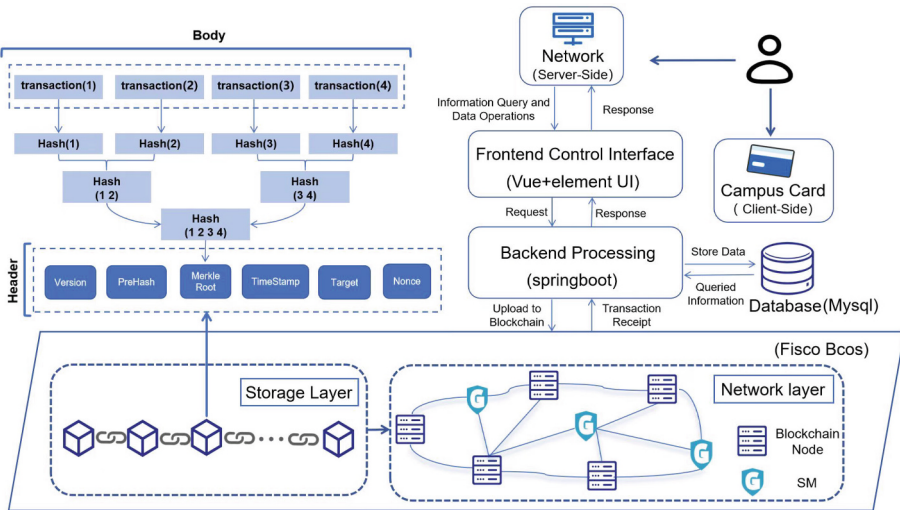


Fig. 1. System architecture.

The blockchain network is responsible for the uploading and distributed storage of data; State Secret Algorithm: uses the State Secret Algorithm to ensure the security and privacy of data, and ensures encrypted processing of data during transmission and storage.

This system is used to manage the campus card and record the use of the user's campus card. There are two types of users: administrator and ordinary user. Each user has a number (administrator is a job number, ordinary user is a student number or job number), name, gender, ID number, class information, and college. Each user only has one campus card, which has the card number and user ID, and contains consumption records. Each consumption record includes the order number, card number, time, change amount (positive amount indicates recharge, negative amount indicates consumption), and post consumption amount. Administrators can add regular users, create new campus cards, and modify information. Regular users can only view their own information and corresponding consumption records. Each user needs to log in with their own ID and password before use, and also needs to save data such as phone number and email (which can be empty).

To ensure the security of the system, the administrator user should be the only role authorized to modify and delete user information. Therefore, the system should implement strict permission control to ensure that only administrators can access user management functions. Administrator users modify users, and the user module interface displays all created users. The created users display their student/employee ID, name, and other information, while retaining the ability to delete and modify. Clicking 'modify' will automatically pop up a pop-up window prompting the administrator user to modify user information (modifiable information includes password, identity, phone number, email; card number and name cannot be changed to ensure the user's identity and uniqueness are not affected). Clicking 'delete' will delete the selected user, but the consumption records of the card number will not be cleared due to the immutability of blockchain.

4 System Testing and Analysis

In this chapter, we conducted a comprehensive performance test for the campus card security deposit system based on the state secret algorithm, aiming to evaluate the security and stability of the system in processing transaction deposits. By performing stress tests using the Java SDK in the development environment and simulating node failures to test the system's recovery and fault-tolerance capabilities, we have achieved a comprehensive evaluation of the system's transaction throughput, network latency, security, and consistency.

Our stress testing method can be accessed at <https://github.com/FISCO-BCOS/java-sdk-demo>. When doing the parameter setting, it is required that count should be able to cover the expected maximum load scenario, so we have to determine the total number of transactions to be sent based on the actual transaction scenario and set the parameter count to 10000.

Considering that when qps (requests per second) is too large, it will allow transactions to pile up in the transaction pool, and these piled up transactions cannot be executed immediately, making the measured TPS smaller than the actual TPS. When setting qps, start with a relatively low value and increase it gradually. Observe the performance of the system under different loads to find the performance limits and bottlenecks of the system.

Considering that in the campus card blockchain depository system, each transaction usually involves the user's asset transfer, deposit and other operations, which need to ensure sequentiality and accuracy. Therefore, using serial pressure testing can better simulate the actual transaction scenarios and ensure that each transaction is executed in a specific order, thus ensuring data consistency and system stability. We will use serial pressure testing to initially confirm the qps value during testing.

qps refers to the number of requests (or queries) that the system can process per second, which is usually used to describe the system's ability to process requests within a certain period of time. tps refers to the number of transactions or transactions that the system can complete per second, which is used to measure the system's actual business processing capability and transaction processing speed.

The relationship between qps and tps is determined by the amount of transactions corresponding to each request (or query). For example, if a query operation usually generates one transaction, then tps can be approximately equal to qps. However, in some cases, a request may cause multiple transactions to be executed, and then the ratio of qps to tps is no longer 1:1. To summarize, qps can be understood as an indicator of the system's load capacity, while tps more directly reflects the system's business processing capability.

Therefore, in order to comprehensively evaluate the performance of the system under different loads and performance limits, we tested the count of 10000, qps range of [100,800], the relevant data are shown in the Table 1.

Table 1. Table of qps and tps values.

qps	tps
100	99.83
200	198.96
300	291.95
400	385.57
500	418.77
600	412.08
700	413.58

From the above table we can see that the system peaks at tps when qps is 500, so below we will set qps to a fixed value of 500 for testing.

The stress test is performed on the group group2 and the result is shown in the Fig. 2.

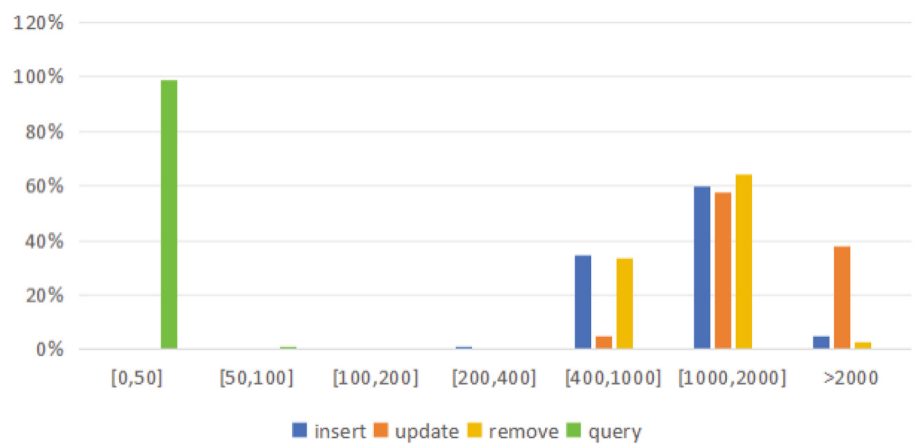


Fig. 2. System architecture.

Most of the insert, update, and remove transactions took between 400 and 2000 milliseconds, and only a few transactions took more than 2000 milliseconds. Comprehensively, the system has high processing capability and fast response speed. And after the test, the test report shows that the error rate of the system is 0.0%, which indicates that the system still maintains stability during the test period.

5 Conclusion

The campus card blockchain depository system based on the state secret algorithm effectively solves the shortcomings of the traditional centralized campus card system in terms of security, data integrity and management transparency by utilizing the decentralized and tamper-proof characteristics of the blockchain. The system ensures data integrity and traceability by recording every transaction and user authentication on the blockchain, while the application of state secret algorithms enhances the security and trustworthiness of transactions and authentication, significantly reducing the risk of data tampering or forgery. In addition, the application of smart contracts realizes automated authority management and security policy enforcement, further improving the security and operational efficiency of the system and providing a more secure and efficient management mode for the campus card system.

References

1. Alston, E.: Blockchain and the law-legality, law-like characteristics, and legal applications. *Econ. Blockchain Cryptocurrency*, 117–144 (2022)
2. Bernabe, J.B., Canovas, J.L., Hernandez-Ramos, J.L., Moreno, R.T., Skarmeta, A.: Privacy-preserving solutions for blockchain: review and challenges. *IEEE Access* **7**, 164908–164940 (2019)
3. Cai, J., Liang, W., Li, X., Li, K., Gui, Z., Khan, M.K.: GTxChain: a secure IoT smart blockchain architecture based on graph neural network. *IEEE Internet Things J.* (2023)
4. El Kafhali, S., El Mir, I., Hanini, M.: Security threats, defense mechanisms, challenges, and future directions in cloud computing. *Arch. Comput. Meth. Eng.* **29**(1), 223–246 (2022)
5. Guo, H., Yu, X.: A survey on blockchain technology and its security. *Blockchain Res. Appl.* **3**(2), 100067 (2022)
6. Jiang, L., Chen, P., Liao, J., Jiang, C., Liang, W., Xiong, N.N.: PCFS: an intelligent imbalanced classification scheme with noisy samples. *Inf. Sci.* 121020 (2024)
7. Lasla, N., Al-Sahan, L., Abdallah, M., Younis, M.: Green-pow: an energy-efficient blockchain proof-of-work consensus algorithm. *Comput. Netw.* **214**, 109118 (2022)
8. Li, H., et al.: FISCO-BCOS: an enterprise-grade permissioned blockchain system with high-performance. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–17 (2023)
9. Liang, W., et al.: Spatial-temporal aware inductive graph neural network for c-its data recovery. *IEEE Trans. Intell. Transp. Syst.* **24**(8), 8431–8442 (2022)
10. Liang, W., Li, Y., Xu, J., Qin, Z., Zhang, D., Li, K.C.: QoS prediction and adversarial attack protection for distributed services under DLAAS. *IEEE Trans. Comput.* (2023)
11. Liang, W., Liu, Y., Yang, C., Xie, S., Li, K., Susilo, W.: On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: a comprehensive survey. *ACM Comput. Surv.* **56**(12), 1–35 (2024)
12. Liao, M., Lu, Y., Li, X., Di, S., Liang, W., Chang, V.: An unsupervised image dehazing method using patch-line and fuzzy clustering-line priors. *IEEE Trans. Fuzzy Syst.* (2024)
13. Liu, Y., Liang, W., Xie, K., Xie, S., Li, K., Meng, W.: Lightpay: a lightweight and secure off-chain multi-path payment scheme based on adapter signatures. *IEEE Trans. Serv. Comput.* (2023)
14. Liu, Y., Li, Y., Lin, S.W., Yan, Q.: MODCON: a model-based testing platform for smart contracts. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1601–1605 (2020)
15. Mangala, N., Venugopal, K., et al.: Light weight circular error learning algorithm (CELA) for secure data communication protocol in IoT-cloud systems. *Int. J. Adv. Comput. Sci. Appl.* **14**(7) (2023)
16. Menon, S., Jain, K.: Blockchain technology for transparency in Agri-food supply chain: use cases, limitations, and future directions. *IEEE Trans. Eng. Manage.* **71**, 106–120 (2021)
17. Mousavi, S.K., Ghaffari, A., Besharat, S., Afshari, H.: Security of internet of things based on cryptographic algorithms: a survey. *Wireless Netw.* **27**(2), 1515–1555 (2021). <https://doi.org/10.1007/s11276-020-02535-5>

18. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Satoshi Nakamoto (2008)
19. Perdana, A., Robb, A., Balachandran, V., Rohde, F.: Distributed ledger technology: its evolutionary path and the road ahead. *Inf. Manag.* **58**(3), 103316 (2021)
20. Procopiou, A., Stavrou, E.: Privacy concerns in smart indoor environments in the internet of everything era: a smart university campus case study. In: *International Conference on Internet of Everything*, pp. 92–109. Springer (2023)
21. Rtayli, N.: An efficient deep learning classification model for predicting credit card fraud on skewed data. *J. Inf. Cybercrimes Res.* **5**(1), 61–75 (2022)
22. Sathya, A., Banik, B.G.: A comprehensive study of blockchain services: future of cryptography. *Int. J. Adv. Comput. Sci. Appl.* **11**(10) (2020)
23. Wang, W., et al.: A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access* **7**, 22328–22370 (2019)
24. Wang, Y., Wang, C., Luo, X., Zhang, K., Li, H.: A blockchain-based IoT data management system for secure and scalable data Sharing. In: Liu, J.K., Huang, X. (eds.) *NSS 2019. LNCS*, vol. 11928, pp. 167–184. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36938-5_10
25. Weber, I., et al.: On availability for blockchain-based systems. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 64–73. IEEE (2017)
26. Xiao, Y., Huo, Y., Cai, J., Gong, Y., Liang, W., et al.: ERF-XGB: an edge-IoT-based explainable model for predictive maintenance. *IEEE Trans. Consum. Electron.* (2024)
27. Xie, S., Xiao, L., Han, D., Xie, K., Li, X., Liang, W.: HCVC: a high-capacity off-chain virtual channel scheme based on bidirectional locking mechanism. *IEEE Trans. Network Sci. Eng.* (2023)
28. Zhang, J., Li, X., Gu, K., Liang, W., Li, K.: Secure aggregation in heterogeneous federated learning for digital ecosystems. *IEEE Trans. Consum. Electron.* (2023)
29. Zhang, L., Ge, Y.: Identity authentication based on domestic commercial cryptography with blockchain in the heterogeneous alliance network. In: *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pp. 191–195. IEEE (2021)
30. Zhao, Y., Yu, Y., Chen, R., Li, Y., Tian, A.: Toward transparent and accountable privacy-preserving data classification. *IEEE Network* **35**(4), 184–189 (2021)



Network Traffic Classification via Deep Hashing Network

Kunxiang Niu^(✉) and Wen Luo

Nanjing University of Science and Technology, Nanjing 210094, Jiangsu, China
kx_niu@njjust.edu.cn

Abstract. The swift growth of the internet has resulted in a more complex and varied network traffic landscape, presenting substantial challenges for accurate traffic classification. Deep learning methods have provided innovative solutions for network traffic classification, but they often require model retraining to meet evolving classification demands.

This paper introduces a new approach for classifying network traffic using deep hashing techniques, which maps network traffic data into binary hash codes for efficient classification. To the best of our knowledge, this is the first research to utilize deep hashing techniques for network traffic classification, bypassing the need for complex feature selection and utilizing raw traffic data as input. Experimental results using publicly available datasets indicate that our method attains high classification accuracy and performs exceptionally well in detecting malicious traffic.

Keywords: Network Traffic Classification · Deep Hashing · Deep Learning

1 Introduction

The swift growth of the internet has resulted in a growing diversity of network traffic types, posing substantial difficulties in precisely categorizing this ever-changing traffic. Network traffic classification involves categorizing and labeling the data transmitted over a network. This traffic can encompass various types, such as web browsing, file transfers, emails, and streaming. Accurate classification enables network administrators to better understand data flows, facilitating network management, optimization, and security measures. It is especially critical in associating traffic with originating applications, making it an essential task in managing and securing networks. In network security, traffic classification forms the basis for tasks like anomaly detection, helping to identify the malicious exploitation of network resources.

Over the past few years, deep learning has garnered considerable interest across diverse domains, and network traffic classification is no exception. Research on traffic classification using deep learning methods has achieved certain results, but there are some problems with this method: on the one hand, professionals are required to perform feature engineering when training the model; on the other hand, Internet applications are emerging in an endless stream and there are many types, and it is difficult to fully

summarize the traffic category with a single label. Meanwhile, using deep learning for classification tasks can usually only identify data types in the training set while unable to classify other categories, which obviously cannot adapt to the ever-expanding scale of the Internet.

Nearest neighbor search seeks to retrieve samples from a database that are most similar to a given query, serving as a fundamental task in fields such as data mining and computer vision. Hashing has become one of the most widely utilized techniques for this purpose, thanks to its efficiency in both computation and storage. Deep hashing, in particular, offers enhanced capabilities for addressing nearest neighbor search problems by representing input data more effectively through an end-to-end neural network architecture. Deep hashing model can adapt to datasets of different sizes and complexities through end-to-end training. When the amount of data increases, deep hashing can continue to classify effectively without the need to completely reconstruct the model.

Based on the advantages of deep hashing, this paper proposes a novel method for network traffic classification using deep hashing. Unlike traditional deep learning that directly outputs specific classifications, our approach transforms network traffic data into hash codes using a neural network, enabling nearest neighbor search within a hash library to classify traffic. To our knowledge, this is the first study to implement deep hashing techniques in the field of network traffic classification.

The contributions of this work are summarized as follows:

- We propose a deep hashing network for processing network traffic, which converts the input traffic data into binary hash codes.
- The classification of the traffic is performed by retrieving the hash codes of the traffic to be classified from a hash library.
- Publicly accessible datasets were utilized to conduct experiments, confirming the efficacy of the proposed method in successfully performing network traffic classification.

The remainder of this paper is organized as follows: Sect. 2 reviews related work; Sect. 3 outlines our proposed classification method, including the construction of the deep hashing model; Sect. 4 presents the conducted experiments; and Sect. 5 concludes the paper.

2 Related Work

2.1 Traffic Classification

Traffic classification indicates the activity of classifying network traffic according to specific criteria and has long been a critical task in network management and cybersecurity. By classifying traffic into distinct categories, network administrators can better understand data flows, facilitating network optimization, management, and security measures. Associating network traffic with its originating application is particularly crucial in network security, where it serves as a foundation for detecting and mitigating threats.

Traditional techniques for traffic classification can generally be divided into four major strategies [1]: port-based, deep packet inspection (DPI), statistical-based, and

behavior-based. Port-based methods classify traffic by associating services or applications with registered port numbers, such as HTTP, which typically uses port 80. While port-based classification was one of the earliest techniques, its effectiveness has significantly diminished in modern networks due to the widespread use of random ports and port obfuscation. For instance, Madhukar et al.'s experiments indicated that relying solely on port-based methods resulted in the inaccurate classification of nearly 70% of network traffic [2]. DPI examines the actual content of data packets to identify specific services or applications by matching packet content to predefined protocol signatures. While this method can be effective for unencrypted traffic, it faces significant limitations with encrypted data, raises privacy concerns, and imposes substantial computational overhead on network systems. Statistical-based and behavior-based methods, on the other hand, leverage the distinct statistical and temporal patterns of traffic generated by different applications or services [3]. These methods are more flexible, as they are capable of handling both encrypted and unencrypted traffic by examining characteristics like packet size, flow duration, and packet intervals.

In recent years, deep learning has gained prominence due to its superior performance across various fields, including categorizing network traffic [4]. Deep learning models automatically extract significant characteristics from network traffic data, enabling the accurate categorization of application-layer protocols, network services, and even malicious activities. One of the pioneering works by Moore et al. proposed a method for classifying network traffic utilizing the naive Bayes principle, which modeled the probabilistic relationships between network traffic features and traffic categories [5].

Other researchers have explored more innovative approaches, such as transforming network traffic data into image-like formats for classification using neural networks. This technique eliminates the need for manual feature engineering by directly using raw traffic as input, and it pioneered the use of feature representation techniques for classifying malicious software traffic [3].

Additionally, deep learning models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been used to capture the hierarchical structure of network traffic [6]. Since network traffic consists of sequential data, organized in bytes, packets, sessions, and flows (analogous to characters, words, sentences, and documents in natural language processing), one-dimensional CNNs have been particularly effective for traffic classification [7]. For example, one study utilized CNN and LSTM models in a novel intrusion detection system (HAST-IDS) to learn both low-level spatial and high-level temporal features from traffic data [8]. Similarly, RNNs have been applied to transform network flows into word-like sequences, where these sequences are modeled as language models, enabling the detection of anomalous or malicious activities in network traffic [9]. Recent studies have also applied graph neural networks (GNNs) to further mining network traffic characteristics. [10] summarizes the application of graph-based deep learning solutions to various communication network problems like traffic prediction and intrusion detection. [11] utilized GNNs to extract flow-level features from traffic interaction graphs. [12] uses GNNs to detect network attacks by arranging network traffic data sequentially and building a graph structure derived from log density information, enabling a deeper understanding of system behavior patterns.

2.2 Nearest Neighbor Search and Deep Hashing

Nearest neighbor search is a method used to find the most similar neighbors within high-dimensional space. For a query point, the algorithm identifies the data point in the data set closest to the query, enabling operations such as classification based on the search results. This technique is fundamental in fields such as data mining and image retrieval. In a d -dimensional Euclidean space R^d , the goal of nearest neighbor search is to find the element $NN(x)$ from the set $\Pi \subset R^d$ such that $NN(x) = \operatorname{argmin}_y \rho(x, y)$, where $x \in R^d$ is the query, and ρ is a distance metric like Euclidean distance or cosine similarity. Since exact nearest neighbor retrieval is computationally expensive, approximate nearest neighbor algorithms are often used to balance efficiency and accuracy.

Hashing algorithms have gained prominence for their ability to boost computational efficiency and reduce memory usage [13, 14]. The objective of hashing is to map the content features of the initial space into a Hamming space, generating compact binary hash codes. Given an input x , the objective of a hashing method is to design a hash function f that maps x to a binary code b for efficient retrieval. An ideal hash function preserves the relative distances of items in the original space, guaranteeing that items near the query in Hamming space remain close in the original space as well.

With the advent of deep learning, deep hashing methods have attracted growing interest. These methods offer two key advantages: deep learning's powerful representation capabilities allow it to learn complex hash functions, and its end-to-end architecture simplifies the process of generating hash codes. In deep hashing, deep neural networks serve as hash functions, mapping inputs into binary hash strings for retrieval. Due to its ability to capture intricate patterns, deep hashing has shown considerable success in image classification [15]. In computer vision, deep hashing networks are often adaptations of standard models like AlexNet [16], VGGNet [17], and ResNet [18], where the classification head is replaced with fully connected layers containing L units for hash code learning. A sign activation function is typically applied at the end of the network to produce the hash codes [15, 19].

3 Methodology

This paper introduces a technique for network traffic classification that leverages deep hashing. For a piece of network traffic data that needs to be classified, the deep hashing network takes this traffic data as input and outputs a binary hash code composed of a sequence of binary strings as the hash code result for that traffic. A nearest neighbor search is conducted on the obtained hash code within a database to identify the traffic that is most similar to it. The traffic classification is determined based on the results of the nearest neighbor search. The training steps and the testing steps are shown in Fig. 1 and Fig. 2.

3.1 Data Preprocessing

We preprocess raw network traffic data to prepare it for input into the neural network in this section. The first step involves segmenting the raw traffic into distinct units. The

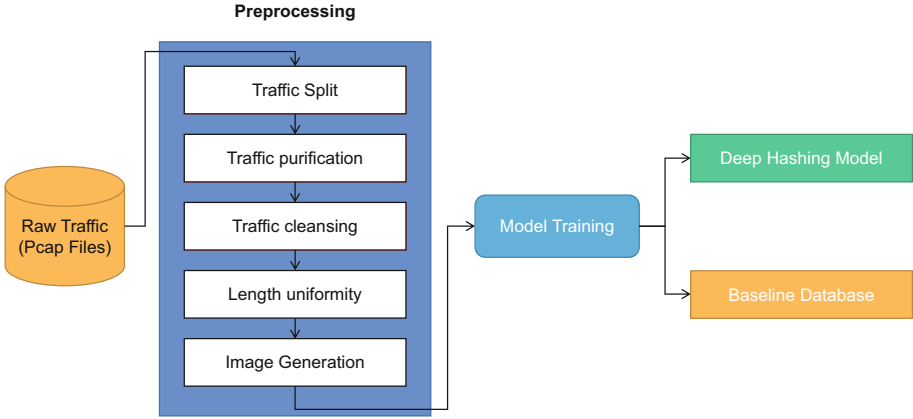


Fig. 1. Training steps of deep hashing network.

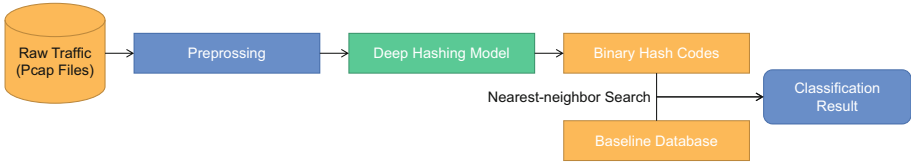


Fig. 2. Using deep hashing network for network traffic classification.

most prevalent representations of network traffic are sessions and flows. Sessions are defined by a five-tuple consisting of the source IP, destination IP, source port, destination port, and transport layer protocol. Flows, while similar to sessions, differ in that they represent traffic in a single direction, meaning that the destination and source IP/port pairs cannot be swapped. In our experimental framework for network data flow analysis, sessions are selected as the primary means of representing network communication patterns. The data set we are using is USTC-TFC 2016 data set [3], which consists of 10 kinds of real network traffic data and 10 kinds of malicious network data. Based on the following reasons, we choose to transform the original network data from pcap format to image format: (1) Deep learning methods have more advantages when processing images; (2) Many deep learning methods that convert network traffic data into images for classification have achieved good results [3]; (3) When processing images, deep hashing can better extract information from the carrier; (4) The visualization result of different classes are easy to distinguish from each other. Figure 3 illustrates the visual representation of all classes within the dataset. It can be easily concluded that different traffic classes exhibit a clear degree of distinction.

The whole data preprocessing procedure consists of the following steps:

Traffic Split. In this section, we split the original network traffic data to obtain several discrete network streams.

Traffic Purification. Removing irrelevant data from traffic packets, including TCP or UDP headers and certain Ethernet-related details at the data link layer, such as MAC

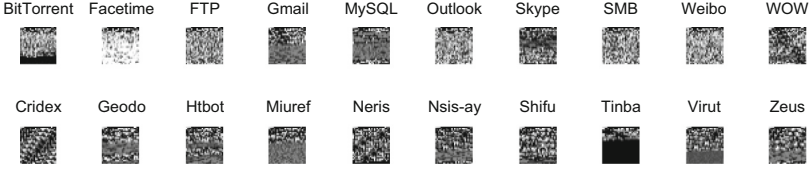


Fig. 3. Visualization of All Traffic Classes

addresses, as they do not contribute to subsequent processes and may even negatively impact the outcomes.

Traffic Cleansing. Removing duplicate and empty files, as these files may impair the model’s learning capabilities.

Length Uniformity: To ensure that traffic data retains consistent length and includes network traffic characteristics as comprehensively as possible for more accurate classification, we fix the data length at 784 bytes in this paper. If the file size exceeds 784 bytes, the excess data is discarded. Otherwise, we add bytes at the end.

Image Generation. The 784-dimensional traffic data in pcap format is converted into $28 * 28$ IDX format pictures. The IDX format is a commonly used file format in the field of machine learning.

To prevent data imbalance from affecting experimental results, we have selected approximately 6000 flows for each type of traffic data, ensuring a rough balance between the various traffic types to avoid any impact on the experimental results.

3.2 Deep Hashing Neural Network

Given a data set $X = \{x_1, x_2, \dots, x_n\}$ with $x_i \in R^d$, the primary objective of a hashing technique is to generate $\{y_i\}_{i=1}^n$ with each $y_i \in \{0,1\}^m$ as the binary code for each input. A good hashing code should project similar data pairs in the original space to similar binary codewords in Hamming space, which means that given two pieces of similar data belonging to the same category, their binary codes tend to have small distance in Hamming space, while two pieces of data from different category tend to have codes that have large distance in Hamming space. The primary objective of hashing technique learning is to identify a suitable transformation $H: X \rightarrow \{0,1\}^m$.

Our approach initially transforms the input data into a feature vector via a series of convolutional and pooling layers,

$$h_k(x) = \text{sigmoid}\left(\sum_{j=1}^m W_j(\text{CNN}(x))\right)$$

where m denotes the number of hash functions, $\text{CNN}(x)$ represents the feature extraction performed on input images derived from network traffic, W_j is the projection vector for the k -th hash function. The learning of each hash function $h_k(\cdot)$ is conducted independently through the application of a linear transformation on a shared feature representation layer. The sigmoid function is widely used in neural networks because of

its readily computable derivatives and physiological characteristics. Since the sigmoid function compresses the signal to $[0, 1]$ and experiments show that its output tends to cluster near zero or one—at the tails of the curve—we use the output of the sigmoid layer to represent our hash codes. Since CNN has demonstrated outstanding performance in various image tasks, we decided to utilize a straightforward CNN model to meet the demands of traffic classification.

3.3 Traffic Classification Using Deep Hashing

After successfully training the deep hashing network using the training data set, we can obtain a well-trained model ready for practical application. The next step involves using this network to process the validation data set, generating a set of hash codes for different types of traffic. These hash codes are stored in a database for future reference. When new network traffic data arrives, it is processed by the trained network, which outputs a sequence of hash codes representing the traffic's features. A nearest-neighbor search is then performed on this new sequence of hash codes within the pre-existing hash code database. This search identifies the hash codes most similar to the new ones. Based on these results, the traffic category is determined. The nearest-neighbor search effectively compares the new traffic data to the stored data, finding the most similar samples and thus classifying the traffic.

4 Experiment

4.1 Dataset

The USTC-TFC 2016 dataset comprises two distinct components. The initial component encompasses ten varieties of malicious traffic sourced from public websites, which were gathered by CTU researchers in real-world network environments spanning from 2011 to 2015. The subsequent component, on the other hand, consists of ten types of normal traffic, captured utilizing IXIA BPS, which includes eight common application types. To prevent data imbalance from affecting experimental results, we have selected approximately 6000 sessions for each type of traffic data, ensuring a rough balance between the various traffic types to avoid any impact on the experimental results. Following preprocessing, the dataset is split into a training set and a testing set in a 9:1 ratio. Once the model is trained, it processes the training set to generate a traffic hash code database, which is then used as the retrieval space for the testing set.

4.2 Model Configuration

We have implemented our proposed methods utilizing the open-source framework known as Caffe [22]. Specifically, in the convolutional layers (first, second, and third), we employ 32, 32, and 64 filters, respectively, each measuring 5×5 in size and followed by the ReLU activation function. Above the third pooling layer, we incorporate a hash mapping layer, which is subsequently followed by a sigmoid layer for compression.

4.3 Evaluation Metrics

We employ a series of metrics to assess and contrast the efficacy of hashing networks, namely Accuracy, Precision, Recall, F1 Score, defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1_score = \frac{2 * Precision * Recall}{Precision + Recall}$$

4.4 Experiment Results

We conducted experiments using the USTC-TFC2016 dataset to assess the model's traffic classification capabilities. The experiments were divided into four parts:

Experiment 1: This part involved processing all data and determining whether the input traffic is normal or malicious.

Experiment 2: In this experiment, only normal traffic was processed and classified based on the application.

Experiment 3: Focusing exclusively on malicious traffic, it was classified based on the application (ten-class classification).

Experiment 4: Processing all traffic data and classifying it based on the application.

Table 1 displays the experimental results and specific data for both ten-class and twenty-class classifications. Even in the twenty-class task where normal and malicious traffic are mixed, our model did not mistakenly classify normal traffic as malicious. This is because, in the hash retrieval space, there is a significant separation between normal and malicious traffic. This also results in the twenty-class task having a retrieval space around the test data similar to the ten-class task, making the specific results (Precision, Recall, F1) for both tasks relatively consistent.

Table 1. Accuracy of different classifiers.

Metric	ACC	PR	RC	F1
Binary	1	1	1	1
10-class Benign	0.932	0.937	0.930	0.930
10-class Malware	0.914	0.902	0.903	0.897
20-class	0.923	0.919	0.916	0.913

Overall, our method achieves high classification accuracy for various classification scenarios. In categories where the model performs poorly, such as the low accuracy for BitTorrent in normal traffic classification, this may be related to the nature of P2P services. For the Gmail category, most misclassifications occur when it is classified as Outlook. This is likely due to both Gmail and Outlook belonging to the Email/WebMail category, sharing certain similarities. This is likely because both Gmail and Outlook belong to the Email/WebMail category, sharing some similarities. As for the lower performance of the Neris and Verit categories in malicious traffic, it may be due to the unique features of these categories, which require further research and investigation. Table 2 shows the specific result of multi-classifiers, and Fig. 4 shows the confusion matrix.

Table 2. Precision, recall, and f1 value of multi-classifiers.

Class	PR	RC	F1	Class	PR	RC	F1
BitTorrent	0.955	0.849	0.901	Cridex	0.857	1	0.920
Facetime	0.991	1	0.995	Geodo	0.941	0.929	0.933
FTP	0.989	0.920	0.953	Htbot	0.959	0.941	0.950
Gmail	0.933	0.785	0.852	Miuref	0.931	1	0.955
MySQL	0.989	0.989	0.989	Neris	0.863	0.760	0.801
Outlook	0.734	0.942	0.828	Nsis-ay	0.934	0.920	0.930
Skype	1	1	1	Shifu	0.921	0.951	0.933
SMB	0.852	0.920	0.885	Tinba	0.866	0.955	0.901
Weibo	0.919	0.890	0.902	Virut	0.924	0.650	0.763
WOW	1	1	1	Zeus	0.852	0.926	0.885

4.5 Comparison

Table 3 presents a comparison of our method with other models on the same dataset for binary classification tasks. Compared to other models, our model achieves excellent binary classification results, effectively identifying malicious traffic in the dataset. Furthermore, in multi-class classification tasks, our method also achieves comparable accuracy results. It’s worth noting that, unlike other methods that require retraining the model for new classification tasks, our method uses the same model for both binary and multi-class tasks. By applying the pre-trained model to the training data set to obtain a hash database, we can fulfill new classification requirements without the need for retraining, while still achieving excellent classification results.

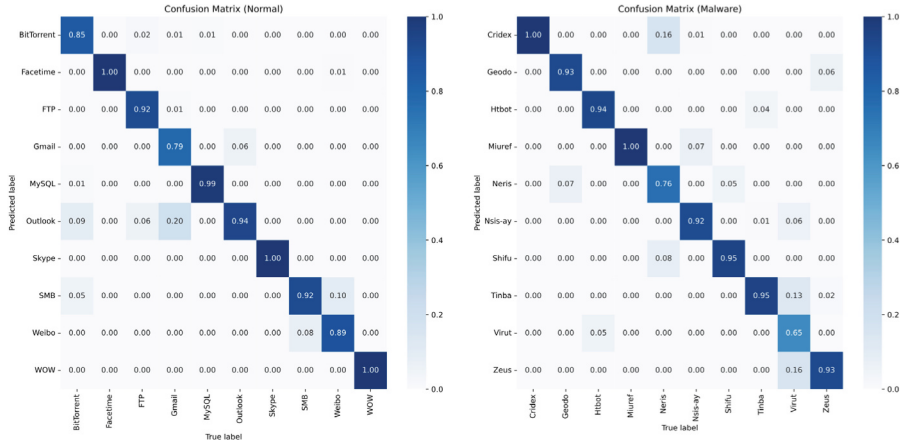


Fig. 4. Confusion matrix of normal traffic and malware traffic.

Table 3. Compared with other method.

Method	ACC_Binary	PR_Binary	RC_Binary	Acc_multiclass(20)
CNN [3]	100	100	100	99.17
DHN [20]	99.94	99.99	/	99.28
LSTM [21]	99.99	99.99	99.99	99.88
Ours	100	100	100	92.3

5 Conclusion

In this paper, we describe a method for categorizing network traffic data by utilizing deep hashing techniques, which involves mapping network traffic into binary hash codes to achieve the goal of classifying network traffic. According to our current understanding, this methodology marks the inaugural utilization of deep hashing within the realm of network traffic classification. It obviates the necessity for intricate feature selection procedures, directly using raw traffic data as input, and does not require supervised training during the process. Utilizing publicly accessible datasets, our experimental findings reveal that this methodology exhibits high precision in classifying network traffic types and possesses robust capabilities for detecting malicious traffic. When benchmarked against cutting-edge traffic classification approaches, our approach can achieve comparable or even superior performance. As part of our ongoing research efforts, we plan to evaluate the performance of this method in other traffic classification tasks, such as identifying encrypted traffic or Tor traffic.

References

1. Biersack, E., Callegari, C., Matijasevic, M.: Data Traffic Monitoring and Analysis. Springer, Berlin (2013). <https://doi.org/10.1007/978-3-642-36784-7>
2. Madhukar, A., Williamson, C.: A longitudinal study of P2P traffic classification. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation, pp. 179–188. IEEE (2006)
3. Wang, W., Zhu, M., Zeng, X., et al.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International conference on information networking (ICOIN), pp. 712–717. IEEE (2017)
4. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
5. Moore, A.W., Zuev, D.: Internet traffic classification using Bayesian analysis techniques. In: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 50–60 (2005)
6. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., et al.: Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access* **5**, 18042–18050 (2017)
7. Wang, W., Zhu, M., Wang, J., et al.: End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 43–48. IEEE (2017)
8. Wang, W., Sheng, Y., Wang, J., et al.: HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* **6**, 1792–1806 (2017)
9. Radford, B.J., Apolonio, L.M., Trias, A.J., et al.: Network traffic anomaly detection using recurrent neural networks. *arXiv preprint [arXiv:1803.10769](https://arxiv.org/abs/1803.10769)* (2018)
10. Jiang, W.: Graph-based deep learning for communication networks: a survey. *Comput. Commun.* **185**, 40–54 (2022)
11. Han, X., Xu, G., Zhang, M., et al.: DE-GNN: dual embedding with graph neural network for fine-grained encrypted traffic classification. *Comput. Netw.* **245**, 110372 (2024)
12. Jianping, W., Guangqiu, Q., Chunming, W., et al.: Federated learning for network attack detection using attention-based graph neural networks. *Sci. Rep.* **14**(1), 19088 (2024)
13. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM* **51**(1), 117–122 (2008)
14. Chen, Z., Zhong, F., Min, G., et al.: Supervised intra-and inter-modality similarity preserving hashing for cross-modal retrieval. *IEEE Access* **6**, 27796–27808 (2018)
15. Cui, H., Zhu, L., Tan, W.: Efficient inter-image relation graph neural network hashing for scalable image retrieval. In: Proceedings of the 2nd ACM International Conference on Multimedia in Asia, pp. 1–8 (2021)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, 25 (2012)
17. 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)
18. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
19. Tan, Q., Liu, N., Zhao, X., et al.: Learning to hash with graph neural networks for recommender systems. In: Proceedings of The Web Conference 2020, pp. 1988–1998 (2020)

20. Wang, B., Su, Y., Zhang, M., et al.: A deep hierarchical network for packet-level malicious traffic detection. *IEEE Access* **8**, 201728–201740 (2020)
21. Hwang, R.H., Peng, M.C., Nguyen, V.L., et al.: An LSTM-based deep learning approach for classifying malicious traffic at the packet level. *Appl. Sci.* **9**(16), 3414 (2019)
22. Jia, Y., Shelhamer, E., Donahue, J., et al.: Caffe: convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678 (2014)



A Trust Management Integrated Bayesian Network Prediction in Vehicular Networks

Hongbin Zhang^{1,2} , Zi Li¹ , Dongmei Zhao² , and Yuelin Liu³

¹ School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang 050000, China

² Hebei Key Laboratory of Network and Information Security, Hebei Normal University, Shijiazhuang 050024, Hebei, China
zhaodongmei666@126.com

³ Zhongxin International College of Engineering, Shenyang Jianzhu University, Shenyang 110168, Liaoning, China

Abstract. There are problems with trust initialization and low evaluation accuracy in the Internet of Vehicles (IoV). This paper proposes a trust management scheme for road prediction based on Bayesian network methods. Firstly, environmental factors are introduced as variables in the Bayesian network for road prediction in trust initialization. The predicted results are used to evaluate vehicle messages for obtain initial trust values and improve model evaluation accuracy. Secondly, in terms of trust processing, an adaptive forgetting factor and cosine similarity are introduced to refine the trust evaluation method, which enhance model evaluation accuracy and resistance to attacks. The experiment shows that the model has a high evaluation accuracy when the proportion of malicious nodes is 40%. The proposed model can effectively identify malicious nodes and has good resistance to switch attacks.

Keywords: Internet of vehicles · Trust management · Trust evaluation · Bayesian network

1 Introduction

With the rapid development of the IoV, there is enormous potential in enhancing road safety and improving traffic efficiency. The IoV perceives the surrounding road environment through onboard sensors and with the help of new generation information and communication technology to achieve efficiently collaborative control of human-vehicle-road-cloud [1, 2]. However, because of the numerous and widely distributed nodes in the IoV, vehicles may receive information from malicious vehicles. Malicious vehicles intentionally broadcast false messages to disrupt the normal driving of other vehicles, which poses a significant threat to the safety and reliability of road traffic.

Many studies suggest that trust management is an effective way to solve the trust problem in the IoV [3]. Trust management can achieve the calculation of vehicle reliability and allocate vehicle trust values, which effectively solve the problems of false

information and malicious vehicles in the IoV [4, 5]. Although trust management can realize the reliability calculation of vehicles, the accuracy of evaluations needs to be improved.

Therefore, this paper proposes a trust management scheme based on Bayesian network prediction. On the one hand, the integration of trust initialization and Bayesian network prediction improve the accuracy of vehicle assessment. On the other hand, the introduction of adaptive forgetting factors and cosine similarity refine the trust evaluation method, which enhance the model's resistance to attacks.

2 Related Work

For example, Tong Xin et al. [6] proposed a multi-factor trust evaluation method based on the Analytic Hierarchy Process (AHP). But it doesn't give a method for calculating the initial trust value. And he introduced a malicious event impact factor, improving the traditional Bayesian approach by increasing the influence of malicious events. There are similar issues in the Social Internet of Things (SIoT). Zhang Hongbin et al. [7] aimed to solve the problems of network sparsity and cold-start in SIoT, and proposed a trust transfer algorithm based on the explicit social relationships and potential features between nodes, which can tap and establish the implicit social relationships among nodes. Wu Haiqin et al. (2015) [8] focused on the need for rapid detection of malicious nodes in the IoV, the concept of negative event influence factors were introduced, which can accelerate the detection of such malicious nodes. She used the Bayesian method to calculate direct trust values and introduced a malicious event impact factor, improving the traditional Bayesian approach by increasing the influence of malicious events. Hu et al. [12] proposed a trust evaluation scheme (REPLACE) for vehicle platooning scenarios on highways, which assisted vehicles in selecting a trustworthy lead vehicle to follow. This scheme primarily evaluated the trustworthiness of the lead vehicle based on feedback data from user vehicles. It can effectively resist false feedback attacks, newcomer attacks, and switch attacks. However, this trust evaluation scheme is only applied able to the problem of selecting the leading vehicle in a vehicle queue, making its application scenario quite limited.

To summarize, the research presents a relatively complete trust management scheme but only consider high evaluation accuracy under normal vehicle interactions. Therefore, this article considers from the perspective of trust initialization, which improve traditional trust management schemes to enhance the accuracy of scheme evaluation.

3 Design of Trust Management Framework

The frame is mainly compose of vehicles and road Side Unit (RSU). The scenario of the trust management framework in this article is shown in Fig. 1.

- 1) Vehicles. The vehicle is equipped with basic computing modules, communication units (On Board Unit, OBU) and the perception module, which calculate vehicle ratings and interact with RSU [10].

2) RSU. Because the capacity of computing and storage is powerful, so the RSU is responsible for collecting nearby messages, broadcasting trust values and accident notifications [11].

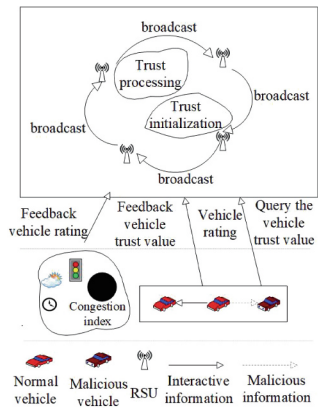


Fig. 1. Scenario of Framework

The framework consists of trust initialization module and trust processing module. The prediction of road conditions is facilitated within RSU, and the trust evaluation is conducted within the vehicles. The initialization module mainly includes road prediction and message evaluation. The prediction results are used to evaluate vehicle messages, which vehicles enable to establish trust even when trust information is scarce. Therefore, the accuracy of the trust evaluation is enhanced. This module provides basis for establishing the initial trust in the trust handling module. The trust processing module includes the collection of vehicle scores and the calculation of trust values. The improved trust calculation method has enhanced the accuracy of the model’s evaluation and its ability to resist attacks. The specific process is shown in Fig. 2.

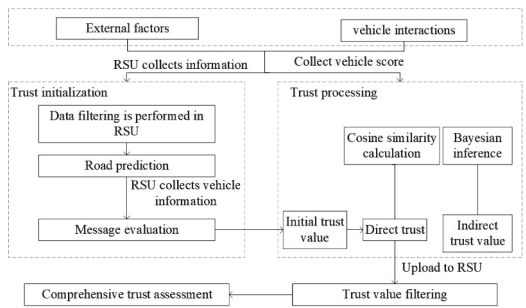


Fig. 2. The trust management framework

3.1 Trust Initialization

Road Prediction. In this paper, the method used in this article for predicting road conditions is Bayesian network learning [11, 12]. The goal of Bayesian network learning is to obtain a Bayesian network structure and the corresponding probability distribution parameters, which the network can effectively describe the dataset's distribution patterns and generate accurate inference results.

Bayesian network learning consists of three parts.

- 1) Variable selection. The selected variables should be relevant and can influence other variables, which ultimately used to establish a Bayesian network.
- 2) Bayesian network structure. The network topology is composed of the network node set and the directed connection set between the network nodes.
- 3) Parameter learning. The network structure estimate the conditional probability distribution of each edge.

Based on this, this article needs to establish a Bayesian network for road prediction. The steps are as follows.

- 1) Based on expert experience, this paper selects weather, time period, visibility, average speed, and congestion index as predictor variables for the Bayesian network, labeled as N. Road conditions are used as the prediction outcome, labeled as S. The following are quantified for N and S. $N = \{\text{weather, time, visible, speed, congestion, index}\}$. $S = \{\text{road condition}\}$. The values are as follows (Table 1):

Table 1. Variable value

Variable	Value
weather	W_1 : sunny W_2 : fog W_3 : rain W_4 : snow
time	T_1 : morning T_2 : Noon T_3 : evening
speed	X_1 : 60 km/h X_2 : 20–60 km/h X_3 : <20 km/h
congestion index	C_1 : <1.4 C_2 : 1.5–1.9 C_3 : >2.0
visible	V_1 : <50 m V_2 : 50–100 m V_3 : 100–200 m V_4 : >200 m
road condition	S_1 : unblocked S_2 : amble S_3 : jam

- 2) Based on expert knowledge and historical data from IoV, structural learning is conducted and the causal relationships between variables are represented by the following formula.

$$\text{speed} = \alpha_1 \text{weather} + \alpha_2 \text{time} + \alpha_3 \text{visible} + \epsilon_1 \quad (1)$$

$$\text{congestion index} = \beta_1 \text{weather} + \beta_2 \text{time} + \beta_3 \text{visible} + \epsilon_2 \quad (2)$$

$$\text{road condition} = \gamma_1 \text{speed} + \gamma_2 \text{congestion index} + \epsilon_3 \quad (3)$$

The coefficients indicating the degree of influence are α , β , and γ , with their subscripts 1, 2, and 3 corresponding to different coefficients. And the error term is ϵ . After determining the causality of each variable, the network structure can be constructed.

- 3) Parameter learning is conducted based on the determined network structure to establish the conditional probability tables between nodes. This paper uses Bayesian estimation method for parameter learning.

$$P(D|h) = \frac{P(D|h)P(h)}{P(D)} \quad (4)$$

$P(D)$ represents the prior probability of D , D represents historical messages in the IoV. $P(h)$ represents the prior probability of h , where h represents candidate hypotheses in the hypothesis space H . $P(D|h)$ represents the probability of training data D when h holds, $P(h|D)$ representing the posterior probability of h .

Then, the trained Bayesian network model can be used to predict the probabilities of S_1 , S_2 , and S_3 .

Message Evaluation. This article uses the predicted results S to assess the initial trust values. Additionally, a suitable evaluation plan needs to be developed to minimize assessment errors. In this article, the probability of S_1 , S_2 , or S_3 that is greater than 50% is used as the final prediction result Y . The process is illustrated in Fig. 3.

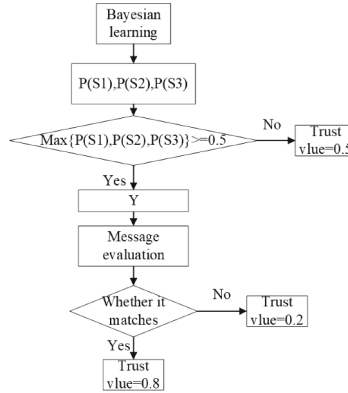


Fig. 3. Evaluation process

After the vehicle obtain the initial trust value, the predicted probability is always less than 50% and especially there are newly registered vehicles or vehicles with no interaction record in a short period of time, which result in inaccurate prediction of road conditions. To solve this problem, this paper give the vehicle of initial trust value is 0.5.

3.2 Trust Processing

In the context of the IoV, the trust management module responsible for handling trust relationships between nodes. The initial trust values obtained from the trust initialization

module will serve as an evaluation factor in the trust processing to enhance the accuracy of the assessment. To improve the traditional methods of calculating trust values, this paper introduces an adaptive forgetting factor and cosine similarity.

Adaptive Direct Trust Assessment. The trust management module must have the ability to identify malicious vehicles and accurately assess vehicle credibility. Vehicles that engage in malicious behavior need to receive timely punishment to reduce their credibility and the impact of negative events should be greater than that of positive events. Therefore, an adaptive forgetting factor β is introduced in this paper [13]. Once a vehicle exhibits malicious behavior, the trust value should quickly converge within the threshold and as a malicious vehicle. It is difficult to forget their behavior for the IoV. The calculation is as follows.

$$\beta = C * (1 - D_{i,j}^{\text{old}}) \quad (5)$$

$D_{i,j}^{\text{old}}$ represents the trust value at the previous moment, C is a fixed parameter with a value of 2.

To conduct direct trust assessment, the formula is as follows [14].

$$DT_{i,j}^{(t)} = \frac{p_{i,j}^{(t)} + 1}{p_{i,j}^{(t)} + n_{i,j}^{(t)} * \beta + 2} \quad (6)$$

$p_{i,j}^{(t)}$ represents the total number of positive feedback from node i to target node j at time t and $n_{i,j}^{(t)}$ represents the total number of negative feedback. If the direct trust value of vehicle i is less than 0.5, then $\beta > 1$, increasing the weight of negative feedback to reduce the trust value of malicious vehicles. And β can be dynamically adjusted as vehicle trust changes. It takes a long time for the trust value to return to normal levels once a vehicle acts maliciously.

Because this paper introduce a time decay function and a sliding window mechanism, so the vehicle i can receive the latest feedback from vehicle j , which improve the traditional methods of calculating the total amount of positive and negative feedback. The improved formula is as follows.

$$p_{i,j}^{(t)} = \sum_{q=1}^m e^{-\lambda(t-t_q)} * p_{i,j}^{t_q} + pf \quad (7)$$

$$n_{i,j}^{(t)} = \sum_{q=1}^m e^{-\lambda(t-t_q)} * n_{i,j}^{t_q} + nf \quad (8)$$

$e^{-\lambda(t-t_q)}$ represents the time decay function, λ represents the control decay rate, and m is the size of the sliding window. pf and nf denote the quantities of positive feedback and negative feedback at time t , respectively.

The resulting modified Bayes formula is as follows.

$$DT_{i,j}^{(t)} = \frac{p_{i,j}^{(t)} + 1}{p_{i,j}^{(t)} + n_{i,j}^{(t)} * \beta + 2} \quad (9)$$

Subsequently, the direct trust values of vehicles will be filtered, and only those exceeding the trust threshold will proceed to the next evaluation step.

Indirect Trust Assessment Using Cosine Similarity. Cosine similarity is a method used to measure the degree of similarity between two vectors in terms of their direction in a multidimensional space [15]. In this paper, it is defined as the evaluation of the same vehicle by vehicle i and K_i . This method converts the evaluation of a single vehicle's assessment into evaluations from multiple vehicles which improve the accuracy of indirect trust assessment. The calculation method is as follows.

$$Sim(A, K_i) = \frac{|\sum_{j=1}^m r_{aj} r_{sj}|}{\sqrt{\sum_{j=1}^m (r_{aj})^2} \sqrt{\sum_{j=1}^m (r_{sj})^2}} \quad (10)$$

The successful interaction rate scoring vectors of vehicle node A and vehicle node K_i for m vehicles within a period of time t are respectively represented as $r_a = [r_{a1} r_{a2} \cdots r_{am}]$ and $r_s = [r_{s1} r_{s2} \cdots r_{sm}]$, and Sim indicates the similarity of two nodes, $Sim \in [0, 1]$.

The method for calculating indirect trust values traditionally is as follows.

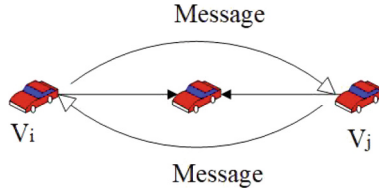


Fig. 4. Indirect interaction

Figure 4 indicate that the interaction between vehicle i and vehicle j relies on the evaluation of a third-party vehicle, but the trustworthiness of the third-party vehicle cannot be determined, which lead to inaccurate indirect trust assessment. Therefore, this paper uses cosine similarity as the weight for indirect trust assessment. The calculation method is as follows.

$$T_r(A, B) = \sum_{i=1}^n Sim(A, K_i) T_d(K_i, B) \quad (11)$$

$T_d(K_i, B)$ represents the direct trust value. Positive evaluations are quantified as 1 and negative evaluations as 0 that can directly observe assessment differences. The greater the discrepancy between different vehicles' evaluations of the same vehicle, the lower the cosine similarity and the lower the indirect trust value.

This paper calculates the comprehensive trust value using a weighted summation method, the weights for initial trust, direct trust, and indirect trust are as follows: $\omega_1, \omega_2, \omega_3$. Because the initial trust value can improve the accuracy of trust evaluation, and direct interaction between vehicles is more reliable than recommended values, so set $\omega_1 = \omega_2 > \omega_3$. The formula for calculating the comprehensive trust value is as follows.

$$T_{(t)} = \omega_1 T0 + \omega_2 DT_{i,j}^{(t)} + \omega_3 T_r(A, B) \quad (12)$$

4 Experiment Analysis

The operating system environment of this paper is Windows 11. The Bayesian network constructed using Netica. The road network and traffic flow are simulated using SUMO, with OMNeT++ controlling the entire simulation process and handling the application logic within Veins. Finally, this paper realizes the simulation of the entire vehicle network. The simulation parameters are shown in Table 2.

Table 2. Simulation parameter

parameter	m	C	Thr1	λ	ω_1	ω_2	ω_3
value	5	2	0.5	0.05	0.4	0.4	0.2

This paper compares the evaluation accuracy under different models, and its calculation formula is as follows.

$$\text{Precision} = \frac{P_{A|A}}{P_{A|H} + P_{A|A}} \quad (13)$$

$P_{A|A}$ indicates that a malicious node is detected as a malicious node, and $P_{A|H}$ indicates that a normal node is detected as a malicious node.

The experiment selected a portion of the Shijiazhuang city map to generate the road network for simulating vehicle interactions.

Since the RFTM [16] and NTM [17] models are similar to the scenarios in this paper and do not consider the impact of trust initialization on the models. So these two models are used for proving the feasibility of the plan.

The experiment simulate the interactions of 100 nodes and focus on two scenarios: speeding and spreading false information. Therefore, nodes that exceed the road's speed limit and spread false information are defined as malicious nodes. In the experiment, 10 malicious nodes are set up and 10 normal nodes are selected.

Figure 5 shows the effect of trust initialization on model accuracy. With the increasing proportion of malicious nodes, the accuracy of evaluation tends to decline slowly, but the accuracy of the proposed model is higher than that of RFTM and NTM models. It can still have more than 85% accuracy when the malicious node reaches 40%. The main reason is that during the trust initialization phase, nodes that spread false information are assigned lower initial trust values through the evaluation of their messages, reducing the impact of malicious nodes in subsequent evaluation. And the introduced adaptive forgetting factor β ensures that their malicious behavior is retained in the IoV for a long time. It keeps the trust values of malicious nodes at a low level. Those below the trust threshold will be filtered out and no longer participating in the current evaluation.

To test the impact of the adaptive forgetting factor on the model, nodes are set as normal from 0–1400 s, malicious from 1750–2100 s, and normal again from 2400–4000 s. The results are compared with the RFTM model and as shown in Fig. 6.

From the figure, it can be observed that the trust values increase steadily from 0–1400 s, but the proposed model rises more slowly. This is because the forgetting factor does

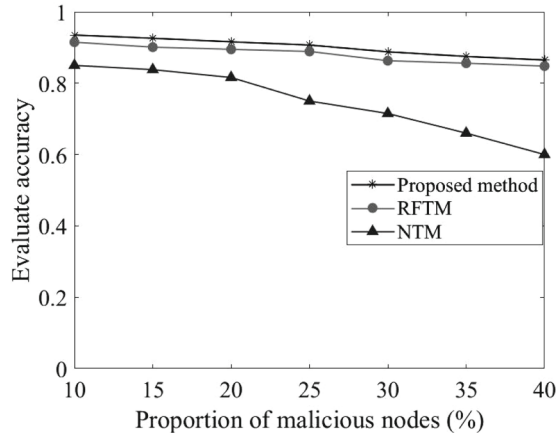


Fig. 5. Evaluation accuracy

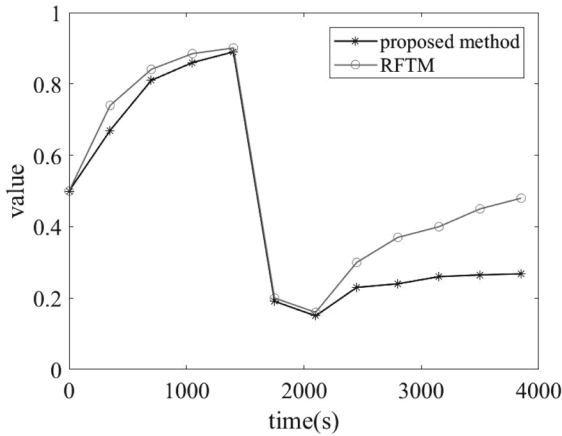


Fig. 6. Node trust value

not allow for a rapid increase in trust even if the nodes perform well. After malicious behavior is executed, both models can identify the malicious vehicles, causing a sharp decline in trust values. However, once the malicious vehicles behave normally again, in the RFTM model, the node’s trust value continues to rise quickly, whereas in the proposed model, the trust value of malicious nodes remains lower. This is because the forgetting factor amplifies the impact of negative events, reducing the influence of malicious nodes on subsequent behavior.

Switch attack is a type of attack that involves forging or tampering with the communication data between nodes. The Switch Attack specifically refers to an attacker who sometimes behaves well and at other times behaves poorly, thereby evading detection by trust management mechanisms. In order to test the anti-switch attack ability of the model, the experiment was carried out for 1.9 h, a total of 10 rounds of experiments. In

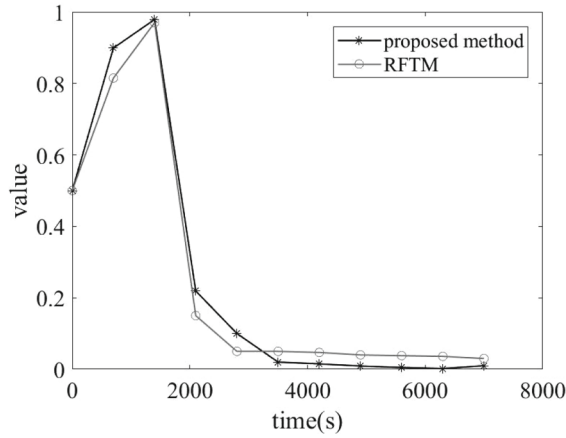


Fig. 7. Comparison of attack resilience

the first two rounds, the vehicles performed well, with their behavior being malicious in the third round.

As shown in Fig. 7, once a vehicle performs a switch attack, its trust value rapidly decrease, eventually approaching zero. Although the convergence rate is slightly lower than that of RFTM model in the period 2000–2700 s, the accuracy is improved by nearly 0.1. This is because once a node performs malicious behavior, it receives a lower trust value during the trust processing phase. And due to the sliding windows and time decay factors are introduced in Eqs. (7) and (8), the model is able to obtain the latest feedback messages from switch attack nodes. At the same time, because of the adaptive forgetting factor that it takes a long time to restore the previous trust value, which has a good ability to resist switching attacks.

5 Conclusion

The paper proposes a trust management scheme based on Bayesian network prediction, which improves the accuracy of vehicle trust assessment in situations involving trust initialization issues. On one hand, considering trust initialization, environmental factors are used for road prediction, and the results are used to assess vehicle messages. On the other hand, the traditional trust processing methods have been improved by introducing adaptive forgetting factors and cosine similarity. Experimental simulation results show that the trust evaluation accuracy of the proposed model is higher and better anti-aggression when trust information is insufficient. The next stage will look at a broader range of predictions that can be made and evaluated in the face of various types of information.

Acknowledgement. This research was supported by National cultural and tourism science and technology innovation project (2020), S&T Program of Hebei No. 22567606H, No. 21310101D.

References

1. Li, J., Zhang, J., Gao, G., et al.: Consensus algorithm for vehicular networks based on reputation mechanism. *Comput. Eng.*, 1–14 (2024)
2. Tang, X., Liang, Y., Chen, W.: Multi stage federated learning mechanism for vehicle networking targeting non-independent and identically distributed data, 1–15 (2024)
3. Wang, X.: Research on Trust Evaluation and Security Management in Vehicular Networks Based on Blockchain Lanzhou University of Technology
4. Azad, M.A., Bag, S., Hao, F., et al.: Decentralized self-enforcing trust management system for social Internet of Things. *IEEE Internet Things J.* **7**(4), 2690–2703 (2020)
5. Chen, J.M., Li, T.T., Panneerselvam, J.: TMEC: a trust management based on evidence combination on attack-resistant and collaborative Internet of vehicles. *IEEE Access* **7**, 148913–148922 (2018)
6. Tong, X.: Research on Adaptive Access Control Trust Mechanisms in Vehicular Networks. Lanzhou University of Technology (2022). <https://doi.org/10.27206/d.cnki.gsgsu.2022.000934>
7. Zhang, H., Fan, F., Zhao, D., et al.: Trust evaluation model for social internet of things based on implicit social relationships. *J. Netw. Inf. Secur.* **9**(02), 56–69 (2023)
8. Wu, H., Wang, L.: Trust-distance-based malicious node detection method for vehicle networks. *Comput. Sci.* **42**(08), 157–160+174 (2015)
9. Hu, H., Lu, R., Zhang, Z., et al.: REPLACE: a reliable trust-based platoon service recommendation scheme in VANET. *IEEE Trans. Veh. Technol.* **66**(2), 1786–1797 (2016)
10. Yang, Z.: Research on Security Mechanisms and Key Technologies for Vehicular Networks. Beijing University of Posts and Telecommunications, Beijing (2019)
11. Wang, C., Guo, Y.: A vehicle network trust management scheme based on logistic regression and blockchain. *Comput. Eng. Appl.* **60**(01), 281–288 (2024)
12. Xu, H.: Traffic accident analysis and prediction research based on improved Bayesian networks. Hangzhou Dianzi University (2024). <https://doi.org/10.27075/d.cnki.ghzdc.2023.000750>
13. Zhang, J.: Research and Implementation of Trust Management Mechanisms for Vehicular Networks. Beijing University of Posts and Telecommunications, Beijing (2020)
14. Ganeriwal, S., Srivastava, M.B.: Reputation-based framework for high integrity sensor networks. *ACM Trans. Sens. Netw.* **4**(3), 66–77 (2004)
15. Ogundoyin, S.O., Kamil, I.A.: A fuzzy-AHP based prioritization of trust criteria in fog computing services. *Appl. Soft Comput.* **97**(80), 106789–106802 (2020)
16. Sun, S.: Research on Trust Management and Evaluation Methods in Vehicular Networks. University of Technology, Xi'an (2023). <https://doi.org/10.27398/d.cnki.gxalu.2023.000229>
17. Wang, T., Zhang, G., Bhuiyan, M.Z.A., et al.: A novel trust mechanism based on fog computing in sensor–cloud system
18. Zhao, X., Xu, H., Wang, T., Jiang, X., Zhao, J.: Research on multidimensional system security assessment based on AHP and Gray Correlation. In: Han, W., Zhu, L., Yan, F. (eds.) CTCIS 2019. CCIS, vol. 1149, pp. 177–192. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-3418-8_13



Research and Development of Distribution Line Defects and Violations Detection Method and Edge Computing System Based on Improved PPYOLOe

Zaixin Chen¹, Peike Du²(✉), Tihua Yan³, and Kaijun Wu²

¹ State Grid Sichuan Nanchong Power Supply Company, Sichuan 637000, China

² State Grid Sichuan Liangshan Power Supply Company, Sichuan 615000, China
dupeike@163.com

³ Xidian University, Xi'an 710126, China

Abstract. In recent years, with the rapid development and widespread application of image recognition technology, deep learning based object detection algorithms have been studied and applied to distribution lines to some extent. Aiming at the problems of low efficiency of traditional target detection algorithm in distribution line scene image detection and bottleneck of cloud server prediction service computing power, the deployment of improved PPYOLOe model algorithm on Edge device is proposed to realize the real-time detection task on distribution lines. Firstly, the use of ESE attention mechanism after the backbone network and CA attention mechanism after the neck feature extraction network has improved the model's feature extraction ability in channels and positions; Secondly, the feature matrices output by the backbone network were fused, which improved the calculation speed of the model and also improved its robustness; Finally, the obtained model was quantified and deployed on the NVIDIA Jetson Xavier of the Aarch64 architecture, achieving edge prediction services. The experimental results show that on the locally collected dataset, the average detection accuracy (mAP) of the algorithm reaches 89.33%. After lightweight, the inference acceleration ratio reached 2.24, with an average detection accuracy of 85.84%, and the processing frames per second (FPS) on Xavier was 4.5.

Keywords: Deep Learning · Object Detection · Edge Computing · Computer Vision

1 Introduction

The power system is a vast system involving national image, people's livelihood, and ensuring the orderly operation of social enterprises. Maintaining the stability and safety of transmission and distribution lines is of great significance to organizations at all levels. However, distribution lines are often intricate, involving environments such as mountains, lakes, urban areas, and remote suburbs. Due to the complexity of geographical

environments, the monitoring and maintainability of distribution line faults are extremely challenging [1]. In the past, traditional technicians conducted inspections by carrying professional equipment and patrolling along transmission lines. China boasts the longest total length of distribution lines in the world, and there is significant uncertainty in the damage to distribution line components. Therefore, manual inspections have limitations such as weak purposefulness, high costs, long duration, and significant safety hazards [2, 3].

In recent years, technologies such as small object detection, human keypoint detection, and human behavior semantic analysis based on deep learning have experienced rapid development in research due to the continuous refinement and optimization of their frameworks, and many structures also exhibit high scalability, enabling the research of image detection technology to achieve rapid progress and practical applications in many engineering projects. In the power system, tasks involving image detection, semantic recognition, and defect identification, such as “correct wearing of safety helmets,” “proper use of safety belts,” “warning of high-altitude littering,” and “loosening or misalignment of transmission tower foundations,” have relevant applications. However, due to the complexity of images involved in actual production environments, algorithms often have high fitting degrees to specific environments, resulting in generally poor generalization and robustness performance, with significant room for improvement in terms of missed detection rates for complex semantic logic and false alarm rates in hazardous environments. Therefore, improving algorithm models for distribution equipment scenarios and enhancing the popularization of artificial intelligence in the power system are of significant importance.

In the context of distribution line scenarios, there are two main issues: complex scenes with limited datasets and problems arising from insufficient cloud computing power. Existing methods such as GAN-based simulated sample data missing processing methods can expand training samples by fusing specific targets into different scenes using GAN, addressing the issue of missing datasets [4]. However, such data is mostly synthesized using images from the internet, which often have inconsistent resolutions with actual scenes and cover a limited range of scenarios. This paper adopts methods such as on-site posing and camera image capture to obtain image sets closest to real-world scenarios, utilizes corresponding data augmentation methods to expand the dataset, and maximizes image utilization efficiency.

Regarding the issue of insufficient cloud computing power, companies generally adopt approaches such as purchasing more graphics cards or multiple servers to address the problem. However, this approach is not sustainable and involves high costs for companies. Moreover, as the number of servers increases, network bandwidth becomes a significant bottleneck limiting the speed of prediction services. This paper adopts edge computing to deploy prediction services, launching the algorithm model as a docker on the host (NVIDIA Jetson Xavier) of edge cameras and exposing an interface to receive image URLs and return prediction results in a specified format.

In summary, this paper analyzes the images collected in distribution line engineering scenarios, proposes data annotation guidelines and data cleaning standards. Based on the Paddle framework’s end-to-end general detection model PPYOLOe, gains are achieved through the insertion of CBAM (Convolutional Block Attention Module) attention, CA

(Coordinate Attention) attention mechanisms, lightweight processing of the detection head PPYOLOe Head, and post-processing logic writing, resulting in improved recall and precision performance of the model in the distribution line test environment.

At the deployment level, this paper primarily utilizes Python language, the Flask web application framework, and other technologies to package the algorithm inference code into a prediction service and expose an interface for receiving image URLs. The camera saves and proxies captured images, which are sent to the interface in the form of URLs. The algorithm then packages relevant violation alarm information related to the images, and finally returns JSON-formatted alarm information, location information, and other details.

2 Object Detection Technology Based on Deep Learning

2.1 Image Recognition, Edge Computing, and Convolutional Neural Networks

Image recognition, as an important topic in the field of computer vision technology, has witnessed rapid development and a vibrant technological ecosystem in recent years. This technology involves processing and analyzing real-world images to enable computers to understand and form a model capable of identifying different patterns, targets, and objects [5]. Utilizing artificial intelligence image recognition technology can effectively detect specific target objects in various scenarios. Simultaneously, replacing manual inspections with computers can significantly liberate labor, especially for tasks involving repetitive labor. Moreover, adopting artificial intelligence for target detection offers higher accuracy and recall rates.

Recent advances in edge computing have significantly enhanced the performance of real-time tasks such as object detection, especially in distributed industrial environments like power systems. By reducing latency and processing data closer to the source, edge computing offers a more efficient solution than cloud computing for real-time detection tasks in challenging environments [6].

Convolutional neural networks (CNNs) are network structures primarily composed of input layers, convolutional layers, downsampling layers (pooling layers), fully connected layers, and output layers. Over the years, CNNs have been widely applied in image recognition and object detection, covering various scenarios such as pedestrian detection, human keypoint detection, human behavior recognition, and pose recognition [7]. The flexibility and robustness of various algorithmic models have been significantly enhanced through researchers' development, enabling artificial intelligence to achieve maximum iteration, expansion, and practical application.

2.2 Related Models

PP-YOLOv1, PP-YOLOv2. PP-YOLOv1 and PP-YOLOv2 are two of the primary models in PaddleDetection, a toolkit developed by Baidu's PaddlePaddle team. PaddleDetection primarily provides a wealth of components for contemporary excellent models, as well as tools for writing validation and testing metrics. The baseline source code, attention mechanism development, and post-processing logic organization processes used in

this paper were all completed based on the PaddlePaddle framework. Among them, the PP-YOLO series of general object detection models boasts advantages in high flexibility, robustness, and stability. On the COCO test-dev2017 dataset, PP-YOLOv1 achieved a precision of 45.9%, with an inference speed of 72.9 FPS in FP32 on a single V100 card, and 155.6 FPS in FP16 with TensorRT enabled on V100. PP-YOLOv2, on the other hand, achieved a 3.6% improvement in prediction accuracy compared to PP-YOLOv1 at the same computational speed, reaching 68.9 FPS per second with an input size of 640×640 , surpassing YOLOv5. Additionally, the PaddlePaddle framework contains machine learning migration tools for deployment on different platforms, enabling cross-compilation of prediction operator libraries from Linux amd systems to armv8 systems, perfectly meeting the needs of model training and edge deployment in this paper.

The PP-YOLO series of object detection algorithms are composed of structures such as an image input terminal, a backbone network, feature pyramid networks (FPN), bounding box (bbox), and category prediction layers.

Among them, PP-YOLOv1 employs ResNet50 as its backbone network, utilizing deformable convolution in the third stage. For the neck part, PP-YOLOv1 adopts PP-YOLOFPN, a bottom-up feature fusion approach. In the Head part, PP-YOLOv1 follows the detection head of YOLOv3, obtaining the desired number of channels through a 3×3 convolution followed by a 1×1 convolution. The loss calculation for the detection head is also consistent with that of YOLOv3, which involves predicting through the three-layer vector values output from the Head of the entire network, weighted summing the loss values for classification, confidence, and the position and size of the predicted box, ultimately calculating the gradients and adjusting the parameters within the network through backpropagation [8]. Grid Sensitivity is used in the post-processing part of the model to address issues arising when the center point of a predicted box is located on a grid edge.

Additionally, PP-YOLOv2 builds upon the PP-YOLO baseline model with numerous extended experiments and model improvements. Firstly, PP-YOLOv2 adds a detection neck that constructs high-level semantic feature matrices for images of all input scales. Simultaneously, it changes the original neck structure FPN to PAN (Path Aggregation Network), a structure that fuses feature information from top to bottom [9]. Secondly, PP-YOLOv2 follows the example of YOLOv4 and YOLOv5, changing the activation function to the Mish function, which is formulated as:

$$f(x) = x \cdot \tanh(\zeta(x))$$

Where $\zeta(x) = \ln(1 + e^x)$, essentially a softmax activation function. Simultaneously, PP-YOLOv2 increases the size and variety of input dimensions, thereby enhancing the ability to capture small-scale information from images. Furthermore, PP-YOLOv2 utilizes IoU-aware loss to improve performance, adopting a soft label format, with the loss calculation formula being:

$$Loss = -t \cdot \log(\sigma(p)) - (1 - t) \cdot \log(1 - \sigma(p))$$

Where t is the IoU value between the predicted anchor box and the ground truth box, $\sigma(\cdot)$ is the sigmoid activation function, and p is a balancing parameter initialized as 0 in PP-YOLOv2.

PPYOLOv2 has demonstrated improvements in both accuracy and speed on edge devices compared to earlier models, making it particularly suitable for real-time object detection in constrained environments. A comparison with other models such as YOLOv5 and YOLOX has shown that PPYOLOv2 offers a favorable balance between detection accuracy and computational efficiency, especially when deployed on devices like NVIDIA Jetson Xavier [10].

PPYOLOe. PPYOLOe is a new model improved based on PP-YOLOv2. In terms of the backbone network, PPYOLOe draws on the practices of YOLOv5 and YOLOX, applying the concepts of residual networks and dense networks to the basic module CSPResBlock in the backbone network. Its network structure is shown in Fig. 1.

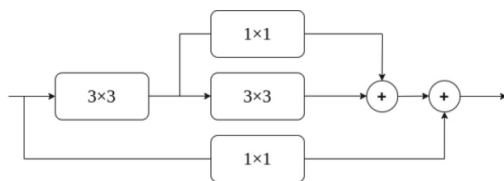


Fig. 1. CSPResBlock structure diagram

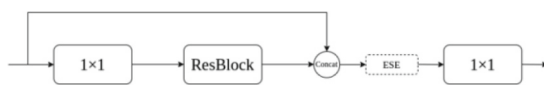


Fig. 2. CSPResStage structure diagram

This module transforms the operation of merging channel numbers into a simple operation of adding feature maps, and during the prediction phase, it enhances the results in the channels through the ESE (Effective SE) attention mechanism. Based on this basic module, CSPResStage is designed, and its network structure is shown in Fig. 2.

This stage uses the SiLU activation function, whose formula is $f(x) = x \cdot \text{sigmoid}(x)$. It effectively avoids neuron death while exhibiting superior nonlinear properties compared to activation functions like ReLU, enabling it to perform better on small sample data.

At the same time, PPYOLOe still utilizes PAN as the Neck part of the network structure, as shown in Fig. 3. However, the Baidu team believes that the features in the Neck have already approached their extreme values, so they removed the shortcut and ESE layers after the Neck.

3 Improvements to the PPYOLOe Algorithm

PPYOLOe is a single-stage Anchor-free model launched by Baidu's team in 2022, based on PP-YOLOv2. It exhibits remarkable performance, outperforming other single-stage models such as YOLOv5 and YOLOX in terms of speed, accuracy, and generalization ability on public datasets. On the COCO test-dev2017 public dataset, the standard-sized

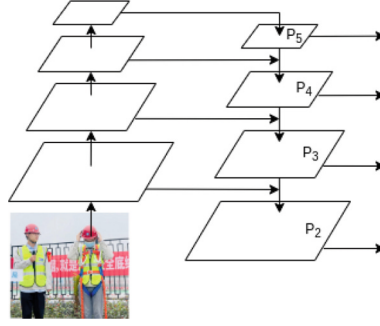


Fig. 3. CSPPAN structure diagram

model of PPYOLOe achieves an mAP value of 53.3, with a computational speed of 78.1 FPS on Tesla V100, surpassing historical models like YOLOv5 and YOLOX [11]. The PaddleDetection toolkit provided by Baidu also offers convenient and extensible model source code and deployment tools, providing significant assistance to other researchers' work.

Due to the complex and diverse scenarios in distribution lines, some detection targets, such as small animal detection and waterlogging detection, involve dense small object detection. Meanwhile, tasks like smoke and fire alarms, and construction personnel violations, involve complex detection backgrounds. Basic open-source object detection models do not exhibit high accuracy in these downstream tasks, necessitating improvements to the model's internal structure and the insertion of attention mechanisms.

3.1 CBAM Attention Mechanism

Firstly, a CBAM attention mechanism module is added after the backbone network. Compared to other attention mechanisms that only focus on channels or spatial dimensions, CBAM combines both dimensions to achieve better results, as shown in Fig. 4.

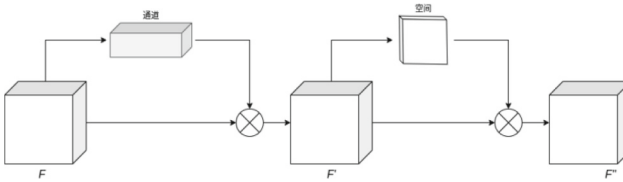


Fig. 4. CBAM structure diagram

$$\begin{cases} F' = F \cdot M_C(F) & (1) \\ F'' = F' \cdot M_S(F') & (2) \end{cases}$$

After the original feature map is input into the CBAM attention module, it first calculates the attention matrix through the channel dimension to obtain the channel

attention map M_C . Then, M_C is multiplied element-wise with the input feature map to obtain the feature map F' . Subsequently, the attention matrix is calculated through the spatial dimension to obtain the spatial attention map M_S . Finally, M_S is multiplied element-wise with the feature map F' to obtain the final result F'' , as shown in formulas (1) and (2) [12].

After adding the attention mechanism, the number of parameters in the model did not increase significantly, nor did the size of the exported model expand noticeably. However, the model's ability to extract features for long-range recognition, identify 3D relationships in 2D images, and resist interference in complex background detection tasks has been improved.

3.2 CA Attention Mechanism

Since some cameras in transmission lines have fixed positions and similar shooting angles in actual scenarios, the probability of some violations occurring at specific locations is higher. Incorporating CA (Coordinate Attention) into the model to increase the model's sensitivity to positional information can effectively improve the recall rate and accuracy of some functional points [13].

CA attention avoids compressing all the spatial information of the feature map into the channels. Instead, it performs pooling on the X-axis and Y-axis of the image separately, obtaining attention maps of $C \times H \times 1$ and $C \times 1 \times W$, respectively. These are then concatenated and downsampled using a 1×1 convolution kernel. Finally, the obtained two sets of attention vectors are used to weight the feature maps in the X-axis and Y-axis directions after being activated by the Sigmoid function.

The CA attention mechanism gives higher weights to the position and orientation information of objects in the image. Due to its plug-and-play nature, extensive experiments have shown that inserting CA after the neck can achieve the highest accuracy improvement for the model. The implementation mechanism of CA is illustrated in Fig. 5.

3.3 Lightweight Detection Head

To lighten the detection head PPYOLOeHead, the original layer attention mechanism in the detection head was replaced with the same ESE attention mechanism used after the backbone network. Simultaneously, the three layers of parameters output by the detection head were fused before performing classification score calculations and loss regression [14]. Comparative ablation experiments revealed that the modified model suffered minimal precision loss while experiencing an increase in inference speed. Additionally, VariFocal Loss was adopted as the loss function instead of the original binary cross-entropy loss, with the formula expressed as follows:

$$VFL(p, q) = \begin{cases} -q(q \log p + (1 - q) \log(1 - p)), & q > 0 \\ -\propto p^\gamma \log(1 - p), & q = 0 \end{cases}$$

Where p represents the scalar value of the classification score vector, and q indicates whether the pixel is background or not. If it is foreground, q is the IoU value between the

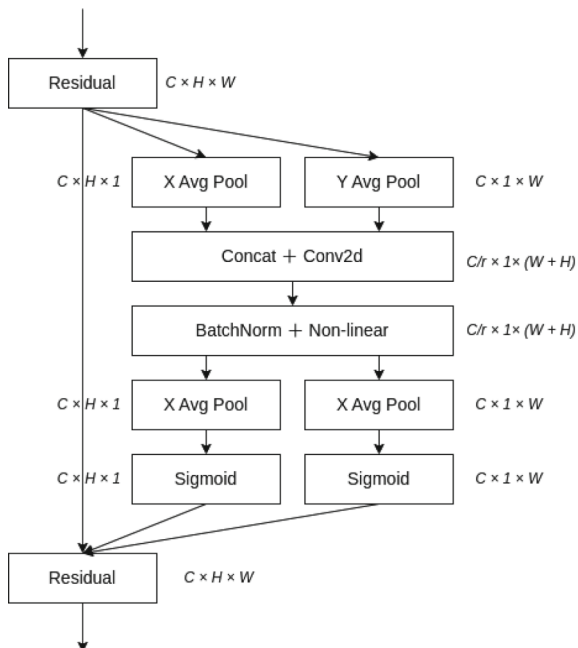


Fig. 5. CA Attention structure diagram

predicted bounding box and the ground truth box at that point; if it is background, q is 0. Both γ and α are adjustable scaling factors, and after testing and parameter tuning, the selected values are $\gamma = 2.0$ and $\alpha = 0.75$. The overall network structure of the improved PPYOLOe is illustrated in Fig. 6.

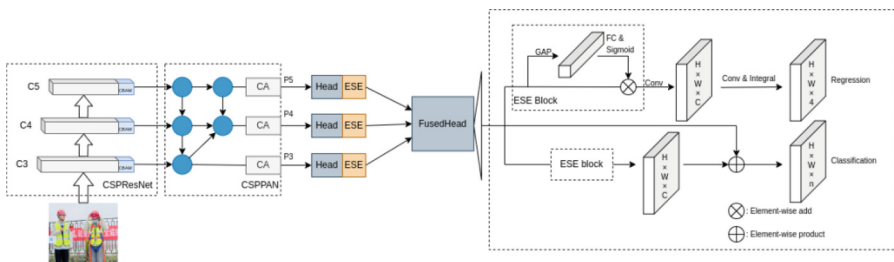


Fig. 6. PPYOLOe improved structure diagram

4 Experiment and Result Analysis

4.1 Training Environment and Evaluation Metrics

The experimental software environment for this paper is based on the Ubuntu 18.04 operating system, Python 3.7.16, paddlepaddle-gpu 2.4.1, utilizing a GPU (Nvidia GeForce GTX 3090) with the CUDA 12.1 acceleration library.

The experimental results use recall, precision, mean average precision (mAP), and frames per second (FPS) as evaluation metrics. The availability and effectiveness of the algorithmic models are assessed by comparing the metrics of different models. All the formulas are as follows. TP stands for true positives, FP stands for false positives, and FN stands for false negatives.

Recall: Represents the proportion of samples predicted as true among all actual true samples by the algorithm.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Precision: Represents the proportion of samples correctly predicted among all samples predicted as true by the algorithm.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

AP (Average Precision) refers to the average precision of prediction for a single category. NC represents the number of categories (in this paper, NC is 203). The formula for calculating the mean Average Precision (mAP) is the sum of the precisions of all categories divided by the number of categories, which is used to evaluate the overall performance of the algorithm model in the object detection task. In the experiment, the IoU threshold used for calculating mAP is set to 0.5.

$$mAP(\%) = \frac{\sum AP}{NC} \times 100\%$$

Frames Per Second (FPS): It is used to analyze the prediction speed of the algorithm model during inference. The time to predict an image includes preprocessing (Pre) time, inference (Inference) time, and non-maximum suppression (NMS) time, with the unit being hertz (Hz).

$$FPS = \frac{1000}{Pre + Inference + NMS} \quad (5)$$

The network parameter configuration for model training is as follows: Momentum is set to 0.9, weight decay to 0.0005, learning rate warm-up for 2 epochs, decay step size of 45, learning rate decay factor of 0.1, and a total of 80 epochs of iteration [15, 16].

4.2 Experimental Dataset

The datasets for distribution line defects and construction personnel violations are both derived from a private database of a certain company. After screening for blurry, overexposed, and underexposed images and removing those with improper labels and mismatched features, a total of 140,176 images were obtained. These images cover 17 equipment-related functional points and 10 human-related functional points, with each category involving interfering factors such as different backgrounds and varying degrees of occlusion. The training set, validation set, and test set were divided in a ratio of 9:0.5:0.5.

The scenarios involved include indoor substation images, outdoor transmission line monitoring images, outdoor high-altitude operation monitoring images, street construction operation monitoring, and more. Some of the images are samples taken during actual work, while others are posed for creating image features [17].

4.3 Training Sample Data Augmentation

Before the training set enters the model, data augmentation operations are performed on the image samples, including the following methods:

Random Pixel Color Distortion. With a 0.5 probability, pixel transformations are applied to the image to change its hue, saturation, sharpness, brightness, and other values, significantly improving detection tasks that are demanding on background materials and lighting conditions, such as waterlogging detection.

Random Image Expansion. With a 0.5 probability, the image is expanded by first selecting a random expansion ratio from a preset value array, calculating the expanded image size, and then converting it into a new image using initialized pixel values (123.675, 116.28, 103.53) as the filling pixels.

Random Image Cropping. With a 0.3 probability, cropping operations are performed on the image. Random crop heights, widths, and starting points are obtained from preset crop ratio (2.0, 0.5) and crop size (0.3, 1.0) arrays. Cropping is repeated until the IoU value between the ground truth box and the cropped box is less than 0.7. Finally, the ground truth boxes located within the cropped region are selected.

Random Image Flipping. With a 0.5 probability, the image is horizontally or vertically flipped, especially for images not involving logical scenarios such as “high-altitude operations.”

4.4 Experimental Results and Comparative Analysis

The pre-trained weights used for the improved PPYOLOe algorithm were obtained from training on the VisDrone public dataset.

Ablation Experiments. To validate the effectiveness of the proposed improvements in the PPYOLOe algorithm model, including the addition of the CBAM attention mechanism, CA attention mechanism, and lightweight detection head, ablation experiments

were conducted on the test set. The test results were used to assess the impact of the improvements on the model’s detection capabilities and algorithm performance under the same experimental conditions. The input image resolution was set to 640×640 , and the test results after 80 epochs of training are shown in Table 1.

Table 1. Ablation experiment test result

Model	Accuracy (%)	Recall (%)	mAP (%)	FPS
PPYOLOe	75.96	75.55	77.13	7.69
PPYOLOe+CBAM	76.42	76.13	77.95	7.61
PPYOLOe+CA	76.66	76.78	78.06	7.71
PPYOLOe+Lightweight	74.89	75.31	76.91	8.57
PPYOLOe+CBAM +CA+Lightweight	77.93	77.01	80.16	8.20

As shown in Table 1, due to the complex scenes in distribution lines and construction environments, as well as the generally limited variation in camera shooting angles, the addition of both CBAM and CA attention mechanisms has resulted in varying degrees of improvement. Specifically, CBAM increased the accuracy of the test set by 0.46%, the recall rate by 0.58%, and the mAP by 0.82%. Meanwhile, the CA attention mechanism improved the accuracy of the test set by 0.70%, the recall rate by 1.23%, and the mAP by 0.93%. On the other hand, although light weighting the detection head led to a slight decrease in accuracy and recall rate, it increased the FPS by 0.88, enabling almost one additional inference per second. This demonstrates the effectiveness of the algorithmic improvements made.

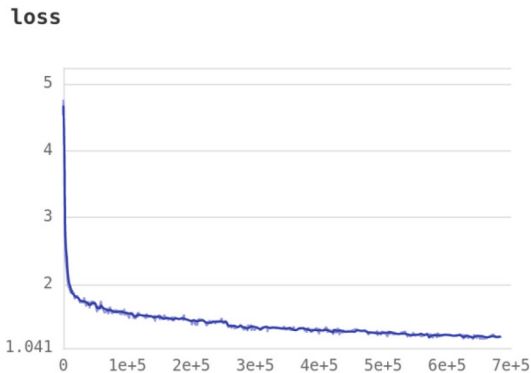


Fig.7. Loss function curve before improvement

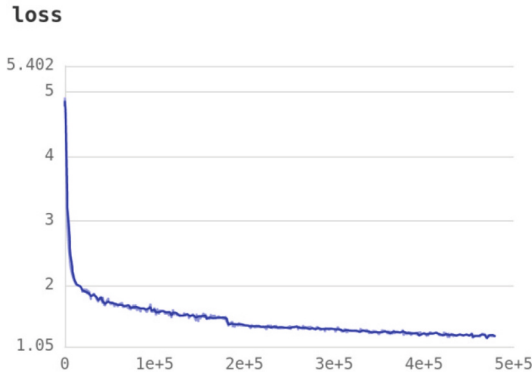


Fig. 8. Loss function curve after improvement

Experiment Visualization and Analysis. As evident from the training loss function curves in Figs. 7 and 8, the improved Loss converges faster, allowing for earlier termination of training and exporting of the corresponding model. This enables the algorithm model to possess better generalization capabilities and robustness, while avoiding overfitting to the training dataset.

Observing the detection results on actual images, the improved detection model shows improvements in scenarios such as wall collapses and damaged tables and chairs that could potentially lead to false positives for waterlogging detection. It also exhibits enhanced capabilities for smoke and fire alarms in open outdoor areas with gray skies and dim lighting conditions. Detection of small animals like mice and snakes in dimly lit environments has also been improved, as well as the detection of semantic features related to human behavior. Below are selected comparative images highlighting improvements in scenarios such as false positives for waterlogging due to wall stains, missed detections of smoke and fire in similar backgrounds, and missed detections of small animals in dim lighting conditions (Fig. 9).

Through the above comparative analysis, it can be concluded that the improved model has better generalization ability and detection effect for detection tasks in complex scenes, improved ability to distinguish similar targets, and simplified detection ability for complex scenes. To a certain extent, it improves the recall and accuracy of detection tasks, reduces false alarm rates and false alarm rates.

5 Model Deployment and Inference Service Construction

The prediction service in this article is deployed in the embedded device NVIDIA Jetson Xavier with arm architecture. The inference code and weight files are loaded on a well configured Docker, exposing the calling interface to the outside world. The client's image on the server will be sent to the interface through a URL proxy, and the inference service directly returns the inference result in JSON format, which includes alarm information, detection box position, and other related information.



- (a1) Wall stains misreporting water accumulation
 (b1) Missing smoke report due to similar background colors
 (c1) Photo too dark, small animal missed reporting



- (a2) Eliminating false alarms after improvement
 (b2) Improved recognition is normal
 (c2) Improved recognition is normal

Fig. 9. Comparison diagram of visual improvement effect

5.1 Reasoning Model Deployment Technology Roadmap

Model Deployment Technology Roadmap. There are various implementation methods for deploying deep learning inference models, including mobile deployment, embedded device deployment, and cloud central server deployment. This article uses embedded devices as cloud servers for the deployment of inference models. The deployed technology roadmap is:

- Convert the model trained above to obtain the relevant structure, parameters, and prediction configuration files of the model.
- Build the armv8 Linux environment and the most basic required environment for model operation in Docker, and store the predicted service code, model exported files, and label information in Docker.
- Accelerate prediction using TensorRT in Docker and start the prediction service code.

The specific method is to first train the network model structure and final parameters under the PaddleDetection suite, and call the API interface to convert them into a prediction model, which includes a prediction configuration XML file, forward prediction weights, and network structure file; Secondly, the following tasks should be done for embedded devices: 1. Flash processing, preparing an appropriate amount of memory space and the correct driver version for inference deployment. 2. Install intermediate files and run them to obtain the basic environment required for Paddle framework prediction. As the Jetson series software maps directly from the local machine when calling platforms such as CUDA in Docker, rather than independently loading Docker, the local

machine needs to be assembled separately. 3. Load the Paddle prediction Docker and start the service, adjust external IO devices such as camera drivers, camera photo resolution, etc.

Jetson Xavier Environment Preparation. Flash processing: Use the latest SDK Manager provided by NVIDIA to burn the firmware of Xavier and install the Ubuntu 18.04 x86_64 system equipped with CUDA and cudnN; Extract the computing platform related files on ARM V8 architecture such as CUDA and cudnN from the assembly files in the system after completion of burning, and use them for the subsequent construction of CUDA in Docker (including libcudnn8_x.x.xx-1+cudaxx. x-arm64. deb, libcudnn8 dev_x.x.x.xx-1+cudaxx. x-arm64. deb, libcudnn8 samples_x.x.xx-1+cudaxx. x-arm64. deb, etc.); After burning, check the installed JetPack package version, CUDA version, and cudnN version, install PaddlePaddle-GPU 2.4.1 in the root directory, and store the previously cross compiled operator inventory locally. Configure the root directory paths of CUDA, cudnN, OpenCV, TensorRT, and compile the entire system once completed.

Docker Environment Preparation. Download a dedicated deep learning image under the armv8 architecture from the NVIDIA Jetson official website. After enabling Docker on Xavier, copy the files that Xavier verifies are feasible into Docker. Cross compile the arm architecture operator library required for the inference model on AMD Linux, and download the arm version of cmake10.2+ to configure the Docker with the correct Python, CUDA, cudnN, and corresponding PaddlePaddle versions; Install the basic OpenCV and TensorRT packages and expose the local 11365 port to the public. The overall deployment architecture is shown in Fig. 10.

5.2 Reasoning Service Engineering Optimization

The Communication Mechanism Between Dockers. Due to the fact that the inference service Docker is located on the same embedded machine as the image processing and old algorithm Docker, using image streaming transmission will consume additional network bandwidth and waste a certain amount of memory space. Therefore, when the old algorithm processes local images, it proxies a URL to the current image. The new algorithm can directly load image related information from this URL into memory and perform real-time inference.

Multi Process Optimization. In the case of a large number of images in the prediction service, a single process will not be able to make real-time predictions on massive images. Therefore, we adopted a message queue and thread pool scheme to optimize the predictor for multiple processes. Each process manages a separate detector, and the output results are returned to the user in a fixed format, which is then reorganized and arranged after acceptance.

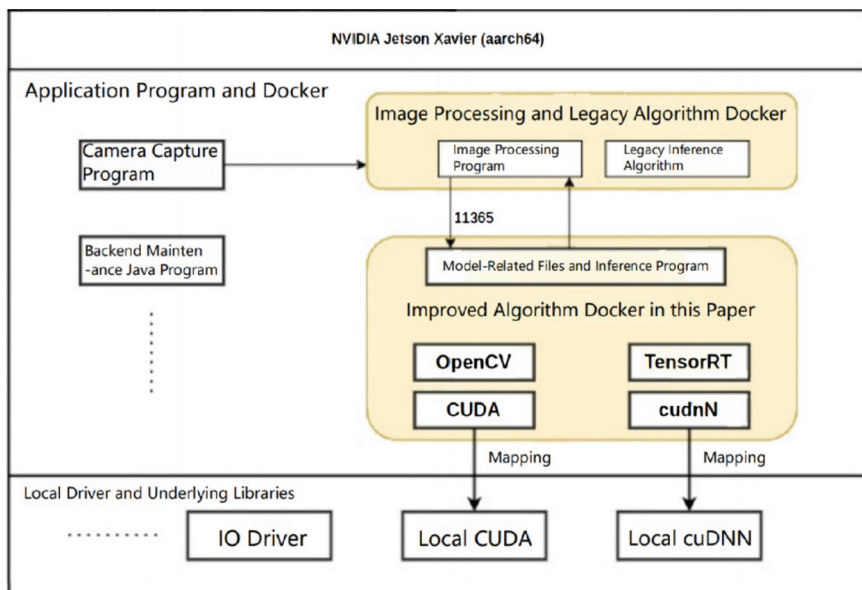


Fig. 10. Deployment architecture diagram of inference service docker on Xavier

6 Conclusion

This paper proposes a detection method based on improved PPYOLOe distribution line defects and violations, and realizes the development and deployment of this detection method in edge computing system. We conducted experiments on various object detection tasks using distribution line and substation monitoring videos as datasets. We improved the PPYOLOe network through CBAM attention mechanism, CA attention mechanism, and detection head lightweight methods, which improved the model's generalization ability, robustness, and detection speed in complex scenarios. This basically meets the accuracy and real-time requirements for detecting equipment defects and violations in distribution line environments.

References

1. Zhuang, H., Yan, F., Mao, X., Xie, Y., Lu, C.: Design and Implementation of a backend monitoring system for active deicing robots for transmission lines. *Autom. Technol. Appl.* **41**(02), 22–27 (2022)
2. Wang, B., Le, H., Li, W., et al.: Improvement of YOLO lightweight network for mask detection algorithm. *Comput. Eng. Appl.* **57**(08), 62–69 (2021)
3. Shi, H., Chen, X., Yang, Y.: Improving the detection method for wearing safety helmets in YOLO v3. *Comput. Eng. Appl.* **55**(11), 213–220 (2019)
4. State Grid Corporation of China Information and Communication Branch, Nanjing Nanrui Information and Communication Technology Co Ltd.: An intelligent completion method and device for missing data in the power grid. CN202211343491.0, 10 January 2023

5. Zhou, F., Jin, L., Dong, J.: Research review of convolutional neural networks. *J. Comput. Sci.* **40**(06), 1229–1251 (2017)
6. Wang, X., Han, Y., Leung, V.C.M.: Convergence of edge computing and deep learning. A comprehensive survey (2020). <https://doi.org/10.1109/COMST.2970550>
7. Li, Y., Hao, Z., Lei, H.: Research review of convolutional neural networks. *Comput. Appl.* **36**(09), 2508–2515+2565 (2016)
8. Long, X.: PP-YOLO: an effective and efficient implementation of object detector (2020). <https://doi.org/10.48550/arXiv.2007.12099>
9. Huang, X.: PP-YOLOv2: a practical object detector (2021). <https://doi.org/10.48550/arXiv.2104.10419>
10. Lema, D.G., Usamentiaga, R., García, D.F.: Quantitative comparison and performance evaluation of deep learning-based object detection models on edge computing devices (2024). <https://doi.org/10.1016/j.vlsi.2023.102127>
11. Xu, S.: PP YOLOE: an evolved version of YOLO (2022). <https://doi.org/10.48550/arXiv.2203.16250>
12. Wang, F., Wang, Q., Chen, J., et al.: Improved faster R-CNN object detection algorithm based on attention mechanism and Soft-NMS. *Prog. Laser Optoelectron.* **58**(24), 405–416 (2021)
13. Zhu, X., Yu, X.: Research on wildlife target detection based on YOLOv5-CA algorithm. *Inf. Technol. Inf.* **267**(06), 32–35 (2022)
14. Li, B., Huang, H., Liu, J., et al.: A lightweight detection model for underwater multi-scale targets in synthetic aperture sonar images based on improved SSD. *J. Electron. Inf. Technol.* **43**(10), 2854–2862 (2022)
15. Li, X., Niu, H.: Facial expression recognition based on VGG-NET feature fusion. *Comput. Eng. Sci.* **42**(03), 500–509 (2020)
16. Luo, X., Liu, Y., Chu, G., et al.: Based on improved YOLOv5 drone image object detection algorithm. *Radio Eng.*, 1–10 (2023)
17. <http://kns.cnki.net/kcms/detail/13.1097.TN.20230411.1311.006.html>



A Joint Optimization Algorithm for Computing Resource Allocation, UAV Trajectory, and Task Offloading in Remote Regional VANETs

Geng Chen^(✉), Yiming Zhang, and Qingtian Zeng

Shandong University of Science and Technology, Qingdao 266590, China
{gengchen,qtzeng}@sdust.edu.cn

Abstract. Reinforcement learning (RL) has demonstrated remarkable effectiveness in addressing task offloading decisions within highly complex network environments. Particularly multi-agent reinforcement learning (MARL), new methodologies have emerged to tackle real-world issues involving both cooperation and competition. We focus on the challenge of task offloading for vehicular users operating in remote areas. To mitigate the scarcity of computational resources resulted by the impracticality of deploying traditional infrastructure like Base Stations (BSs) and roadside units (RSUs) in remote area, we first introduce the use of Unmanned Aerial Vehicles (UAVs) and Low Earth Orbit (LEO) satellites to provide edge service for vehicular users. Furthermore, we put forward an optimization algorithm for allocating computational resource, UAV trajectory optimization, and task offloading based on Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) by making real-time strategies to achieve minimal task completing delay and energy consumption. Finally, we conduct simulation experiments to prove proposed algorithm. The experimental results proved our designed algorithm can reduce task completion delay by 44.4% and task processing energy consumption by 46.8% compared to the MADDPG algorithm.

Keywords: MATD3 · Task offloading · Resource allocation · UAV trajectory · Remote regional VANETs

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61701284, 52374221, the Natural Science Foundation of Shandong Province of China under Grant No. ZR2022MF226, ZR2022MF288 and ZR2023MF097, the Talented Young Teachers Training Program of Shandong University of Science and Technology under Grant No. BJ20221101, the Innovative Research Foundation of Qingdao under Grant No. 19-6-2-1-cg, the Elite Plan Project of Shandong University of Science and Technology under Grant No. skr21-3-B-048, the Taishan Scholar Program of Shandong Province under Grant No. ts20190936.

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2025
H. Jin (Ed.): IAIC 2024, CCIS 2452, pp. 93–105, 2025.
https://doi.org/10.1007/978-981-96-7191-5_8

1 Introduction

Nowadays, Vehicular Ad Hoc Networks(VANETs) are evolving increasingly rapidly. Nevertheless, for VANETs in remote areas, traditional communication methods are frequently unable to meet the unexpected task offloading requirements of vehicle users due to limitations of local infrastructure and geographical challenges [3, 8, 9]. Under such circumstances, introducing UAV and satellite technologies to address the task offloading issue is crucial for realizing the efficient and stable operation of vehicular networks in remote regions [10, 16].

The investigations of task offloading strategies for users under remote regions has become a new hotspot because of the weak network infrastructure and limited computational resource, as well as the high cost of building and maintaining network and computational infrastructure in remote regions. Literature [14] utilized C-BONA heterogeneous power IoT with air-ground integration to alleviate the scarcity of computational resources for IoT devices in remote regions. Furthermore, literature [5] introduced UAVs to enhance the system performance.

Due to limited energy supply in remote regions, the energy consumption of vehicles should be considered. Literature [4] minimized the system energy consumption for the vehicle user task offloading policy uncertainty problem. Further, literature [2] investigated the vehicle user queue task offloading problem in remote region vehicular networking under time-varying channel conditions in continuous time.

Multi-agent deep reinforcement learning (MADRL) is the solution to solve curse of dimensionality problem. Literature [7] considered the spatio-temporal heterogeneity of computational resources and proposed a distributed task offloading algorithm based on MADRL to improve the overall network computational resource utilization. Literature [6] utilized MADRL to optimize the resource allocation scheme and task offloading decision-making strategy, which improves the average utility under this network.

Therefore, to address the shortage of vehicular computational resources and the energy shortage problem in VANETs in remote regions, we design the following strategies. i) Introducing UAVs and LEOs to cope with the shortage of vehicular computational resources. ii) Applying a multi-intelligence deep reinforcement learning methodology, we propose the algorithm for computational resource allocation, UAV trajectory optimization, and task offloading based on MATD3 (MACUT). The energy consumption and task completing delay are minimized by obtaining optimal resource allocation strategy, UAV trajectory optimization strategy and task offloading strategy. Specifically, the contributions are as follows:

- We propose MACUT algorithm to obtain the optimal resource allocation, UAV trajectory optimization, and task offloading policies to minimize energy consumption and task completing delay of vehicular users in VANETs in remote area.
- We put forward an innovative approach to jointly and collaboratively train UAVs and vehicles as agents at the same time. By defining a specific action

space and reward mechanism, the two are motivated to realize efficient collaboration during the training process. Meanwhile, we utilize the characters of MA-DRL which is able to solve the curse of dimensionality problem to improve the performance of whole system.

- We perform a set of experiments about algorithm comparison, hyperparameter comparison, and the effect of varied task size on system performance. The experimental results show MACUT performs better in decreasing average task completing delay and energy.

2 System Model

We consider a remote area scenario. In this scenario, communication devices such as BSs and RSUs are difficult to set up. In the area without BS coverage, assuming that there are vehicles traveling and relatively sparsely distributed, the vehicles generate a fixed number of tasks every t periods, and the tasks are too big to be processed by vehicles themselves due to their own limited computational resources.

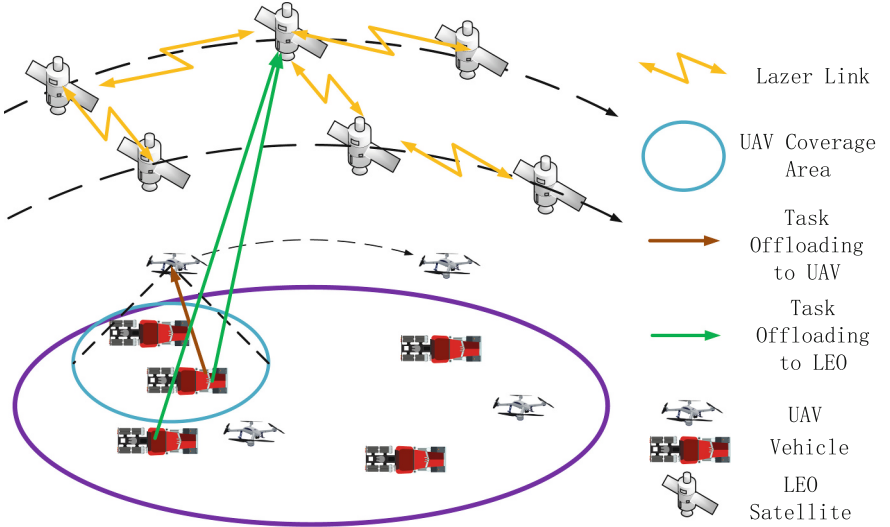


Fig. 1. System Model.

To solve this problem, we consider the introduction of UAVs and LEOs to provide services to vehicles. As shown in Fig. 1, our considered scenario consists of a ground part, an air part and a space part. The ground part consists of large vehicles with limited computational resources, which are represented by the set $V = \{v_1, \dots, v_m, \dots, v_M\}$, the vehicles' position coordinates are denoted

as $b_m(t) = \{x_m^v(t), y_m^v(t)\}$. The vehicles drive at a uniform speed V_v , the driving direction can be denoted by $d_m(t)$ which meets $|d_m(t)| = 1$. The air part consists of J UAVs equipped with edge servers. The space part consists of a LEO satellite and its four neighboring satellites, which are assumed to completely cover all ground vehicles, and vehicles can directly offload the generated tasks to LEO for processing. Thus, there are a total of $J + 1$ edge servers, denoted by the set $\kappa = \{k_1, k_2, \dots, k_j, \dots, k_J, k_{J+1}\}$, where k_{J+1} denotes LEO and $k_1, \dots, k_j, \dots, k_J$ denote UAVs. UAVs can provide high transfer rate for data transferring and sufficient computational resources, but are intermittently available for ground vehicles due to their mobility. We assume that UAVs are flying at a constant speed V_u at an altitude H_u , so we can assume constant power consumption of the UAV. Thus, we primarily focus sight on the power consumption issues arising during task offloading and computation. UAVs' horizontal coordinates are represented as $g_j(t) = \{x_j^k(t), y_j^k(t)\}$. The direction of flight of a UAV is denoted by $d_j(t)$ and $|d_j(t)| = 1$. Although LEO offers relatively small transmission rates, it has more powerful computational resources and, because of their wide coverage, are always available for ground vehicles for relatively long periods of time.

We adopt time slot model. There are T time slots we have set, and each time slot's duration length is τ . Time slots is set as $\dot{T} = \{1, \dots, t, \dots, T\}$. In this scenario, LEO, vehicle and UAV positions do not change in the hourly time slot t , and are dynamically changing in the large time slot. The UAV position and flight direction relationship is

$$\sqrt{(x_j^k(t+1) - x_j^k(t))^2 + (y_j^k(t+1) - y_j^k(t))^2} = V_u \tau d_j(t) \quad (1)$$

similarly, the relationship of ground vehicle position and driving direction is given by

$$b_m(t+1) - b_m(t) = V_b \tau d_m(t) \quad (2)$$

In each time slot, the vehicle randomly generates tasks, and we assume the tasks can be split proportionally. When tasks are generated by vehicles, they can be computed in vehicular users' side or offloaded from vehicles to LEO or UAV for computing.

2.1 Communication Model

When the task is offloaded from v_m to k_j , we consider two different communication models, ground to UAV communication model and ground to LEO communication model.

Ground-UAV Communication Model: The path loss between vehicle v_m and server k_j can be obtained from the following equation.

$$L_{m,j,t} = 20 \lg \left(\frac{4\pi f_c \sqrt{H_u^2 + \|g_j(t) - b_m(t)\|^2}}{c} \right) + P_{m,j,t}^{LoS} \eta_{m,j,t}^{LoS} + (1 - P_{m,j,t}^{LoS}) \eta_{m,j,t}^{NLoS} \quad (3)$$

where $\eta_{m,j,t}^{LoS}$ and $\eta_{m,j,t}^{NLoS}$ are additional losses for line-of-sight (LoS) and non-line-of-sight (NLoS) links, respectively [1]. f_c is carrier frequency and c is light speed. $P_{m,j,t}^{LoS}$ denotes LoS probability of the ground-UAV uplink. It can be obtained from the following equation

$$P_{m,j,t}^{LoS} = \frac{1}{1 + x_1 \exp \left\{ -x_2 [\arctan(\frac{H_u}{\|g_j(t) - b_m(t)\|}) - x_1] \right\}} \quad (4)$$

x_1 , x_2 , $\eta_{m,j,t}^{LoS}$ and $\eta_{m,j,t}^{NLoS}$ are determined by environment states [12]. Therefore, the ground-UAV uplink transmission rate can be calculated as

$$R_{i,j}(t) = B_{i,j} \log_2 \left(1 + \frac{o_{m,j}(t) P_{TX} 10^{-L_{m,j,t}/10}}{\sigma^2} \right) \quad (5)$$

where $B_{i,j}$, P_{TX} and σ^2 denote the uplink channel bandwidth, v_m transmission power, and noise power.

Ground-LEO Communication Model: LEO is an edge server that can execute tasks from v_m . We assumed that the ground-LEO transmission rate is fixed and less than the ground-UAV transmission rate.

2.2 Task Model

Tasks are modeled as a two-dimensional tuple $\varphi_m(t) = \{A_m(t), U_m(t)\}$, among which, $A_m(t)$ is task size and $U_m(t)$ is task tolerance delay. $A_m(t)$ can be divided into two separate parts $A_m^l(t)$ and $A_m^{off}(t)$, where $A_m^l(t)$ is executed in vehicle locally and $A_m^{off}(t)$ is offloaded to edge server for performing. Therefore, the task partitioning at v_m is

$$\begin{cases} A_m^l(t) = \varpi^l A_m(t) \\ A_m^{off}(t) = \varpi^{off} A_m(t) \\ \varpi^l + \varpi^{off} = 1 \end{cases} \quad (6)$$

where ϖ^l and ϖ^{off} denote the split ratio for task $A_m(t)$ to be computed locally and offloaded to an available edge server for computation, respectively. We set the binary variable $s_{m,j} = \{0, 1\}$ to denote the availability of the edge servers with respect to the vehicle v_m . If $s_{m,j} = 1$, edge server k_j will be available with respect to vehicle v_m , otherwise it is not available. We set $o_{m,j} = \{0, 1\}$ to represent the task offloading strategies. If $o_{m,j} = 1$, vehicle v_m will offload the task to k_j , otherwise the task will not be offloaded to k_j to execute.

Local Computing: When $A_m^l(t)$ is computed locally in the vehicle, the task completing delay of $A_m^l(t)$ is

$$D_m^l(t) = \frac{A_m^l(t) \lambda_m}{f_m(t)} \quad (7)$$

where $f_m(t)$ is the allocated CPU frequency of v_m for executing $A_m^l(t)$, and λ_m denotes the computational density (cycles/bit).

Edge Computing: $f_{m,j}(t)$ is setted to denote the allocated CUP frequency at k_j in time slot t for executing $A_m^{off}(t)$. When $A_m^{off}(t)$ is offloaded to UAV, the task completing delay of $A_m^{off}(t)$ is

$$D_m^{off}(t) = \frac{A_m^{off}(t)\lambda_m}{f_{m,j}(t)} + \frac{A_m^{off}(t)}{R_{i,j}(t)} \quad (8)$$

where $\frac{A_m^{off}(t)}{R_{i,j}(t)}$ denotes the transmission delay for $A_m^{off}(t)$ from v_m to UAV.

When $A_m^{off}(t)$ is offloaded to LEO, the completing delay of the task is

$$D_m^{off}(t) = \frac{A_m^{off}(t)\lambda_m}{f_{m,j}(t)} + \frac{A_m^{off}(t)}{R_{g,s}(t)} \quad (9)$$

where $\frac{A_m^{off}(t)}{R_{g,s}(t)}$ denotes the transmission delay for $A_m^{off}(t)$ from v_m to LEO.

Thus, the completing delay of task $A_m^{off}(t)$ is

$$D(t) = \max\{D_m^l(t), D_m^{off}(t)\} \quad (10)$$

2.3 Energy Consumption Model

The energy consumption of v_m includes: energy consumption for tasks computing at v_m side $E_m^l(t)$ and energy consumption for tasks offloading $E_m^{off}(t)$. Among which, $E_m^l(t)$ is defined as

$$E_m^l(t) = \nu[f_m(t)]^3 \frac{\lambda_m A_m^l(t)}{f_m(t)} \quad (11)$$

where ν is the computation power parameter. When $o_{m,j}$ is given, $E_m^{off}(t)$ is

$$E_m^{off}(t) = \sum_{j=J+1} o_{m,j}(t) p_{TX} \frac{Q_m^{off}(t)}{R_{g,s}} + \sum_{k_j \in \kappa \setminus k_{J+1}} o_{m,j}(t) p_{TX} \frac{Q_m^{off}(t)}{R_{m,j}(t)} \quad (12)$$

where former denotes the energy consumption for task offloading to ground-LEO link. Latter denotes ground-UAV link. Thus, in the t th time slot, the overall energy consumption of v_m is

$$E_m(t) = E_m^l(t) + E_m^{off}(t) \quad (13)$$

3 Problem Formulation

We aim to minimize average tasks completing delay and average vehicular energy consumption in remote region VANETs through optimizing computational resource allocation, UAV trajectory and task offloading strategies. This problem can be described as

$$\begin{aligned}
P : & \min_{A^l(t), A^{off}(t), g(t), f(t), o(t)} \sum_{v_m \in V} E_m(t) - [U_m(t) - D_m(t)] \\
s.t. & C1 : o_{m,j} \in \{0, 1\} \quad \forall v_m \in V, \forall k_j \in \kappa, \forall t \in \\
& C2 : \sum_{k_j \in \kappa} o_{m,j}(t) = 1 \quad \forall v_m \in V, \forall t \in \\
& C3: Eq.(6) \\
& C4: f_m(t) \leq f_{m,\max}(t) \quad \forall v_m \in V, \forall t \in \\
& C5 : \sum_{v_m \in V} f_{m,j}(t) \leq f_{j,\max}(t) \quad \forall k_j \in \kappa, \forall t \in \tag{14} \\
& C6: D_m(t) \leq U_m(t) \\
& C7 : g_j(0) = g_j^0 \quad \forall k_j \in \kappa, j \in [1, J] \\
& C8 : sa_d \leq g_j(t) \leq sa_u \quad \forall k_j \in \kappa, j \in [1, J] \\
& C9 : ||g_j(t+1) - g_j(t)|| \leq \varepsilon \quad \forall k_j \in \kappa, j \in [1, J] \\
& C10 : (1) \quad \forall k_j \in \kappa, j \in [1, J+1] \\
& C11 : (2) \quad \forall v_m \in V
\end{aligned}$$

$A^l(t) = (A_m^l(t) : v_m \in V)$ and $A^{off}(t) = (A_m^{off}(t) : v_m \in V)$ denote the tasks executed in vehicle and offloaded to edge servers. $g(t) = (g_j(t) : k_j \in \kappa)$ denotes the UAV trajectory optimization strategy. $f(t) = (f_m(t), f_{m,j}(t) : v_m \in V, k_j \in \kappa)$ denotes the resource allocation strategy. $o(t) = (o_{m,j}(t) : v_m \in V, k_j \in \kappa)$ denotes the task offloading strategy. $C1$ and $C2$ denote that for v_m , its generated tasks can only be offloaded to at most one edge server. $C3$ denotes the constraints on task splitting. $C4$ and $C5$ denote the constraints of vehicle and server computational resource allocation. $f_{m,\max}(t)$ and $f_{j,\max}(t)$ denote the maximum available computing resources at v_m and k_j . $C6$ is task completing delay constraint. $C7$ is initial position of k_j . $C8$ is k_j service region, where sa_d and sa_u denote the upper and lower bounds of the region. $C9$ is safe distance constraint between UAVs. ε is minimum distance between UAVs. $C10$ denotes UAV position and flight direction constraints. By exploring and obtaining the optimal flight direction strategies, UAV can get the optimize its trajectory and provide services for more vehicle users to achieve the minimal energy consumption and task completing delay. $C11$ denotes vehicle position and driving direction constraints.

4 An Optimization Algorithm for Computational Resource Allocation, UAV Trajectory Optimization, and Task Offloading Based on MATD3

During this section, we put forward a MACUT algorithm for remote region VANETs. MACUT is a MADRL algorithm, which the agents are trained centralized and executed distributed.

Computing resource allocation, UAV trajectory optimization, and task offloading problems in this scenario are modeled as a markov decision process (MDP), which can be defined $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R})$. $\mathcal{S}$ is the ensemble of environmental state variables, $\mathcal{O}$ is the ensemble of observed states for each agent, $\mathcal{A}$ denotes the ensemble of actions for each agent, and $\mathcal{R}$ is reward function for agent.

The computational resource allocation, UAV trajectory optimization, and task offloading problems are modeled as follows.

Environmental State: The environmental state variables include the ground vehicle's position $b_m(t)$, the UAV's position $g_j(t)$, the task information, the computation task load in vehicle side $A_m^l(t)$, the task processing load on server side $Y_{m,j}(t)$, the maximum computing capability of vehicle $f_{m,max}$, the maximum computing capability of server $f_{j,max}$, the computation resource allocation strategy at the vehicle side f_m , and resource allocation strategy for UAV f_j .

$$\mathcal{S}(t) = \{b_m(t), g_j(t), \varphi_m(t), A_m^l(t), Y_{m,j}(t), f_{m,max}, f_{j,max}, f_m, f_j\} \quad (15)$$

Observation: We set the vehicle and UVA as agent respectively. For vehicle, the states it can observe include the vehicle's own position $b_m(t)$, the vehicle's generation task information $\varphi_m(t)$, the vehicle's local computing task size $A_m^l(t)$, the vehicle's offloading task size $A_m^{off}(t)$, the vehicle's maximum computation capacity $f_{m,max}$, and the vehicle's computation resource allocation strategy f_m .

$$\mathcal{O}_m(t) = \{b_m(t), \varphi_m(t), A_m^l(t), A_m^{off}(t), f_{m,max}, f_m\} \quad (16)$$

For UAV, the states that it can observe include vehicle position $b_m(t)$, UAV position $g_j(t)$, UAV computing task size $Y_{m,j}$, UAV computing resource allocation strategy f_j , and UAV maximum computing capacity $f_{j,max}$.

$$\mathcal{O}_j(t) = \{b_m(t), g_j(t), Y_{m,j}(t), f_j, f_{j,max}\} \quad (17)$$

Action: During the training process, the vehicle makes the action $\mathcal{A}_m(t)$ which includes task offloading ratio ω^{off} , task offloading strategy $o_{m,j}(t)$ and vehicle side computing resource allocation strategy f_m , and the UAV makes the action $\mathcal{A}_j(t)$ which includes flight direction $g_j(t)$ and computing resource allocation f_j .

$$\mathcal{A}_m(t) = \{\omega^{off}, o_{m,j}(t), f_m\} \quad (18)$$

$$\mathcal{A}_j(t) = \{g_j(t), f_j\} \quad (19)$$

Reward: P is a minimization problem that improves the overall system performance by minimizing the energy consumption ratio and the difference between task tolerance delay and task completing delay, so we set reward functions of the vehicle and the UAV to $\Psi_m(t)$ and $\Psi_j(t)$, respectively.

$$\Psi_m(t) = E_m(t) - [U_m(t) - D_m(t)] \quad (20)$$

$$\Psi_j(t) = \sum_{v_m \in V} E_m(t) - [U_m(t) - D_m(t)] \quad (21)$$

5 Simulation Results

During this section, we perform simulation and show the validity of our MACUT. Experiment results show the advantage of our algorithm in improving the reward, decreasing average vehicle energy consumption and average tasks completing delay. We compare MACUT with multi-agent deep deterministic policy gradient (MADDPG) [11]. Furthermore, the learning hyperparameters in MACUT are studied. Finally, we explore how task size influence the system performance.

5.1 Simulation Parameters

Table 1. Simulation parameters.

Parameter	Value	Parameter	Value
Vehicle number	10	The number of UAVs	2
Task size	1 Mb	Task tolerance delay	[70,80] ms
Duration of a time slot τ	100 ms	Transmission power P_{TX}	23 dBm
Computational density λ_m	1000 cycles/bit	Bandwidth $B_{i,j}$	10 MHz
Power parameter ν	$5 \times 10^{-26} J/Hz^3/s$	Path loss $\eta_{m,j,t}^{LoS}$ and $\eta_{m,j,t}^{NLoS}$	0.1, 21
x_1 and x_2	4.88, 0.43	Noise power σ^2	-114 dBm

During this part, we provide simulation results to show our proposed MACUT's performance. The simulation is implemented in the remote regional VANETs where the length and width is 2000 m. The vehicles drive with a uniform speed 25 km/h considering the poor road condition [13]. We put 2 UAVs which fly in the scenario. For simplifying the simulation, ensuring the reproducibility of experiments and focusing on our investigation object, we assume an ideal experimental condition in which the UAVs fly at a uniform speed and have enough energy as edge servers [15]. We set the decentralized actor and centralized critic of the MACUT scheme are both designed as a fully connected network with 2 hidden layers with 128 and 64 neurons respectively. Other simulation parameters are summarized in Table 1.

5.2 Results Analysis

The Comparison of Hyperparameter: MACUT is simulated at different learning rates (LRs) to explore the optimal LRs, which can be observed in Fig. 2. Our goal is to minimize the reward function, so lower reward denotes better performance. The algorithm converges fastest when LR is set to 0.01, but the reward convergence value is higher. The high LR will make algorithm be too quickly biased towards utilizing known strategies without sufficiently exploring other possibilities in the environment, leading the model to fall into suboptimal strategies and receive low rewards. The algorithm converges slower and the reward value after convergence is higher when LR is set to 0.0001, so the best LR is 0.001.

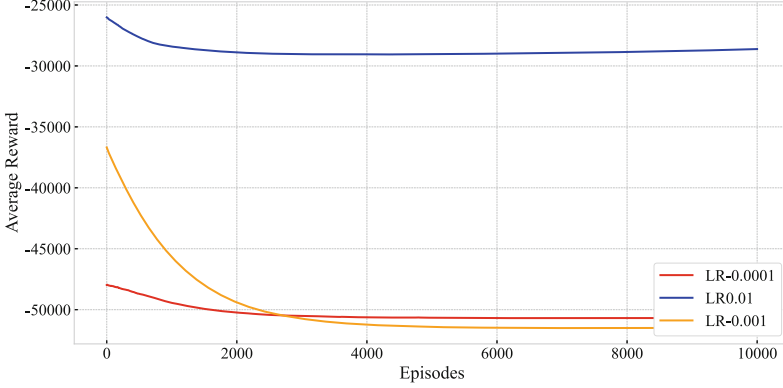


Fig. 2. The Performance Versus Different Learning Rate.

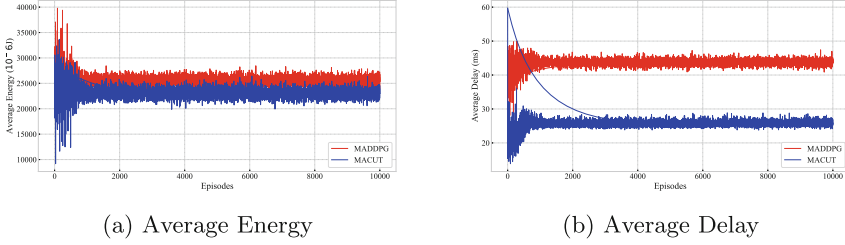


Fig. 3. The Performance Comparison of Different Algorithms.

The Performance Comparison of Different Algorithms: Considering fairness of the simulation process, we have set the same reinforcement learning parameters in different algorithms, including learning rate 0.001, discount factor 0.95, batch size 256, and soft update coefficient 0.01.

From Fig. 3(a) and Fig. 3(b), we can observe our proposed MACUT algorithm performs better in decreasing task completing delay and energy consumption. Compared with MDDPG, MACUT can reach lower target Q-value estimation error, which enables the agents to avoid local optimum. Therefore, MACUT has a better performance. MACUT has reduced 44.4% average task completing delay and reduced 46.8% in average task completing energy consumption compared with MDDPG.

The Performance Comparison with Different Task Size: Furthermore, we explore how different task size influences system performance. From Fig. 4(a) and Fig. 4(b), we can observe as task size increases, vehicle's task completing delay increases and even exceeds its tolerance delay, while other environment variables remain constant. Similarly, vehicle's task completing energy consumption also increases as the task size increases, which shows that the scenario we designed is compatible with reality and reflects the reasonableness of our

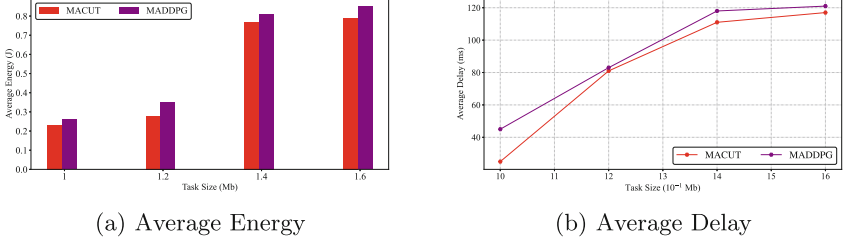


Fig. 4. The Performance Comparison with Different Task Size.

simulation parameter selection. When the task size exceeds 1.2Mb, we can find that the variations of both task completion delay and energy consumption tend to moderate, which shows good robustness of our proposed MACUT facing the offloading computational demands of large tasks. Under the same conditions, as the task size increases, our proposed MACUT performs still better than MADDPG, which indicates that our designed algorithm is scalable and still has superior performance under different environmental conditions.

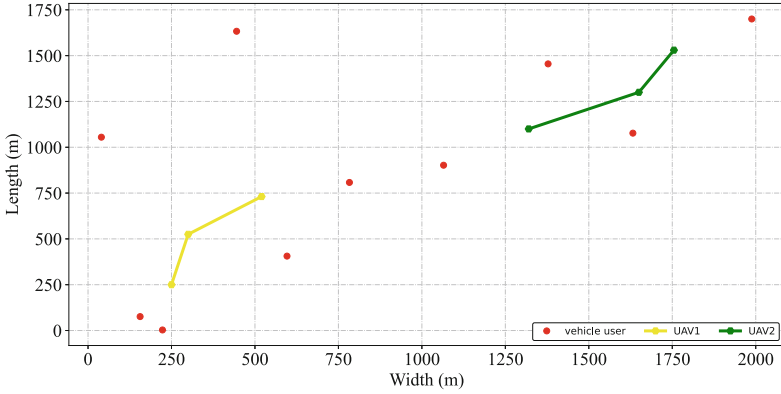


Fig. 5. UAV Trajectory.

UAV Trajectory Optimization. Figure 5 is the trajectory of UAV movement under large time slot T . It can be observed that the UAV flies to the dense area of the vehicle users to provide edge services for more vehicle users, which further shows that the algorithm we designed is effective and reasonable. The limited computational resources on the vehicle side and LEO side are mitigated by optimizing the UAVs trajectory so that the UAVs can provide edge services to as many users as possible at the same time. Meanwhile, the result also makes it feasible for the vehicle and the UAV to train together at the same time as

agents. Although the actions and rewards of UAV and vehicle are different, the UAV can enable the user to get higher rewards by performing the flying actions, and the higher the reward increases the UAV rewards, so the two can enhance system performance through collaborative training.

6 Conclusion

We first introduce UAV and LEO to alleviate the shortage of computing resource in remote regional VANETs. Moreover, we propose a MACUT algorithm, which optimizes the computing resource allocation, UAV trajectory optimization and task offloading problems, to reduce energy consumption and delay required by the vehicle to complete the tasks it generated, and improve the system performance. Finally, through experiments, we verify the advanced and scalability of our algorithm through the comparison between different algorithms, hyperparameter comparison and the comparison of environmental parameter. However, the stability of MACUT for whole system and the algorithm's scalability still remains to be further explored. In future, we are going to test the influence of more complex environmental variables on system performance and the applicability of our proposed algorithm in different scenarios such as public safety and smart city solutions.

References

1. Al-Hourani, A., Kandeepan, S., Lardner, S.: Optimal lap altitude for maximum coverage. *IEEE Wirel. Commun. Lett.* **3**(6), 569–572 (2014). <https://doi.org/10.1109/LWC.2014.2342736>
2. Bi, S., Huang, L., Wang, H., Zhang, Y.: Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks. *IEEE Trans. Wireless Commun.* **20**(11), 7519–7537 (2021). <https://doi.org/10.1109/TWC.2021.3085319>
3. Bute, M.S., Fan, P., Liu, G., Abbas, F., Ding, Z.: A collaborative task offloading scheme in vehicular edge computing. In: 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), pp. 1–5 (2021). <https://doi.org/10.1109/VTC2021-Spring51267.2021.9448975>
4. Chen, Y., Ai, B., Niu, Y., Zhang, H., Han, Z.: Energy-constrained computation offloading in space-air-ground integrated networks using distributionally robust optimization. *IEEE Trans. Veh. Technol.* **70**(11), 12113–12125 (2021). <https://doi.org/10.1109/TVT.2021.3116593>
5. Guo, H., Wang, Y., Liu, J., Liu, C.: Multi-UAV cooperative task offloading and resource allocation in 5G advanced and beyond. *IEEE Trans. Wireless Commun.* **23**(1), 347–359 (2024). <https://doi.org/10.1109/TWC.2023.3277801>
6. Hazarika, B., Singh, K., Biswas, S., Mumtaz, S., Li, C.P.: Multi-agent DRL-based task offloading in multiple RIS-aided IOV networks. *IEEE Trans. Veh. Technol.* **73**(1), 1175–1190 (2024). <https://doi.org/10.1109/TVT.2023.3302010>
7. Hou, Y., Wei, Z., Zhang, R., Cheng, X., Yang, L.: Hierarchical task offloading for vehicular fog computing based on multi-agent deep reinforcement learning. *IEEE Trans. Wireless Commun.* **23**(4), 3074–3085 (2024). <https://doi.org/10.1109/TWC.2023.3305321>

8. Kwak, J., Kim, Y., Le, L.B., Chong, S.: Hybrid content caching in 5G wireless networks: cloud versus edge caching. *IEEE Trans. Wireless Commun.* **17**(5), 3030–3045 (2018). <https://doi.org/10.1109/TWC.2018.2805893>
9. Liu, Y., Peng, M., Shou, G., Chen, Y., Chen, S.: Toward edge intelligence: multi-access edge computing for 5g and internet of things. *IEEE Internet Things J.* **7**(8), 6722–6747 (2020). <https://doi.org/10.1109/JIOT.2020.3004500>
10. Liu, Y., Zhang, H., Zhou, H., Long, K., Leung, V.: User association, subchannel and power allocation in space-air-ground integrated vehicular network with delay constraints. *IEEE Trans. Network Sci. Eng.* **10**(3), 1203–1213 (2023). <https://doi.org/10.1109/TNSE.2022.3169635>
11. Maleki, M.R., Mili, M.R., Javan, M.R., Mokari, N., Jorswieck, E.A.: Multi-agent reinforcement learning trajectory design and two-stage resource management in comp UAV VLC networks. *IEEE Trans. Commun.* **70**(11), 7464–7476 (2022). <https://doi.org/10.1109/TCOMM.2022.3211083>
12. Peng, H., Shen, X.: Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks. *IEEE J. Sel. Areas Commun.* **39**(1), 131–141 (2021). <https://doi.org/10.1109/JSAC.2020.3036962>
13. Qin, P., Fu, Y., Tang, G., Zhao, X., Geng, S.: Learning based energy efficient task offloading for vehicular collaborative edge computing. *IEEE Trans. Veh. Technol.* **71**(8), 8398–8413 (2022). <https://doi.org/10.1109/TVT.2022.3171344>
14. Qin, P., Fu, Y., Zhao, X., Wu, K., Liu, J., Wang, M.: Optimal task offloading and resource allocation for C-noma heterogeneous air-ground integrated power internet of things networks. *IEEE Trans. Wireless Commun.* **21**(11), 9276–9292 (2022). <https://doi.org/10.1109/TWC.2022.3175472>
15. Zhang, X., et al.: Energy-efficient computation peer offloading in satellite edge computing networks. *IEEE Trans. Mob. Comput.* **23**(4), 3077–3091 (2024). <https://doi.org/10.1109/TMC.2023.3269801>
16. Zhang, Z., et al.: 6G wireless networks: vision, requirements, architecture, and key technologies. *IEEE Veh. Technol. Mag.* **14**(3), 28–41 (2019). <https://doi.org/10.1109/MVT.2019.2921208>



LSTMR: Spatio-Temporal 3D Multiscale ResNet Model for Cellular Network Traffic Prediction

Geng Chen^(✉), Weiming Wang, and Qingtian Zeng

Shandong University of Science and Technology, Qingdao 266590, China
{gengchen,qtzeng}@sdust.edu.cn

Abstract. Predicting urban cellular network traffic is crucial for urban construction planning and base station construction distribution, but due to a number of complex factors (e.g., dynamic spatial and temporal dependencies, complex spatial dependencies, external environments, etc.), counting the details of these efforts is not easy for current operators. Forecasting for regions is based on historical data to predict future conditions in urban areas. In the previous literature, the more commonly used methods are based on the extraction of temporal features by LSTM and the extraction of local spatial correlation features by local Convolutional Neural Networks (CNN) algorithms for solving region-based problems. On this basis, we take the use of spatio-temporal 3D grouped multiscale ResNet combined with LSTM temporal blocks to achieve the corresponding predictions. ResNet also allows the combination of spatio-temporal features with external factors. Residual Network (ResNet) combines tight and periodic 3DCNN branching with traffic movement trends as well as other external factors to forecast future call-in and call-out traffic for regional cellular network traffic. To assess the effective results of our work, we base it on a real-world cellular network traffic dataset. In contrast to other related current work used for prediction, the experiments demonstrate a reduction in RMSE of 4.7% and 3.3%, respectively, when prediction is performed over a dataset from the Italian city of Milan.

Keywords: Cellular network traffic prediction · Spatiotemporal multiscale three-dimensional contextual neuro network · Residual local network · Periodicity · Closeness and Trend

This work was support by the National Natural Science Foundation of China under Grant No. 61701284, 52374221 the Natural Science Foundation of Shandong Province of China under Grant No. ZR2022MF226, ZR2022MF288 and ZR2023MF097 the Talented Young Teachers Training Program of Shandong University of Science and Technology under Grant No.BJ20221101 the Innovative Research Foundation of Qingdao under Grant No. 19-6-2-1-cg the Elite Plan Project of Shandong University of Science and Technology under Grant No. skr21-3-B-048 the Taishan Scholar Program of Shandong Province under Grant No. ts20190936.

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2025
H. Jin (Ed.): IAIC 2024, CCIS 2452, pp. 106–120, 2025.
https://doi.org/10.1007/978-981-96-7191-5_9

1 Introduction

In the last few years, as the communications network technical knowledge evolved and the cellular communications technical knowledge progressed [6], on the continuous popularity of these mobile devices and AI technologies, the transmission of information augmented by computing devices and smart technologies has become faster, smarter and more controllable [18]. Predict cellular network traffic precisely on the region is crucial for the further laying of large areas of network information technology base stations and the effective allocation of communication network resources, and can help the relevant departments used in the development of the optimal resource layout map. Recently, The gradual popularization of AI technology and deep learning methods has led fellows to increasingly leverage these techniques for predicting visualization data [1]. By utilizing actual data from existing areas, predictions can provide more accurate insights into future scenarios, as each city locality possesses unique characteristics influenced by both nearby and distant regions [3]. The dynamic and real-time nature of traffic context changes in local areas presents significant challenges. Region-based cellular network traffic prediction is essential for the transport department, enabling more effective management of base stations and facilitating optimal resource allocation [9]. Consequently, traffic prediction methods are gaining prominence, as illustrated in Fig. 1. In addition, ResNet enables convolutional neural networks to connect across layers for more in-depth architecture of the network. The methodology was proven in a telecom dataset from Communication dataset of the city on the basis as collected by the telecom italia grand data competition. The dataset consists of two months of continuous data with one hour intervals. We weighed the proffered technique against some of the methods commonly used on prediction work and proved its excellent properties.

The method combines outputs from 3 networks, assigning different weights to each branch, while also integrating outputs from external factors. This approach is versatile, applicable to various datasets, and supports different types of area-based forecasting to derive multiple evaluation metrics for urban regions. We present LSTMR, a novel method that captures the spatio-temporal characteristics of local areas through local 3D CNNs. It accounts for spatial correlations, temporal dependencies, and external influencing factors within the local context. LSTMR shows improved flexibility in area-based predictions, as validated by the Milan dataset. The model can directly capture complex spatial correlations based on data graphical detailing, a new expansion ResNet networks can be grouped in 3D multiscale without pooling operations. The multi-scale architecture through grouping is sufficient to obtain a larger receptive domain, which can appropriately reduce the consumption caused by machine computation and storage.

2 Methodology

In summary, our approach differs significantly from the above research work. Based on some design aspects of this model, we design a new zone-primarily

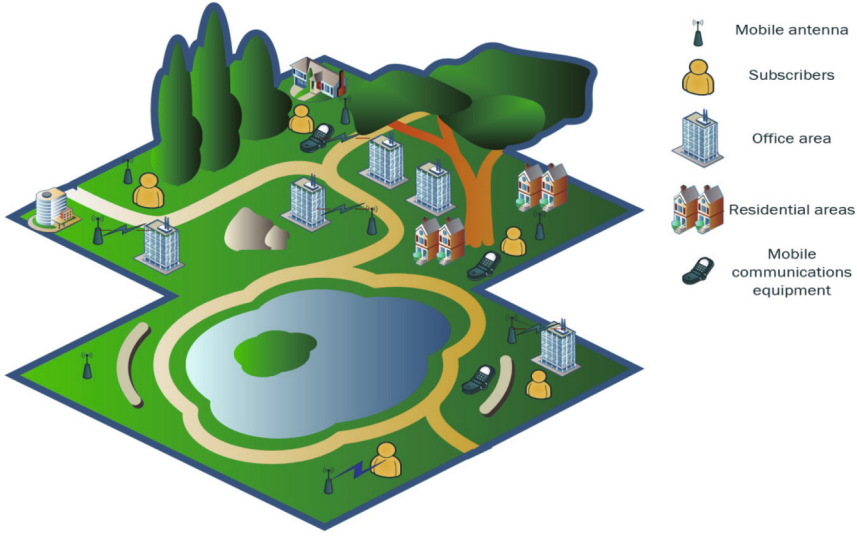


Fig. 1. Realistic cellular network traffic construction layout.

based feature acquisition algorithm and spatio- time correlation between several local zones of a urban. The spatio-time correlation relevant characteristics of Several partial domains are available extracted as well as learnt at the same time, enabling the evolution of statistical data resource mobilisation from a low level to a high level of historical data migration. We will define some definition in this part of the chapter. Based on these notions, we formulate a forecasting scheme based on zones. Based on this Scenario, a number of concepts are given using the Milan dataset as an example. The relevant details are given below (Fig. 2):

Definition 1 (Regions): Word in a sentence, the city of Milan, Italy, based on its geographical location in latitude and longitude, can be divided into a lattice diagram, where permanent features are called zones and each zone can be represented by a set of axes: $C_{jw} = \{c_{1,1}, c_{1,2} \cdots c_{j,w}\}$.

J*W corresponds to row J and column W of the grid. The figure shows localised areas of the entire sub-districts (about 40) of the city of Milan.

Definition 2 (Callin and Callout Volume): We hypothesise to let P be the set the GPS position at time epoch i. The data of cellular traffic can be visualised to some extent in a number of ways, and based on this visualisation process, heat maps can be formed by normalising the intensity of cellular traffic based on the division of its corresponding grey scale values. The call-in call-out for each region j is $n_t^{\text{in},j,w}, n_t^{\text{out},j,w}$ as follows:

$$n_t^{\text{in},j,w} = \sum_{\Gamma_r \in P} \{\beta > 1 | u_{\beta-1} \notin c_{i,j} \wedge u_{\beta} \in c_{i,j}\} \quad (1)$$

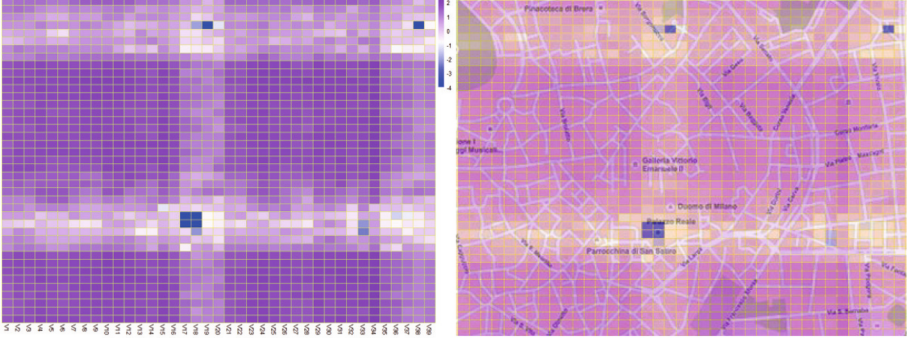


Fig. 2. Call visualisation of cellular network traffic in selected areas of Milan.

$$n_t^{\text{out},j,w} = \sum_{\Gamma_r \in P} \{\beta > 1 | u_{\beta-1} \notin c_{i,j} \wedge u_\beta \in c_{i,j}\} \quad (2)$$

where $\Gamma_r: u_1 \rightarrow u_2 \rightarrow \dots \rightarrow u_{|\Gamma_r|}$ is a trace in Γ_r and u_β is a location-based coordinate. $u_\beta \in c_{i,j}$ denotes that location is within region i,j .

Definition 3 (spatial dependence of the neighbourhood): The scene is shown in Fig. 3, the traffic of communication networks in the neighbourhood interacts within the city. The outbound traffic from nearby areas (let M2, M3, M4) influences the inbound traffic from area M1. Similarly, outbound calls from M4 affect inbound calls from other regions (e.g., M2.) Inbound calls from region M4 affect its own outbound calls.

Definition 4 (Remote Spatial Dependencies): Cellular network traffic in remote areas of the same city may have an impact on each area. For example, the construction of network base stations and the low penetration coverage of mobile telecommunication devices by residents in remote neighbourhoods affects the speed and capacity of the transmission channel for cellular network traffic (e.g., inbound and outbound calls from M1 are affected by M5 and M6).

Definition 5 (External Factors): A number of external factors also influence urban cellular network traffic. For example, during hot summer days, people want to stay indoors or at home, so people may use the cellular network in order to talk and spend time. If the weather is bad, people likewise like to access the internet and talk on holiday.

Scenario 1 (region-based urban prediction): History-based local area cellular network traffic is set to be $\{Y^t | t = 0, 1, 2, \dots, n\}$, time interval as 1,2,3, . . . n, the ultimate goal is to predict Y^{n+1} , which represents the call-in and call-out for each zone in the next zone $n+1$.

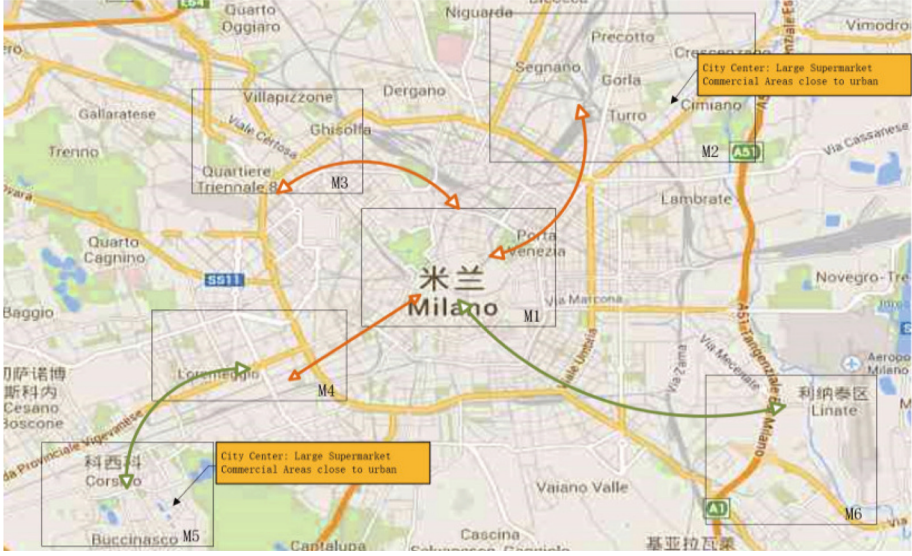


Fig. 3. Dependence of traffic call-in and call-out on closeness and remoteness M1 is the centre of Milan, M5 and M6 are located in the suburbs.

2.1 3DCNN Module Block

This section outlines the urban communication network and the problem of predicting cellular network traffic. We will explore the components of the structure and capabilities of the model that have been offered, beginning with an explanation of how multiple local 3D CNNs can capture temporal correlations among various local regions, enabling accurate predictions of future conditions in urban areas based on historical data [4]. We will also analyze how these multiple temporal correlations influence region-based predictions and propose a method to model them effectively [4]. While local 2D CNNs address the weak correlations between a region and its neighbors-primarily from a spatial perspective-2D CNNs have limitations; they can only extract temporal features using alternative methods, such as LSTM [13]. However, this approach primarily establishes temporal connections at higher hierarchical layers, leaving spatio-time correlation features at lower layers underutilized. In contrast, 3D CNNs leverage 3D convolution and pooling layers to successfully study spatio-time correlation characteristics [7]. The three dimensional involution formula is follows:

$$u_{ji}^{\beta}(x, y, z) = \sum_{m, n, l} Z_i^{\beta-1}(x - m, y - n, z - l) Q_{ji}^{\beta}(m, n, l) \quad (3)$$

$$Z_j^{\beta} = f \left(\sum_i u_{ji}^{\beta} + b_j^{\beta} \right) \quad (4)$$

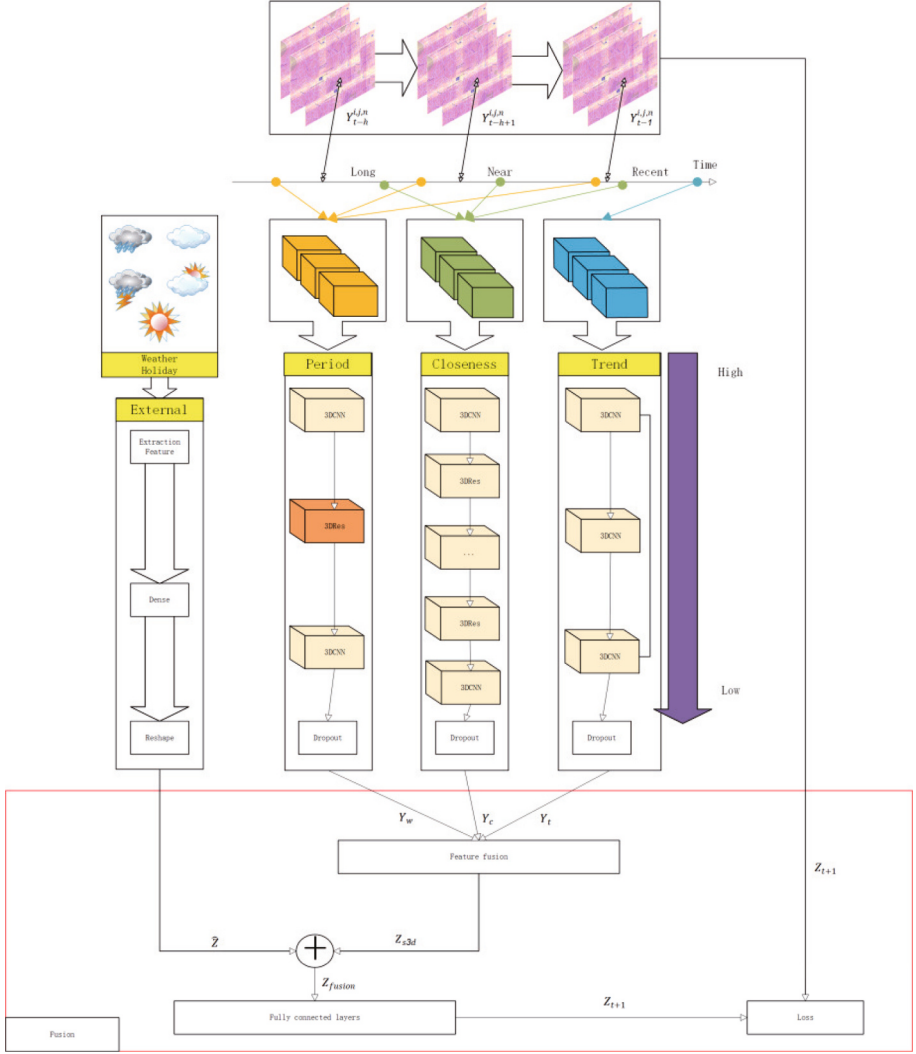


Fig. 4. LSTMR specific structure.

The function f indicates the rectified linearity unitary (ReLU) and b indicates the bi-polar term in the characteristic picture. In the case of three-dimensional CNNs, while low-rank characteristics can be utilised more efficiently, their performance will still suffer due to the weak dependencies between different areas. To overcome this restriction, we introduce a partial three-dimensional CNN method, which solves both the time limitation problem usually faced by partial two-dimensional CNNs and the difficult problem of spatial region correlation in three-dimensional CNNs. This method enhances spatial relevance by creating spatio-time correlation of multiple partial areas [5].

2.2 Local Space Block

A detailed description of our proposed LSTMR framework, which employs a region-based forecasting approach to predict future conditions in urban areas, is presented in this section. The architecture of our suggested method is illustrated as Fig. 4. There are four aspects of our suggested model: partial three CNN, ResNet, heightened feature convergence, and exterior block fusion. We were able to study the spatio-time correlation dependence characteristics of multiple partial zones by LSTMR. The spatial view is characterised by the fact that the surrounding neighbourhood of per geographical boundary is centred as $R_{j,i}$. In Fig. 4, as orange line indicates inbound and outbound calls from a neighbouring region of space by space limit; as dark green streak indicates call outs obey a current zone to a more distant region of space. The characteristics of the time perspective include diverse trends, cycles with close relatedness, as summarised in the table below orange, dark green, and blue, respectively. The heatmap structure is based on a small portion of the dataset taken out as an input, which has a numerical axis of timestamps extracted every hour, and a horizontal axis based on blocks of regions of the Milan dataset, which ultimately come together to form the heatmap. Three-dimensional configuration and three-dimensional aggregation layers are The light blue squares and dark green squares are plotted in the light blue cubes and dark green cubes, separately. In which every branch, the above attribute features are characterised by a unique 3DCNN [11]. LSTMR has the ability to assemble the outputs of the 3 networks - Tendency, Periodicity and Proximity - by assigning distinct priorities to individual branches in a democratic manner. The exterior module takes into account weather conditions, event data, and other relevant influences. We extract key details from these external sources manually and feed them into two fully connected layers for further processing. Eventually, loss is calculated by combining the fusion of information from external modules and spatio-time correlation features. We consider nearby area as the $l \times l$ imagery and the $R_{j,i}$ within a area in centre direct. $R_{j,i}$ denotes the channel for the demand value, and control granularity of space by L-granularity. The boundaries of a geographical constituency, we use the method of vectorial addition metric. Area image vector was denoted like $Y_{t-1}^{J,W} \in R^{h \times l \times l \times c}$, as time span t , individual placements i, j , duration h and frame passage p . Native three-dimensional CNN transforms $Y_{t-1}^{j,i}$ to $Y_{t-1}^{j,i,0}$ as their input and pass them to the N concentration levels. The transformation is specified as below, in terms of the n th level for instance:

$$Y_{t-1}^{j,i,n} = f(Y_{t-1}^{j,i,n-1} * Q_{t-1}^n + b_{t-1}^n) \quad (5)$$

where $*$ denotes three-dimensional convolution operation, f means the optimisation function denoted a function that takes the largest value of the f variable x from 0 to c ; $Y_{t-1}^{1,\dots,N}$, $b_{t-1}^{1,\dots,N}$ denote the n th layer convolution parameters. These two parameters Q_t^n and b_t^n were common to all localities $j, i \in J, I$.

2.3 Resnet Module

Convolutional Neural Networks (CNN) are able to extract different capabilities depending upon the internetwork layer. Distinctive traits provide alternate information. We suppose that the magnitude of the entered value of the information in the file city-wide region is 8×8 , as the kernel of the three-dimensional volume deposition is set to $3 \times 3 \times 3$ [8]. Modelling spatio-time correlation across a range of localities areas of a towns involve a comparatively greater number of volumetric layers. So as so extract feature information from localised prediction data using CNNs, we seek to gain as much depth in the grid as we can. Nevertheless, if the programme is deepened arbitrarily, gradient diffusion or gradient explosion can easily occur. Thus, the resulting profound residual learning enabled convolutional neural networks to connect across layers with a deeper network structure. This architecture can deliver the most advanced results in areas such as image classification, target detection and image segmentation. Based on the aforementioned deliberations, we incorporate the idea of deep residual networks in our model and define the three-dimensional residual network in LSTMR as below: $T = f(x) + x$. With the addition of residual constructs, CNNs extract more traits out of the incoming traffic, notably focussed on learn the spatio-time correlation diagnostic property about multiple localised areas of the urban. Moreover, this would be a superior way to tailor the organisation of the project to output improved outcomes.

2.4 Equalisation Characteristic Integration

For the following reasons, as defined in Sect. 2, local areas are subjected to plural chronological relevance. Use the Milan dataset for instance, the cellular network traffic in the urban area, the data is studied. There is relatively little variation from 5am to 6am. Yet, the change from 6 p.m. onwards about 7 p.m. is noticeable. Three characteristics about cellular network traffic are shown by Fig. 5. The trend, cycle divisions and close branches have varying degrees of impacts on the social area due to the three separate branches spatio-time correlation hallmarks. Above the identity of the estate is learnt from 3DCNN [14]. LSTMR dynamically aggregates the outputs of triple axis [i.e., closeness (Fig. 5.a), period (Fig. 5.b), and trend (Fig. 5.c)] and assigns different weights to different branches with the following equation: $Z_{\text{fusion}} = Y_c \otimes Z_c + Y_w \otimes Z_w + Y_d \otimes Z_d$.

2.5 External Module Integration

It is well known that external modular considerations play an impact on the area-grounded forecasts. As such, accounting for these elements in area-primarily based projections allows for more robust modelling of the future surrounding realities. We output external factors via two wholly-attached tiers. To activate each subfactor in the feature retrieval process, we use a fully linked stratum. Afterwards, we map the signatures using the all-connectivity dimension Z_{ex} , so that it has the same dimension as Z_{s3d} . We combine Z_{s3d} with the output of an

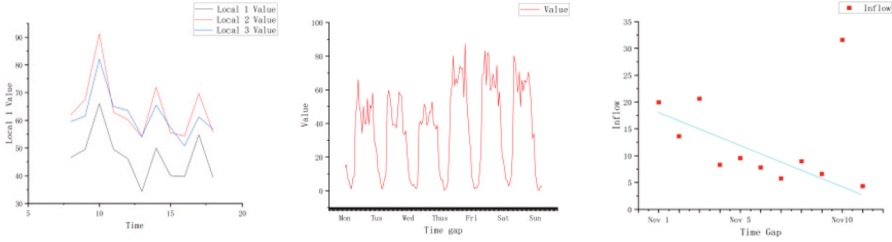


Fig. 5. Temporal correlation on the Milan dataset[(a)closeness (b) period (c) trend].

external block connects layers with the Tanh features. Tanh as a hyperbolic tangent function that will guarantee an output that lies in the range of -1 through 1 during backpropagation learning. The fused output is defined as Equation:

$$\hat{Z} = Z_{s3d} + Z_{ex} \quad (6)$$

2.6 Optimisation Problems and Model Training

In our LSTM, on the basis of Adam's calculation as an optimiser. With input spatio-time correlation information is $Y_c Y_w Y_d$. Feature information is extracted through our model to predict be worth of t moment and convert Translate the problem into a reentry assignment. Adam was selected as the optimiser cause it is possible to calculate the first and second magnitude estimators of the gravity, designing independent adaptive learning rates for different parameters. That optimisation project makes use of the Adam feature to maximise the spatio-time correlation information. The propounded moulds are implemented by means of the following tensor flow.

3 Experiment Session

3.1 Assessment of Milan Data Sets

During our experiments, our proposed LSTM is evaluated from real-world authentic data (i.e., the Milan, Italy data). The substance of this data set are described in detail below. (Data set: Milan data was collected from 1 November 2013 at 0:00 hrs to 1 January 2014 at 23:00 hrs). In this data set, Milan is segmented into 32×32 zones and the temporal interval was fixed at an hour. Inbound and outbound cellular network traffic for this area was obtained in the data set. Regarding the external characteristics, we count them by extracting some sets on the original data set. For example, after determining the timestamp of the data set, some data are extracted and counted based on information about the local weather and possible holidays in Italy, resulting in the so-called external features data set. Firstly, 3D grid images extract from data sets using algorithms. After that, 3D grid image values are normalised to $[0,1]$ using the MinMax method, while the predicted values are evaluated for denormalisation.

An early termination technique was employed to discourage overfitting while all model training. For external factors, continuous features (e.g., weather and temperature) are scaled using the MinMax method, whereas monothermal coding is used to transform discrete features (e.g., holidays, weekdays, and weekends) [8]. Transformations are $Z_{\text{new}} = \frac{Z - Z_{\text{min}}}{Z_{\text{max}} - Z_{\text{min}}}$ Where Z is the constant value of the sensor data; Z_{max} is the largest point of the sample data, Z_{new} is the lowest figure of the collected data of the sample, all data is the mapping value between $[0,1]$. Use of LSTMR, a comparison with other established measures for a well-rounded volume analysis.

3.2 Experimental Setup

A functioning computer terminal with a GPU with the necessary programming facilities and licenses will have to be configured so that the perform experiments are conducted and the performance of the proposed model is appraised. All of the baseline hyperparameters were tuned by lattice seeking in order to obtain optimal results from the practice figures. Ten tests were conducted for each method, giving average results for the models. In the name of assisting to solve overfitting problem, the value of ‘dropout’ is given as 0.25. In addition, prior work was initialise filter weights for all contextual and fully-connected layers. We built a model based on tensor dimensions and Kailas. For the Milan data set, due to its spatial dimension regular dataset is larger than $32*32$, let’s create a three-dimensional volume scale of ever branches to $4*32*32$. The evaluation metrics, Root Mean Square Error (RMSE) [2] and Mean Absolute Error (MAE), which are commonly used in urban cellular network traffic prediction, are used as an elicitation that serves to rate the usefulness of all acquisition policies. These are as in the following terms:

$$MAE = \frac{1}{N} \sum_{u=1}^N |Z^i - Z_{pre}^i| \quad (7)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{u=1}^N (Z^i - Z_{pre}^i)^2} \quad (8)$$

3.3 Method of Baseline and Ablation Experiments

The purpose is to perform a collaborative analysis of our proposed model, we applied several most advanced base-line methods. The precise minutes of the practice at which methods of completion were compared are set out below.

Methods of forecasting historical time series:

HA: Prediction of inbound/outbound calls in each area by aggregated values from previous time intervals based on the Historical Average (HA) algorithm.

ARIMA [17]: Autoregressive trading routines are the most widely adopted form of used time series model forecasting.

Machine acquisition methods:

RAF [12]: Random forest is a complex settlement to categorisation with regression problems which uses integrated practicing method called sacking.

LinUOTD [15]: LinUOTD utilises a straightforward model framework in an effort to refrain from model redesign and a linear regression model to regularise spatio-time correlation diagnostic trait.

XBT [20]: XBT is for constructing and enlarging tables founded on close-tree learning algorithms and sparse perception algorithms that can expand more unknown sources with fewer available alternatives.

CLSTM [13]: CLSTM incorporates a volumetric fabric on top of LSTM. The convolutional layer enables better learning of spatio-temporal associated features.

DSD [10, 16]: DSD makes use of an arboreal synthetic nerve mesh network to predict differences in traffic supply and demand patterns. It should be viewed as an extended MTP model, dealing with supply and demand patterns and exogenous contextual messages (i.e. weather and weekday holiday information).

STDN [21]: STDN combines a local 2DCNN, LSTM and an attentional mechanism means by which aerospatial and spatial information can be accessed in an optimal way.

STG [22]: The methodology can predict the traffic flow gloriously in accordance with the urban traffic circulation layout, and does the prediction task by semantic encoding based on multi-layer sensing.

LSTM3D is proposed to implement repeated three-dimensional CNNs, but there are always shortcomings due to the lack of a neighbourhood perspective.. The dataset achieves the minimum RMSE and MAE in Milan, Italy data-set. Thus, we get a clear glimpse of what we are proposing LSTM3D-Res frameworks obtain as highlighted in font in Table 1, the accuracy is much higher Inbound and outbound traffic counting methods were considered on the Milan, Italy. Table 1 presents detailed results of inbound as well as outbound performance evaluation in Milan, Italy, showing comparison results of our proposed LSTM3D with existing strategies in terms of RMSE and MAE.

For a fair performance comparison, we evaluated these ways in a Nvidia 4060. Among our comparisons between LSTM3D and LSTM3D-Res, we can see that the runtime of LSTM3D is less than that of LSTM3D-Res. The major reason for this is due towards LSTM3D's cross-layer linking manipulations, and our proposed LSTM3D-Res for LSTM3D in significantly improved runtime benchmarking. Our resulting LSTM3D-Res performance plots exhibit the RMSE and MAE after training verification against the Milan, Italy database set. LSTM3D-CT and LSTM3D-CTP further enhance upon performance by adding trend and period subdivisions. LSTM3D-Res reduces the RMSE and MAE values additionally compared to other

benchmarking tests. LSTM3D proved that taking into consideration the several factors contributes to the accuracy of the values particular to the city area. We demo the LSTM3D performance by displaying RMSE and MAE for Milan, Italy data set.

The treatments as follows:

LSTM3D-W: We merely regard spatio-temporal features closer to the branch.

LSTM3D-CW: Our framework uses only proximity branches and trend branches.

LSTM3D-CT: With this approach, our framework solely works with proximity and trend branches.

LSTM3D-CWT: In this kind of estimation, our framework takes advantage of closeness, trend and cycle branching.

LSTM3D-CTP: Our framework uses proximity, trend and period branches.

LSTM3D: Within this structure, our model uses closeness, trend, cycle and external branching.

LSTM3D-ResNet-CTP: At this kind of mapper, our figure utilises closeness, trend, and period branches while performing regression based on residual networks.

LSTM3D: Our model makes use of tightness, trends, cycles and external branches and combines them with residual network work.

LSTM3D further minimises RMSE and MAE values relative to other benchmarks. Compared to LSTM3D, LSTM3D-Res consumes some runtime but is still within acceptable limits, while LSTM3D-Res performs comparatively well. Consequences of ablation experiments and predicted results correspond to Figs. 6 and 7, respectively.

Table 1. Milan baseline comparison.

Method	Inflow	Inflow	Outflow	Outflow
	RMSE	MAE	RMSE	MAE
HA	50.44	25.63%	50.56	40.21%
ARIMA	22.56	12.87%	22.98	22.24%
RAF	22.03	12.11%	21.73	12.33%
LinUOTD	21.79	20.56%	21.54	20.43%
XBT	19.62	11.03%	18.45	17.85%
CLSTM	19.34	18.75%	19.91	19.28%
DSD	17.98	17.53%	18.37	18.06%
STDN	16.77	15.17%	17.52	15.73%
STG	14.43	8.79%	14.16	8.87%
LSTM3D	14.05	8.67%	13.91	8.73%

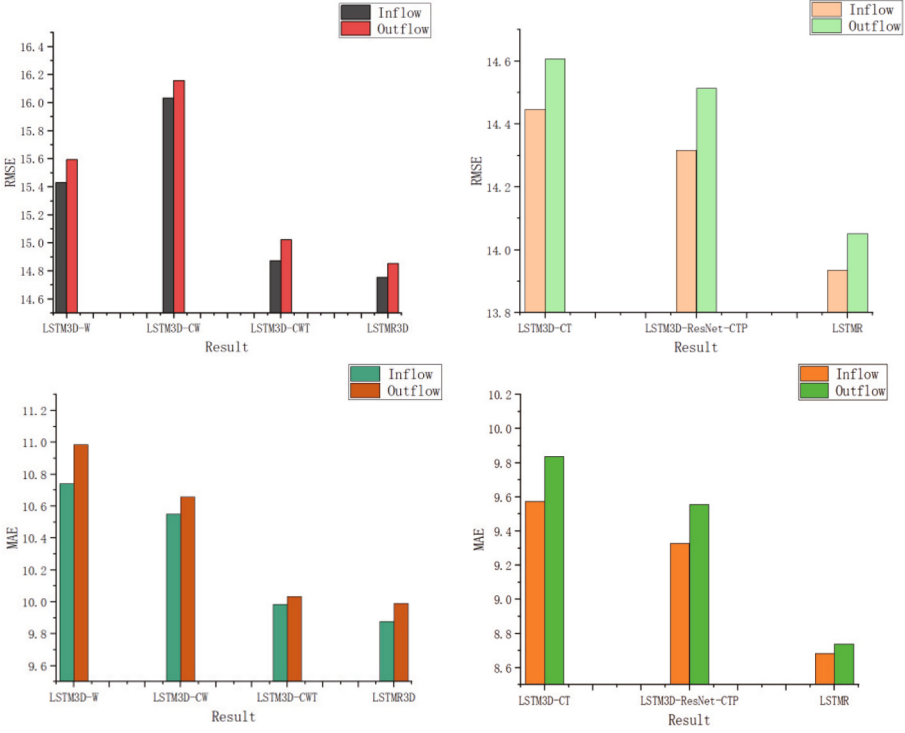


Fig. 6. LSTM metrics performance for different feature modules Milan dataset.

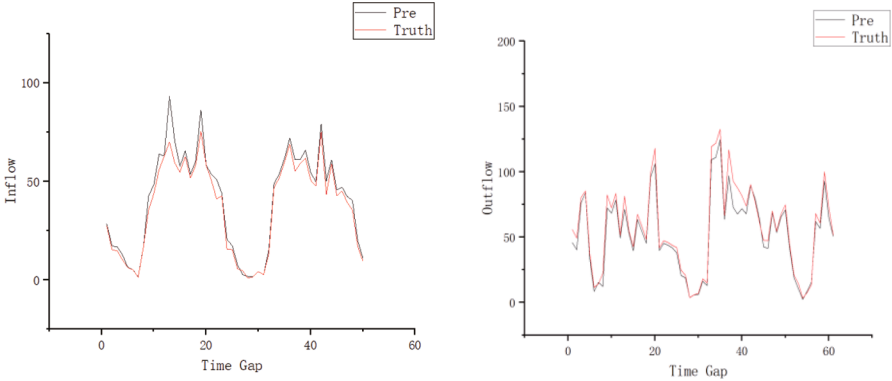


Fig. 7. Comparison of LSTM's prediction of cellular network traffic over a two-day.

4 Conclusion

Based on utilising three-dimensional convolutional techniques in computer visualisation, we propose 3DCNN-based extraction of time-space correlations

traffic prediction in Milan cellular networks. In addition, inspired by the use of initial Resnet network in ImageNet classification, and in combination with the features of most of today's models used for traffic flow prediction, we obtain good anticipation results. In this article, we are proposing a spatio-temporal correlation feature acquisition method that considers the relevance of space-time (in relativity) centred features in a local area and the influence of external block factors based on regional prediction. A new regime of zone-based energy harvesting and end-to-end multiplicative spatio-temporal correlation-dependent learning structures are engineered in our suggested model. In future work, our research direction will further explore better integration of learning spatio-time correlation features in the framework, add more semantic information to a fram for learning through graphics, and try to use a more efficient loss function in the temporal information.

References

1. Buzachis, A., Celesti, A., Galletta, A., Fazio, M., Fortino, G., Villari, M.: A multi-agent autonomous intersection management (MA-AIM) system for smart cities leveraging edge-of-things and blockchain. *Inf. Sci.* **522**, 148–163 (2020)
2. Chai, T., Draxler, R.R.: Root mean square error (RMSE) or mean absolute error (MAE)?-arguments against avoiding RMSE in the literature. *Geosci. Model Dev.* **7**(3), 1247–1250 (2014)
3. Chen, C., et al.: Exploiting spatio-temporal correlations with multiple 3D convolutional neural networks for citywide vehicle flow prediction. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 893–898. IEEE (2018)
4. Chen, Y., Zou, X., Li, K., Li, K., Yang, X., Chen, C.: Multiple local 3D CNNs for region-based prediction in smart cities. *Inf. Sci.* **542**, 476–491 (2021)
5. Chen, Y., Zou, X., Li, K., Li, K., Yang, X., Chen, C.: Multiple local 3D CNNs for region-based prediction in smart cities. *Inf. Sci.* **542**, 476–491 (2021)
6. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. *Int. J. Forecast.* **22**(3), 443–473 (2006)
7. Feng, n., Guo, C., Chao, S., Zhu, Q., Wan, H.: Multi-component spatial-temporal graph convolution networks for traffic flow forecasting. *J. Softw.* **30**(3), 759–769 (2019)
8. He, R., Xiao, Y., Lu, X., Zhang, S., Liu, Y.: ST-3DGMR: spatio-temporal 3D grouped multiscale ResNet network for region-based urban traffic flow prediction. *Inf. Sci.* **624**, 68–93 (2023)
9. Jiang, R., et al.: DeepUrbanEvent: a system for predicting citywide crowd dynamics at big events. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2114–2122 (2019)
10. Lin, Z., Feng, J., Lu, Z., Li, Y., Jin, D.: DeepSTN+: context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1020–1027 (2019)
11. Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., Wang, Y.: Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *sensors* **17**(4), 818 (2017)
12. Nie, L., Jiang, D., Yu, S., Song, H.: Network traffic prediction based on deep belief network in wireless mesh backbone networks. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–5. IEEE (2017)

13. Polson, N.G., Sokolov, V.O.: Deep learning for short-term traffic flow prediction. *Transp. Res. Part C: Emerg. Technol.* **79**, 1–17 (2017)
14. Qiu, Z., Zhu, T., Jin, Y., Sun, L., Du, B.: A graph attention fusion network for event-driven traffic speed prediction. *Inf. Sci.* **622**, 405–423 (2023)
15. Ren, Y., Chen, H., Han, Y., Cheng, T., Zhang, Y., Chen, G.: A hybrid integrated deep learning model for the prediction of citywide spatio-temporal flow volumes. *Int. J. Geogr. Inf. Sci.* **34**(4), 802–823 (2020)
16. Wang, D., Cao, W., Li, J., Ye, J.: DeepSD: supply-demand prediction for online car-hailing services using deep neural networks. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 243–254. IEEE (2017)
17. Wang, Y., Li, L., Xu, X.: A piecewise hybrid of ARIMA and SVMs for short-term traffic flow prediction. In: *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part V* 24, pp. 493–502. Springer (2017)
18. Zhang, C., Zhang, H., Qiao, J., Yuan, D., Zhang, M.: Deep transfer learning for intelligent cellular traffic prediction based on cross-domain big data. *IEEE J. Sel. Areas Commun.* **37**(6), 1389–1401 (2019)
19. Zhang, C., Zhang, H., Yuan, D., Zhang, M.: Citywide cellular traffic prediction based on densely connected convolutional neural networks. *IEEE Commun. Lett.* **22**(8), 1656–1659 (2018)
20. Zhang, H., Cao, L., Gao, S.: A locality correlation preserving support vector machine. *Pattern Recogn.* **47**(9), 3168–3178 (2014)
21. Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., Li, T.: Predicting citywide crowd flows using deep spatio-temporal residual networks. *Artif. Intell.* **259**, 147–166 (2018)
22. Zhao, L., Gao, M., Wang, Z.: ST-GSP: spatial-temporal global semantic representation learning for urban flow prediction. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 1443–1451 (2022)



An Efficient Name Lookup Approach Based on Character Indexing

Dacheng He¹✉, Man Jiang², and Ke Xu³

¹ School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China

dachenghe@hnust.edu.cn

² School of Architecture and Art Design, Hunan University of Science and Technology, Xiangtan, China

³ Computer Network Information Center, Chinese Academy of Sciences, Beijing, China

Abstract. Name lookup is a key operation in new network architectures, such as Information-Centric Networking (ICN), Content Delivery Network (CDN), and 5G core networks. Due to the longest prefix match in large-scale rule tables, it faces several challenges in terms of search speed, storage overhead, and security. This paper proposes a high-performance name lookup approach, building fixed length character indexing for variable length name components. Aiming at the performance bottleneck caused by the high detection times of the mainstream multi-level hash name lookup algorithm, the character indexing is convenient to read in once for decreasing hash detections. The cache utilization of character indexing is higher than existing algorithms in hash detections, which can decrease the memory access overhead and speed up the match processing. In this way, this approach can quick filter mismatched contents which relieving the flooding attacks. Our experimental results show that this approach can improve the name lookup performance obviously in the above scenarios.

Keywords: name lookup · character indexing · network security

1 Introduction

Routing forwarding is one of the most essential functions for the Internet. In the traditional TCP/IP architecture networks, the core operation of route forwarding is IP addresses lookup. However, Internet users are more interested in contents rather than IP addresses. In the new network architectures, such as Information-Centric Networking (ICN) [1, 2], Content Delivery Network (CDN) [3], 5G core networks, etc., contents become more important gradually, instead of IP addresses. Name lookup is the core operation of route forwarding in these new network architectures.

Name lookup adopts mainly Longest Prefix Match (LPM), which is similar to IP lookup. And it may also adopt prefix match, exact match and other match strategies in the different applications and networks, such as the process of forwarding data packets by the user plane function (UPF) network element in the 5G core network, the process of

information transmit by blockchain [4–7], the process of searching the Internet of Things (IoT) devices [8, 9], etc. As the contents are far more complicated than IP addresses, Name lookup faces several challenges in terms of search speed, storage overhead, and update cost for its large-scale rule tables. In this paper, we mainly research name lookup of LPM, because its challenges are the greatest, and its approach can adapt other match strategies. And in the following, we use Longest Name Prefix Match (LNPM) refer the core process of data forwarding in name lookup.

It is proposed in this paper an efficient name lookup approach based on character indexing called CINL. CINL builds fixed length character indexes for variable length name components, aiming at the performance bottleneck caused by the high detection times of the existing mainstream multi-level hash name lookup algorithms. The character indexing is convenient to read in once for decreasing hash detections, causing cache utilization of character indexing is higher than existing algorithms in hash detections. Thus, the main contributions are as follows:

- It is proposed a storage structure of character indexing to store names, which makes a hierarchical string with variable length name components into a fixed length string.
- It is proposed a high-performance name lookup approach based on character indexing. CINL decreases the hash memory access overhead and speed up the match processing.
- It is proved the high efficiency of CINL through theoretical analysis and the experiments. CINL can improve the name lookup performance obviously for dealing with filtering mismatched names quickly.

The remaining of this paper is organized as follows. The related work is presented in Sect. 2, the detailed description of CINL is in Sect. 3. To show the effectiveness of CINL, theoretical analysis and the experiment results are shown in Sect. 4, and finally, summary and future works are presented in Sect. 5.

2 Related Work

According to different storage structures, there are mainly three types of name lookup methods, trie-based and hash table-based lookup methods. Trie-based methods always maintain a prefix trie storing different length names. In this way, many names with the same prefix share the same storage nodes, improving storage efficiency. This storage structures has relatively poor performance in search and updating process, because each process needs to operating in the whole prefix trie. Hash table-based lookup methods usually group names on their lengths, each group maintains a hash table for one length. In this way, it has high search and updating performance, and can be further accelerated by optimization strategies such as binary search and greedy search. In order to reduce conflicts or maintain additional instruction information, their storage overhead is generally high. Bloom filter-based name lookup methods have small storage space and fast search speed. They can quickly filter out a large number of mismatched name prefixes, but there is a problem of false positives.

Flooding attack, as a type of DDoS attack, typically means that malicious attackers send a large number of legitimate but meaningless name requests to the target, causing name lookup to malfunction. Researchers reduce the risk of flooding attacks in the

network by monitoring client request frequency and traffic, limiting customers with traffic anomaly and frequent requests. Compagno A, Conti M, Gasti P, et al. designed a traffic restriction mechanism to assess the level of customer threat based on the number of requests over a period of time [10]. When the number of customer requests exceeds the threshold, the system will gradually limit the customer’s traffic until their request frequency decreases before gradually recovering. The GLRT detector [11] is used for flood attack alarm and balancing detection delay, reducing network performance loss while warning of flood attacks. Although this type of mechanism can reduce the harm of flooding attacks, it would affect the normal name lookup transmission of the network.

2.1 Hash Table-based Name Lookup Methods

The hash structure is one of the main structures for storing data. And the methods based on hash structures are widely used in the name lookup processes. The main processes of hash table-based name lookup methods are as shown in Fig. 1. Firstly, these methods store the name prefixes in the corresponding hash tables, which building by the length of each name prefix. The matching processes mainly consist of two stages. Firstly, they select the corresponding hash table based on the length of the target name, then match it within the corresponding hash table through hash detection. And finally they perform string matching based on the hash detection results to determine whether the target name matches the name prefix successfully.

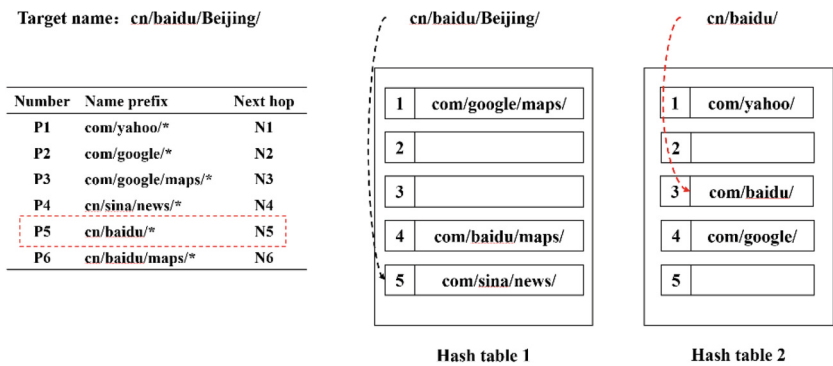


Fig. 1. An example storage structure of Hash table-based.

So W et al. proposed a two-stage name lookup method [12], which choose different strategy to select initial length. It selects the length M with the most name prefixes as the initial length, instead of the longest length. To ensure the correctness, its hash table with length M stores extra information for name prefixes longer than M. This method further improves the efficiency of name lookup through the fingerprint search, the pretreatment and other ways. Yuan H & Crowley P proposed an efficient name lookup method based on the different data processing pressure by different network locations [13]. In the Internet, the data processing of the core routers and networks is far fore than that of the edge routers and networks. The method deployed the fingerprint structure to name

prefixes aggregation in core networks, and the edge networks finish the complete name lookup processing. Since the substantial traffic in the core network and little traffic in the edge network, the method can schedule resources reasonably while bringing efficient performance. Wang Y, Xu B, Tai D, et al. proposed a greedy mechanism to improve name lookup efficiency [14]. The method used perfect hash structures and fingerprint indexes to increase match processes.

2.2 Bloom Filter-based Name Lookup Methods

Bloom filter is a structure with low memory consumption and high lookup efficiency used in many fields [15, 16]. The features of bloom filter are suitable for name lookup. And researchers proposed several bloom filter-based name lookup methods. The processes of bloom filter-based name lookup methods are similar to hash table-based name lookup methods. The bloom filter structures are deployed as aided tools to accelerate the name lookup processes.

Wang Y, Pan T, Mi Z, et al. proposed an efficient method called NameFilter [17], it also has two steps which are similar to hash-based methods. It expanded bloom filters from 1- dimension to N- dimension and transposed them. NameFilter decreased the times of detections from N to 1. In order to align all the bloom filters, NameFilter needs extra cost. Dai H, Lu J, Wang Y, et al. proposed the BFAST methods [18]. The method used aided count bloom filters to merge several indexes in one. This method also utilized multi-thread and other optimization methods to name lookup efficiently.

In summary, hash structure methods have good performance in speed and broad applicability, while bloom filter structure methods can combine other data structures to further increase forwarding efficiency. Moreover, bloom-filter can filter mismatched target name quickly, which relieving the flooding attack.

3 A Name Lookup Approach based on Character Indexing

3.1 Motivation

As mentioned before, the name structures with hash tables and bloom filters have high performance in general. Our paper selects this type of structure to improve the efficiency of name lookup. These existing name lookup algorithms typically are grouped names by length and select the name length as filtering rules since the impacts of IP lookup methods. However, the name length is a natural attribute of each name, which cannot reflect the feature of names. As a filtering rule, it hardly works. It means that any name needs to match n times while n is the length of the name, leading to the relatively high cost for the matching detection process.

We notice that each name is composed of several words and vocabulary, which can reflect the feature of the name itself and have higher discrimination than length. If using them as the filtering rule instead of the length, it can significantly reduce the times of name matching and improve the success rate of name matching. So we extract corresponding characters from each name as indexes, reducing the detection costs from 3 aspects. First, hash calculation only involves the character indexing rather than the

whole name, decreasing the computational costs. Second, each character indexing has the fixed length, which is convenient to read in once for calculation. Finally, the space of bloom filters is much smaller than that of hash structures, multiple consecutive bloom filter detections can utilize the memory cache more efficient than hash detections. In this way, the meaningless names caused by flooding attack would be quickly filtered out.

3.2 The Name Lookup Approach based on Character Indexing

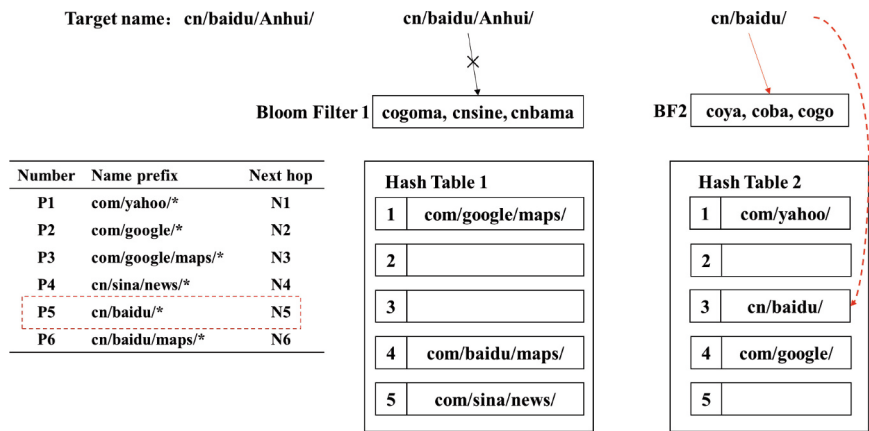


Fig. 2. An example storage structure of character indexing

As mentioned before, if the fixed length of index structures is 2, which means the length of name character indexing is 2n (n is the number of name components). For a target name, the approach gets its character indexing in the same way. As similar to the traditional matching process, this approach also has two steps, the length matching process and the name matching process. First CINL utilizes character indexing to determine the length of the target name prefix. Second CINL matches the name prefixes in hash table with this length. Comparing with traditional name lookup with hash tables, CINL just needs once hash lookup after determining the length as shown in Fig. 2.

When the target name “cn/baidu/Anhui” comes, CINL gets its character indexing as “cnbaAn” (n = 2). Considering the length of longest name prefix is 3, the target name first matches “cnbaAn” in the 3-length bloom filter (BF1). Since matching failed in BF1, it continue to match “cnba” in the 2-length bloom filter (BF2). As matching successful in BF2, the target name matches in Hash Table 2(HT2) and gets the prefix P5. During LNPM, it just has only once hash detection since the index detections with bloom filters. In addition, the character indexing with fixed length can matches in bloom filter detections regularly, while other approaches need to reload the information for different lengths. Algorithm 1 shows the operations of the name lookup process in CINL.

Algorithm 1: name lookup approach based on character indexing

Input: T , the target name; I , the character indexing; $L[MAX]$, the record array; $HT[MAX]$, the hash tables

Output: $Length$, the length of matching successful

```

1: Calculate the length of  $T$  as  $TL$ ; # calculate the length of the target name
2: for  $i = TL$  to 1 do
3:   Select  $I$  in  $L[i]$ ; #select the corresponding character indexing
4:   Match  $I$  in  $HT[i].bloom$ ; #bloom filter detections
5:   if Matched
6:     then Match  $T$  in  $HT[i].hash$ ; #hash detections
7:     if Matched
8:       then Record the length of location; #record the successful length
9:   end for
10: return  $Length$ .
```

3.3 Theoretical Analysis

This subsection discusses the theoretical analysis of CINL. As mentioned above, CINL can match the name prefix efficiently with only once hash detections. In the traditional hash methods, the time cost T equals L (the maximum value between the length of the target name and the length of the longest name prefix) times hash detections in the worst case. Formula 1 shows the time cost during LNPM with 2-steps name lookup approaches. T_{length} refers to the time cost of confirm target length, T_{hash} refers to the time cost of hash detections. Since T_{hash} is much larger than T_{length} , it usually uses other ways to reduce the hash detections. In this way, we reduce the value of L_2 as possible. For example, it can increase L_1 and decrease L_2 by bloom filter, making fewer total consumption. On this basis, CINL further reduces the value of m by character indexing. CINL can get all the character indexing for read each name information once, instead of recalculating each index of different lengths in other approaches. Therefore, CINL can improve more efficient in LNPM.

$$T = m \sum_1^{L_1} (T_{Length}) + n \sum_1^{L_2} (T_{Hash}) \quad (1)$$

Since CINL uses the bloom filters, it should calculate the false-positive F , which is defined as in Formula 2 and 3, k is the hash times of a bloom filter. And n refers the numbers of elements, it means how many the “1” bit in bloom filters. And m refers how many bits of each bloom filter. If $k = 3$, $m = 10n$, false-positive F is only $8.12 * 10^{-7}$.

$$F = (1 - p)^k \quad (2)$$

$$p = e^{(-kn/m)} \quad (3)$$

It follows that CINL utilize character indexing with bloom filters to speed up LNPM efficient, which can substantially reduced the times of hash detections.

4 Experimental Evaluation

4.1 Experimental Environment

The server configuration used in the experiment is as follows: the CPU is AMD EPYC 7F52 16 Cores, 256GB of memory, and the operating system is CentOS 7.8.2003 (with Linux 3.10 kernel).

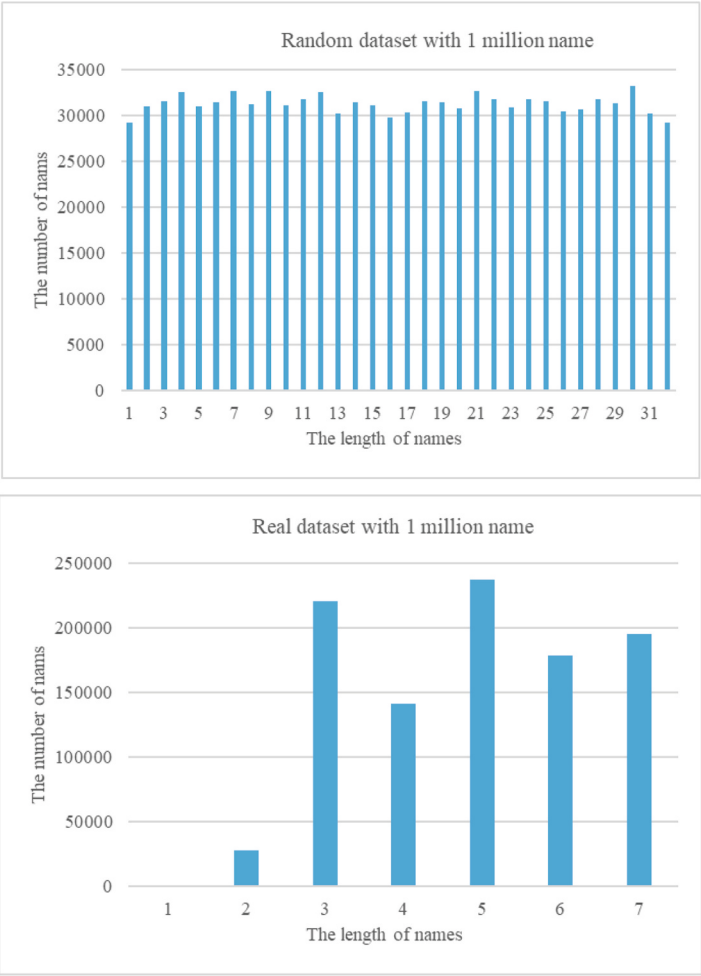


Fig. 3. Number distribution of 1 million names in different scenarios

Experimental dataset: In order to comprehensively evaluate the performance of the algorithm, two types of datasets were selected and generated to simulate the name lookup process in different scenarios. The experiment selected a real-world URL dataset as the rule set, which was collected and obtained through the Internet. The URLs of multiple

domestic and foreign websites with high usage rates, as well as other URL links related to their content, were collected to ensure that the final real dataset not only comes from real-world URL sets, but also avoids most of them coming from a few websites, which leads to high repetition rates in the first few components and better restores the real situation. The maximum length of a name in a real dataset is 7. By selecting 7970 words from the Oxford Dictionary to form a word bank, one word was randomly selected as a component, and the length of the name was randomly generated, with a maximum of 32 words randomly selected from the word bank to form a name. This generation method itself is more in line with the definition of names. The maximum length of a real dataset name is 7, and the distribution of the number of lengths is also uneven; The maximum length of the random dataset name is 32, and the distribution of the number of lengths is average. Figure 3 shows a real dataset with a scale of 1 million, where the distribution of lengths is uneven; Fig. 3 shows a random dataset with a scale of 1 million, and the number of names of each length is approximately evenly distributed between 29000–33000. The experiment uses real datasets in matching scenarios and real and random datasets in mismatched scenarios. The real dataset is divided into six scales: 0.5 million, 1 million, 1.5 million, 2 million, 2.5 million, and 3 million; The random dataset is divided into four sizes: 0.5 million, 1 million, 3 million, and 5 million.

4.2 Time Consumption

Experimental comparison methods: In order to evaluate the performance of CINL, multiple existing length based hierarchical hash name lookup algorithms were selected for comparison in the experiment. Including 1) LNPM method based on fingerprint hashing [19], 2) the binary name lookup method [20], and 3) the name lookup method based on greedy mechanism [21]. In addition, CBS [22] method and NameTrie [23] method from recent years were selected for performance comparison.

Experimental dataset: In order to comprehensively evaluate the performance of the algorithm, two types of datasets were selected and generated to simulate the name lookup process in different scenarios. The experiment selected a real-world URL dataset as the rule set, which was collected and obtained through the Internet. The URLs of multiple domestic and foreign websites with high usage rates, as well as other URL links related to their content, were collected to ensure that the final real dataset not only comes from real-world URL sets, but also avoids most of them coming from a few websites, which leads to high repetition rates in the first few components and better restores the real situation. The maximum length of a name in a real dataset is 7. By selecting 7970 words from the Oxford Dictionary to form a word bank, one word was randomly selected as a component, and the length of the name was randomly generated, with a maximum of 32 words randomly selected from the word bank to form a name. This generation method itself is more in line with the definition of names. The maximum length of a real dataset name is 7, and the distribution of the number of lengths is also uneven; The maximum length of the random dataset name is 32, and the distribution of the number of lengths is average. Figure 3 shows a real dataset with a scale of 1 million, where the distribution of lengths is uneven; Fig. 3 shows a random dataset with a scale of 1 million, and the number of names of each length is approximately evenly distributed between 29000–33000. The experiment uses real datasets in matching scenarios and real and

random datasets in mismatched scenarios. The real dataset is divided into six scales: 0.5 million, 1 million, 1.5 million, 2 million, 2.5 million, and 3 million; The random dataset is divided into four sizes: 0.5 million, 1 million, 3 million, and 5 million.

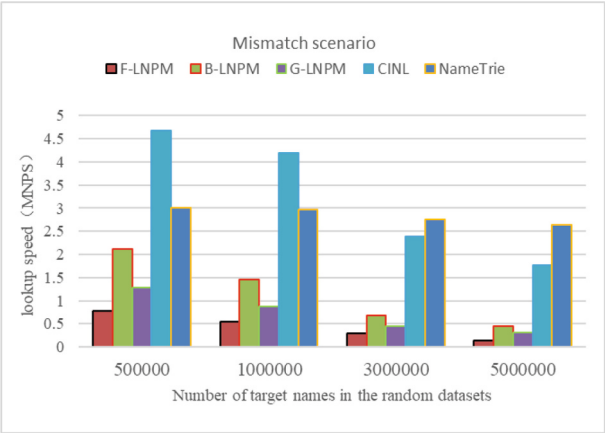


Fig. 4. Name lookup efficient of different methods in the real datasets

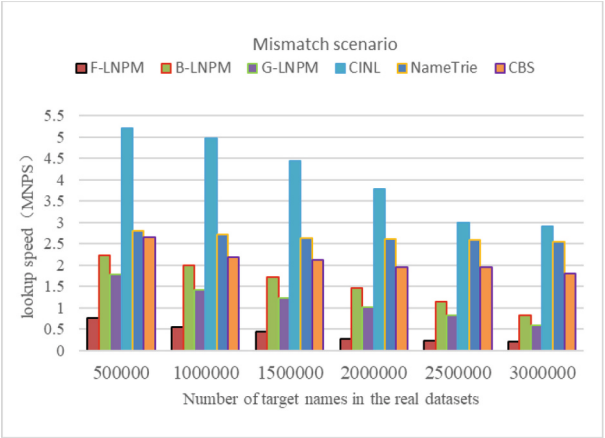


Fig. 5. Name lookup efficient of different methods in the real datasets

To relieve the flooding attack, name lookup approaches would have well performance in mismatch scenarios. Figure 4 shows the performance of various methods in random datasets. Taking 1 million random name prefixes as an example, the search performance of CINL, F-LNPM, G-LNPM, and NameTrie algorithms are 4.19MNPS, 0.55MNPS, 46MNPS, 0.88MNPS, and 2.97MNPS, respectively. CINL increases lookup speed successively for 7.62 times, 2.87 times, 4.76 times, and 1.41 times than other methods. It shows that in mismatch scenarios, CINL and NameTrie, name lookup methods with the filtering function, would have significant performance advantages. The reason is that in

mismatch scenarios, all hash detections will ultimately fail, approaches with filtering function would replace hash detections with less cost methods in mismatch scenarios, achieving performance improvements. NameTrie is a prefix tree lookup algorithm that performs name prefixes lookup from short to long. With the number of mismatch target names increases, its performance would better. CINL would be good at handling in the mismatch scenarios, and as the mismatch name increasing, its performance advantage is more obvious compared to other methods. Figure 5 also shows the performance of various methods in real datasets. The overall performance is basically consistent with the random situation.

5 Conclusions and Future Work

Name lookup is a key operation in new network architectures, it faces several challenges in terms of search speed, storage overhead, and security. The paper proposes an efficient name lookup approach CINL. It utilizes character indexes to reduce the number of hash detections and improve the efficiency of name lookup. CINL can reduce the computational cost of detection from 3 aspects. First, hash calculation only involves the character indexing rather than the whole name, decreasing the computational costs. Second, each character indexing has the fixed length, which is convenient to read in once for calculation. Finally, the space of bloom filters is much smaller than that of hash structures, multiple consecutive bloom filter detections can utilize the memory cache more efficient than hash detections. CINL can relieve the flooding attack by quickly filter out the meaningless name prefixes. The experimental results show that CINL can even improve up to 70% in the flooding attack scenarios. CINL can be combined with existing hash search schemes, this part of the work would be implemented in the future.

Acknowledgment. The work was supported by the National Natural Science Foundation of China under Grants Number 62472168 and 62072170.

References

1. Liang, C., Yu, F.R., Zhang, X.: Information-centric network function virtualization over 5g mobile wireless networks. *IEEE Network* **29**(3), 68–74 (2015). <https://doi.org/10.1109/MNET.2015.7113228>
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) *Conference 2016, LNCS*, vol. 9999, pp. 1–13. Springer, Heidelberg (2016)
3. Abdallah, H.B.H., Louati, W.: Ftree-CDN: hybrid CDN and P2P architecture for efficient content distribution. In: *Proceedings of the 27th Euromicro Int. Conference on Parallel, Distributed and Network-Based Processing*. Piscataway, NJ: IEEE, pp. 438–445 (2019)
4. Liang, W., Liu, Y., Yang, C., et al.: On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: a comprehensive survey. *ACM Comput. Surv.* **56**(12), 1–35 (2024)
5. Zhou, S., Li, K., Chen, Y., et al.: TrustBCFL: mitigating data bias in IoT through blockchain-enabled federated learning. *IEEE Internet of Things Journal* (2024)
6. Liang, W., Yang, Y., Yang, C., et al.: PDPChain: a consortium blockchain-based privacy protection scheme for personal data. *IEEE Trans. Reliab.* **72**(2), 586–598 (2022)

7. Zhang, S., Yan, Z., Liang, W., et al.: BCAE: A blockchain-based cross domain authentication scheme for edge computing. *IEEE Internet of Things Journal* (2024)
8. Cai, J., Liang, W., Li, X., et al.: Gtxchain: a secure iot smart blockchain architecture based on graph neural network. *IEEE Internet of Things Journal* (2023)
9. Zhang, S., Hu, B., Liang, W., et al.: A trajectory privacy-preserving scheme based on transition matrix and caching for IIoT. *IEEE Internet of Things J.* (2023)
10. Compagno, A., Conti, M., Gasti, P., et al.: Poseidon: mitigating interest flooding DDoS attacks in named data networking. In: 38th Annual IEEE Conference on Local Computer Networks. IEEE, pp. 630–638 (2013)
11. Nguyen, T., Mai, H.L., Cogranne, R., et al.: Reliable detection of interest flooding attack in real deployment of named data networking. *IEEE Trans. Inf. Forensics Secur.* **14**(9), 2470–2485 (2019)
12. So, W., Narayanan, A., Oran, D.: Named data networking on a router: fast and DoS-resistant forwarding with hash tables. In: *Architectures for Networking and Communications Systems*. IEEE, pp. 215–225 (2013)
13. Yuan, H., Crowley, P.: Scalable pending interest table design: From principles to practice. In: *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, pp. 2049–2057 (2014)
14. Wang, Y., Xu, B., Tai, D., et al.: Fast name lookup for named data networking. In: 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS). IEEE, pp. 198–207 (2014)
15. Kirsch, A., Mitzenmacher, M.: Less hashing, same performance: building a better bloom filter. *Random Structures and Algorithms* **33**(2), 187–218 (2008)
16. Liang, W., Long, J., Li, K.C., et al.: A fast defogging image recognition algorithm based on bilateral hybrid filtering. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* **17**(2), 1–16 (2021)
17. Wang, Y., Pan, T., Mi, Z., et al.: Namefilter: Achieving fast name lookup with low memory cost via applying two-stage bloom filters. In: 2013 Proceedings IEEE INFOCOM. IEEE, pp. 95–99 (2013)
18. Dai, H., Lu, J., Wang, Y., et al.: BFAST: unified and scalable index for NDN forwarding architecture. In: 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, pp. 2290–2298 (2015)
19. Seo, J., Lim, H.: Bitmap-based priority-NPT for packet forwarding at named data network. *Comput. Commun.* **130**, 101–112 (2018)
20. He, D., Zhang, D., Xu, K., et al.: A fast and memory-efficient approach to NDN name lookup. *China Communications* **14**(10), 61–69 (2017)
21. So, W., Narayanan, A., Oran, D., et al.: Named data networking on a router: forwarding at 20 gbps and beyond. *ACM SIGCOMM Computer Communication Review* **43**(4), 495–496 (2013)
22. Huang, K., Wang, Z., Xie, G.: Scalable high-speed NDN name lookup. *Proceedings of the 2017 Symposium on Architectures for Networking and Communications Systems* **2018**, 55–65 (2018)
23. Ghasemi, C., Yousefi, H., Shin, K.G., et al.: On the granularity of trie-based data structures for name lookups and updates. *IEEE/ACM Trans. Networking* **27**(2), 777–789 (2019)



A Multi-channel Target Direction Speech Separation Model Based on Efficient Convolutional Networks

Zaixin Chen¹, Yang Mao²(✉), Tihua Yan³, and Biqing Li²

¹ State Grid Sichuan Nanchong Power Supply Company, Nanchong, China

² State Grid Sichuan Guangyuan Power Supply Company, Guangyuan, China
maoyang6196@163.com

³ Guangzhou Institute, Xidian University, Guangzhou, China

Abstract. The purpose of speech separation algorithms is to solve the “cocktail party problem” by separating the voice of the main speaker from the mixed speech. Currently, most speech separation algorithms based on deep learning usually require permutation invariant training due to the need to separate multiple target sources of speech at once. However, this paper proposes a new multi-channel target direction speech separation model, which only separates speech in the target direction without separating all sources together, thereby directly avoiding permutation invariant training. In this model, multi-channel spatial features guide the model to learn speech related information in the target direction. And in the learning of temporal information, unlike other speech separation models, this paper uses an efficient convolutional network, U-shaped convolutional block network, as the separator of the model. This paper proves that this temporal model is more efficient in learning spatiotemporal features. Finally, the model is tested using the multi-channel speech dataset created. The experiment shows that the proposed model has better separation performance compared to other single channel speech separation models and multi-channel speech separation models.

Keywords: Speech separation · Speech extraction · Convolutional network

1 Introduction

Speech separation refers to the process of extracting target speech from mixed sound signals, which is widely used in areas such as enhancing speech recognition, hearing aids, and multi-channel communication systems. In multi-channel speech separation, effectively handling interference and noise at the same frequency caused by multiple speech signals remains challenging, as speakers’ voices are generated from directional sources at different locations. Multi-channel speech separation typically involves two mainstream approaches: spatial filtering by combining with beamforming or imitating the filtering and summation method of beamforming, and speech separation incorporating spatial features.

Traditional speech enhancement methods using beamforming typically consist of two parts: initial filtering with a linear spatial filter, followed by further noise reduction with a post-filter. When applying deep learning methods to spatial filtering [1, 2], the main advantage lies in the nonlinear capabilities of deep learning. Different neural networks are used in literature [3] and [4] to obtain time-frequency masks, which are then employed in Minimum Variance Distortionless Response (MVDR) beamforming. In contrast, literature [5] utilizes another Generalized Eigenvalue (GEV) beamformer. However, since these beamformers are purely based on statistical theory, preliminary neural network estimation errors in the first stage can significantly affect subsequent beamforming results. To address this, some models adopt an iterative approach to approximate the optimal beamformer performance. Beam-Guided TasNet [6] processes speech in two stages: the first stage separates speech using multi-channel TasNet and MVDR beamforming, while the second stage guides multi-channel TasNet to separate the signals again using MVDR beamformed signals. This second stage is repeated multiple times for better separation. Another solution to this problem is to directly use neural networks to learn the beamformer coefficients instead of cascading them with neural networks, and then imitate the beamformer's filtering and summation method [7–9] to estimate signals.

Another approach for multi-channel speech separation involves directly incorporating additional spatial features into the model input, transforming the multi-channel speech separation problem into a single-channel speech separation problem on a reference microphone. Wang et al. [10] designed two directional features to improve speech separation: Inter-channel Phase Differences (IPD) and output from fixed beamforming. Similarly, Gu et al. extracted speech in the target direction using multiple spatial features [11] and later proposed combining spatial features for speech separation in the time domain [12]. With recent significant advancements in deep learning across various domains, its application in speech separation algorithms has garnered widespread attention. Currently, various deep learning-based methods, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs), and Transformer models, have achieved remarkable performance improvements in speech separation tasks. These methods capture complex time-frequency structures and long-term dependent features by learning nonlinear mapping relationships from mixed signals to clean targets. Additionally, the Permutation Invariant Training (PIT) strategy has been introduced to address frequency permutation ambiguity in multi-channel speech separation. Despite achieving significant performance improvements on various benchmark datasets, these methods still face limitations, such as computational complexity, training data requirements, and generalization capabilities.

This paper focuses on studying speech separation models based on this approach, exploring the impact of various spatial features on the performance of speech separation models, and proposing a multi-channel target direction speech separation model.

2 Basic Theory

2.1 Multi-channel Spatial Features

When the input speech is multi-channel speech collected by a microphone matrix, the speech of the same sound source captured by different microphones differs in phase and amplitude. Therefore, multi-channel speech contains spatial location information of the sound source. It is easy to think of utilizing the spatial characteristics of multi-channel speech as additional information input for the network, aiming at speech separation in the target direction. This can fundamentally solve the matching problem between input and output speech. This is the multi-channel target directional speech separation model.

Next, we introduce several common spatial features of speech in multi-channel speech separation models.

(1) Inter-channel Phase Difference (IPD)

The formula for calculating the inter-channel phase difference is as follows:

$$IPD_{t,f} = \angle Y_{t,f}^p - \angle Y_{t,f}^q \quad (1)$$

$$\cos IPD_{t,f} = \cos(\angle Y_{t,f}^p - \angle Y_{t,f}^q) \quad (2)$$

$$\sin IPD_{t,f} = \sin(\angle Y_{t,f}^p - \angle Y_{t,f}^q) \quad (3)$$

Given a pair of microphones, assuming microphone p and microphone q, let $X_p(t, f)$ and $X_q(t, f)$ represent the spectral representations after the short-time Fourier transform of the signals received by microphone p and microphone q, respectively, where t is the time index and f is the frequency index. $\cos(IPD)$ represents the cosine function of the inter-channel phase difference, and $\sin(IPD)$ represents the sine function of the inter-channel phase difference. In the absence of reverberation, IPD can be approximated as a linear function of frequency with a deterministic time delay. According to the ideal amplitude-frequency mask, different time-frequency bins are independently occupied by sound sources from different directions, and the time-frequency bins occupied by sound sources from the same direction have the same time delay, which can be represented by IPD. Therefore, for each frequency band, the sound source belonging to each time-frequency bin can be determined based on the IPD of that bin.

(2) Angular Feature (AF)

When the azimuth angle of the target speech is known, the angular feature AF is usually used to characterize the speech in the target direction. The calculation formula is shown in (4). Assuming that the microphone matrix has M microphones, AF is calculated by selecting P microphones excluding the reference microphone. In the formula below, $d_i(\theta, f)$ represents the steering vector of the ith microphone at target azimuth θ and frequency f. IPD_{iq} represents the inter-channel phase difference between the ith microphone and the reference microphone. $\Re$ denotes taking the real part. The design principle of $AF_i(\theta, f)$ is that when a time-frequency bin of mixed speech is mainly occupied by speech from the target direction, the cosine

similarity between the steering vector $d_i(\theta, f)$ in the target direction and IPD_{iq} approaches 1; otherwise, it is 0.

$$AF_{\theta,tf} = \text{Re} \left(\sum_{i=1}^P \frac{\langle a_{\theta,f}^i, \exp(IPD_{t,f}^i) \rangle}{\|a_{\theta,f}^i \exp(IPD_{t,f}^i)\|} \right) \quad (4)$$

Based on the time-frequency sparsity of speech, when the direction of the speaker of the interference source is also known apriori, a pre-masking operation is often applied to AF. The specific operation is shown below:

$$AF_{\theta,tf} = AF_{\theta,tf} \Gamma(AF_{\theta,tf} - AF_{\beta,tf}), \quad \beta = 1, 2, \dots, N \quad (5)$$

where θ_{others} represents the directions of other speakers, N is the number of interfering speakers, and the function $\Gamma()$ means that if the calculation result inside the brackets is positive, the function result equals 1; otherwise, it equals 0.

(3) Multi-channel Convolution and MCS (Multi-Channel Summation)

Multi-channel convolution and MCS resemble the filtering process of beamforming. The main idea is to use a data-driven approach to learn latent spatial features through a two-dimensional convolutional layer, aiming to make each two-dimensional convolution kernel a delay-and-sum filter pointing in a certain direction. The specific calculation can be expressed by the following formula:

$$MCS^{(n)} = \sum_{c=1}^C y_c \otimes k_c^{(n)} \quad (6)$$

The set $K = \{\dots, k^{(n)}, \dots\} \in \mathbb{R}^{(C \times L \times N)}$ consists of a group of two-dimensional convolutional kernels, where each $k^{(n)} = \{\dots, k_c^{(n)}, \dots\} \in \mathbb{R}^{(C \times L)}$ has a kernel size of $C \times L$, representing the convolution operation. From a filter perspective, when C represents the number of signal channels and L represents the window length, $k^{(n)}$ serves as the weight coefficients in beamforming. Once K is trained, each $k^{(n)}$ transforms into a directional delay-sum beamformer, and $MCS^{(n)}$ represents the delay-sum beamforming response in a certain direction.

2.2 U-shaped Convolutional Block Network

The U-shaped convolutional block network, proposed by Wang et al., is a convolutional module inspired by the U-net architecture used in image segmentation algorithms [13]. By stacking such U-shaped convolutional blocks, the network is designed to mine both shallow and deep features of speech signals in the temporal domain, thereby learning the temporal information of speech. The specific structure of the U-shaped convolutional block is shown in Fig. 1.

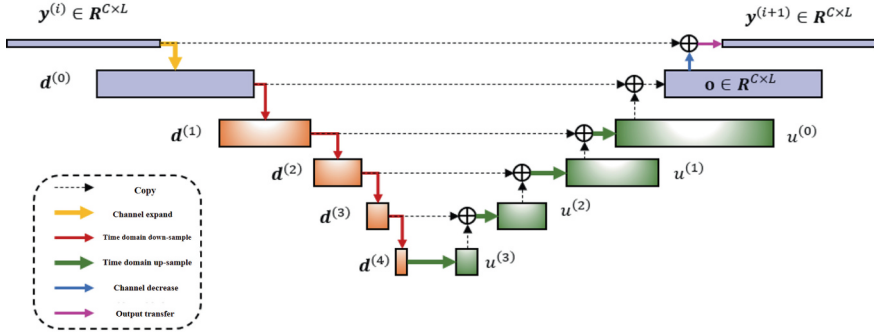


Fig. 1. U-shaped Convolutional Block Network

The U-shaped convolutional block first expands the channel dimension from C to C_U , then performs temporal down-sampling through Q consecutive depth-wise separable convolution operations to increase the receptive field, obtaining higher-order temporal features. This is followed by nearest-neighbor temporal interpolation for Q times upsampling to restore the time resolution consistent with the previous layer, and features of the same layer are fused through residual connection operations to create new features and address the problem of gradient vanishing. The final convolution layer restores the dimension size of the output features to match the original input features, thus maintaining the same feature dimensions for the input and output of each U-shaped convolutional block. The specific algorithm process is shown as Algorithm 1. Here, $y^{(i)}$ represents the input of the i th U-shaped convolutional block network, and $y^{(i+1)}$ represents the output of the i th U-shaped convolutional block network; $\text{PReLU}_C: \mathbb{R}^{(C \times L)} \rightarrow \mathbb{R}^{(C \times L)}$ represents the PReLU activation function with C parameters. Its mathematical form is specified as:

$$\text{PReLU}_C(y_{i,j}) = \max(0, y_{i,j}) + a_i \cdot \min(0, y_{i,j}), i = 1 \cdots C, j = 1 \cdots L \quad (7)$$

In the algorithm, $\text{Conv1D}_C(C, K, S): \mathbb{R}^{(C_{\text{in}} \times L_{\text{in}})} \rightarrow \mathbb{R}^{(C \times L)}$ represents a one-dimensional convolution with output channels C , kernel size K , and stride S . $\text{DWConv1D}_C(C, K, S): \mathbb{R}^{(C_{\text{in}} \times L_{\text{in}})} \rightarrow \mathbb{R}^{(C \times L)}$ represents a one-dimensional depth-wise separable convolution with the same parameters. $\text{LN}(\cdot)$ represents layer normalization. $\text{l}_U: \mathbb{R}^{(C \times L_{\text{in}})} \rightarrow \mathbb{R}^{(C \times U \times L_{\text{in}})}$ denotes the nearest-neighbor linear interpolation, where U is the upsampling factor. For simplicity, subsequent convolution operations in the document are described in the above format.

Algorithm 1: Forward Propagation Process of the U-shaped Convolutional Block Network

Input : $y^{(i)} \in \mathbb{R}^{(C \times L)}$

Output : $y^{(i+1)} \in \mathbb{R}^{(C \times L)}$

// Expand channel dimension

$$q \leftarrow \text{PReLU}_{C_U} \left(\text{LN} \left(\text{Conv1D}_{C_U, l, 1} \left(y^{(i)} \right) \right) \right)$$

$$d^{(0)} \leftarrow \text{PReLU}_{C_U} \left(\text{LN} \left(\text{DWConv1D}_{C_U, K_U, l} (q) \right) \right)$$

for $j = 1; j++$; while $j \leq Q$ do

// Continuously downsample through depth-wise separable convolutions

$$d^{(j)} \leftarrow \text{PReLU}_{C_U} \left(\text{LN} \left(\text{DWConv1D}_{C_U, K_U, S_U} \left(d^{(j-1)} \right) \right) \right)$$

$$d^{(j)} \leftarrow \text{PReLU}_{C_U} \left(d^{(j)} \right).$$

end

$$u^{(Q)} \leftarrow d^{(Q)}$$

for $j = Q - 1; j--$; while $j \geq 0$ do

// Upsampling

$$u^{(j)} \leftarrow d^{(j)} + I_{S_U} \left(u^{(j+1)} \right)$$

end

$$o \leftarrow \text{LN} \left(\text{Conv1D}_{C, l, 1} \left(\text{PReLU}_{C_U} \left(\text{LN} \left(u^{(0)} \right) \right) \right) \right)$$

return $\text{PReLU}_C(y^{(i)} + o)$

Wang and others have demonstrated that the U-shaped convolutional block network is more efficient for single-channel speech separation tasks compared to the TCN (Temporal Convolutional Network). Moreover, models constructed with U-ConvBlock require fewer parameters and floating-point operations compared to models based on TCN or Recurrent Neural Network architectures, making them more suitable for devices with limited computational power. Therefore, this paper proposes the use of this architecture in multi-channel target direction speech separation models.

3 Multi-channel Target Direction Speech Separation Model

The proposed multi-channel target direction speech separation model, TSS-UConvNet, is structured as shown in Fig. 2, following the commonly used encoder-separator-decoder architecture for speech separation. The model's inputs include the mixed speech from an array of M microphones, denoted as $x \in \mathbb{R}^{(M \times T)}$, and the azimuth angles of all speakers, $\theta = [\theta_{\text{target}}, \theta_{\text{interfere1}}, \dots]$, where θ_{target} is the azimuth angle of the target speech, and the others are angles of interfering speeches. The encoder module extracts temporal and various spatial features from the multi-channel speech input, then fuses these spatio-temporal features and feeds them into the separator composed of

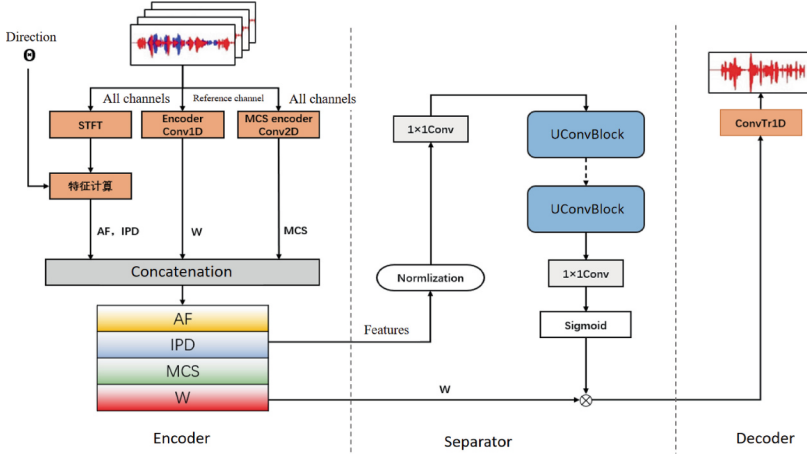


Fig. 2. TSS-UConvNet Network Architecture Diagram

stacked U-shaped convolutional blocks. The separator generates a mask for the target direction speech, which, when element-wise multiplied with the encoded mixed speech, results in the encoded target speech. Finally, the target speech is decoded through a one-dimensional transposed convolution. The algorithmic process of each module is described sequentially below.

(1) Encoder Module

The encoder module is composed of three parts: the time-domain encoder, the STFT (Short-Time Fourier Transform) layer, and the 2D convolution layer.

The time-domain encoder selects one channel from the input multi-channel mixed speech $x \in \mathbb{R}^{(M \times T)}$ as the reference channel, resulting in the reference channel speech $x_{\text{ref}} \in \mathbb{R}^{(1 \times T)}$. The reference channel speech x_{ref} is then frame-overlapped and divided into speech segments of length K_{ε} , denoted as $x_i \in \mathbb{R}^{(1 \times K_{\varepsilon})}$ for $i = 1, 2, \dots, L$, where L is the total number of speech segments. Similar to the short-time Fourier transform used in signal processing, the encoder transforms the time-domain speech signal into a latent domain representation with C_{ε} dimensions, $X_i \in \mathbb{R}^{(1 \times C_{\varepsilon})}$, through a one-dimensional convolution operation. This convolution acts fundamentally as using the convolution kernel as a basis function to represent the speech signal, leveraging data-driven methods to derive the most suitable C_{ε} -dimensional feature representation. The specific computation process of the time-domain encoder is as follows:

$$X_i = \text{ReLU}(\text{Conv1D}_{C_{\varepsilon}, K_{\varepsilon}, K_{\varepsilon}/2}(x_i)), i = 1, 2, \dots, L \quad (8)$$

The one-dimensional convolution has C_{ε} channels, a receptive field of K_{ε} , and a stride of $K_{\varepsilon}/2$. After encoding, x_{ref} yields the high-dimensional features $W = [X_1, X_2, \dots, X_L] \in \mathbb{R}^{(C_{\varepsilon} \times L)}$.

The STFT layer transforms the original waveform of the mixed speech from all channels into the frequency domain. It then calculates the inter-channel phase differences (IPD) according to formula (4-1), denoted as $\text{IPD}_{(k,f)}^i$ for $i = 1$ to P ,

where k is the index of the speech segment. The total number of segments is the same as in the encoder, ensuring consistent time steps between features. This requires that the size of the convolutional receptive field and the stride in the encoder match the window length and frame shift in the STFT. P , the number of channel pairs, is given as $P = M(M-1)/2$ if M is the number of channels. After calculating the inter-channel phase differences, the spatial features $\cos\text{IPD} \in \mathbb{R}^{(C_s \times L)}$, $\sin\text{IPD} \in \mathbb{R}^{(C_s \times L)}$, and $\text{AF} \in \mathbb{R}^{(C_a \times L)}$ are computed sequentially. Assuming FFT is the number of frequency domain samples in the Fourier transform, $C_s = P \cdot (\text{FFT}/2 + 1)$ and $C_a = \text{FFT}/2 + 1$.

Similarly, the 2D convolution layer ensures that the extracted multi-channel convolution and $\text{MCS} \in \mathbb{R}^{(C_M \times 1 \times L)}$ are consistent in time steps with the previous features, setting the convolution kernel size to $M \times K_\epsilon$.

Finally, all feature encodings $\{W, \sin\text{IPD}, \cos\text{IPD}, \text{AF}, \text{MCS}\}$ are concatenated along the channel dimension to achieve feature fusion, resulting in the spatio-temporal feature encoding $F \in \mathbb{R}^{(C_e \times L)}$, where $C_e = C_\epsilon + 2C_s + C_a + C_M$.

(2) Separator

The specific process of the separator is as follows:

1. The spatio-temporal feature encoding F first passes through a LayerNorm layer, which normalizes each channel in the time domain. Subsequently, a one-dimensional convolution with a kernel size of 1 is used to reduce the number of channels to C .

$$y_0 = \text{Conv1D}_{C,1,1}(\text{LN}(F)) \in \mathbb{R}^{C \times L} \quad (9)$$

2. Next, the intermediate feature y_0 undergoes nonlinear transformation through B stacked U-shaped convolutional block networks. Utilizing the U-shaped convolutional block network to extract and aggregate information across multiple resolution levels, the output feature $y_B \in \mathbb{R}^{(C \times L)}$ is obtained. Finally, to derive the mask m , the number of channels in y_B needs to be restored to match that of the temporal encoding W .

$$m = \text{Sigmoid}(\text{Conv1D}_{C_\epsilon,1,1}(y_B)) \in \mathbb{R}^{C_\epsilon \times L} \quad (10)$$

It is important to note that the model extracts only the speech in the target direction and does not output interfering speech. Therefore, a Sigmoid function is used as the activation function, adjusting each element in each channel of the mask m to range from 0 to 1.

3. Finally, the mask m for the target speech is element-wise multiplied with the temporal encoding w to obtain the encoded target speech V . here, $\odot$ Denotes element-wise multiplication.

$$V = m \odot W \quad (11)$$

(3) Decoder

The decoder operates in reverse to the time-domain encoder, using a one-dimensional transposed convolution to transform the encoded target speech V back into a waveform representation, thus producing an estimate of the target direction

speech, denoted as $\hat{s}$.

$$\hat{s} = \text{ConvTr1}D_{C_\varepsilon, K_\varepsilon, K_\varepsilon/2}(V) \quad (12)$$

4 Evaluation Metrics

Speech quality assessment methods can be divided into subjective and objective evaluation methods. Subjective evaluation methods involve human participation and thus have relatively poor stability. Objective evaluation methods assess speech quality from two perspectives: one from the perspective of human auditory perception, where the most widely used method is the Perceptual Evaluation of Speech Quality (PESQ), and the other from the signal perspective, which includes the Source-to-Distortion Ratio (SDR) and its improved version, the Scale-Invariant Source-to-Distortion Ratio (SI-SDR).

(1) Perceptual Evaluation of Speech Quality (PESQ)

Table 1. MOS Subjective Evaluation Rating Scale

Level	Score	Attention Level
Excellent	5	Complete relaxation of attention
Good	4	No deliberate attention needed
Normal	3	Moderate attention required
Poor	2	Focused attention required
Bad	1	Unable to hear clearly

PESQ uses the speech loss model defined in ITU standard G.107 to assess speech quality. The model receives both clean and distorted speech signals, processes them through a series of filtering and time-alignment operations, and calculates two distortion parameters. These distortion parameters are then mapped to the Mean Opinion Score (MOS). The range of PESQ values is from -0.5 to 5, corresponding to the MOS subjective evaluation ratings shown in Table 1.

(2) The Source-to-Distortion Ratio (SDR) and the Scale-Invariant Source-to-Distortion Ratio (SI-SDR)

SDR is defined as the ratio of the power of the target signal to the power of the noise present in the estimated signal. This measure gives an indication of the strength of the desired signal in relation to unwanted noise.

SI-SDR, on the other hand, first involves normalizing the signals to reduce the effects of signal level changes before calculating the ratio of the signal power to the noise power. This scale-invariance is crucial for fair comparison of different systems where the absolute signal level might vary.

The specific calculation methods for these two metrics are as follows:

$$e_{noise}^{(1)} = \hat{s} - s \quad (13)$$

$$s_{target} = \frac{\langle \hat{s}, s \rangle s}{||s||^2} \quad (14)$$

$$e_{noise}^{(2)} = \hat{s} - s_{target} \quad (15)$$

$$SDR = 10 \log_{10} \frac{||s||^2}{||e_{noise}^{(1)}||^2} \quad (16)$$

$$SI - SDR = 10 \log_{10} \frac{||s_{target}||^2}{||e_{noise}^{(2)}||^2} \quad (17)$$

In this context, $\langle \hat{s} \rangle$ and $\langle s \rangle$ represent the estimated speech by the network and the target clean speech, respectively. It is important to note that both $\langle \hat{s} \rangle$ and $\langle s \rangle$ are processed to have zero mean.

Finally, the model's performance for speech separation is evaluated using the gain forms of the three aforementioned methods, namely PESQi, SDRi, and SI-SDRi, to analyze the performance of the model. Additionally, the objective function chosen for training the multi-channel target direction speech separation model is Scale-Invariant Source-to-Distortion Ratio (SI-SDR).

$$\mathcal{L}(\Theta|s, \hat{s}) = -SI - SDR(s, \hat{s}) \quad (18)$$

The objective function, as described above, uses Θ to represent the model's parameters. Because a higher SI-SDR indicates greater similarity between the estimated speech and the target speech, the loss function is taken as the negative value. This approach incentivizes the model to minimize the loss by maximizing the SI-SDR, thus improving the fidelity of the speech separation.

(3) Multi-channel Microphone Speech Dataset

To validate the effectiveness of the proposed model, this paper uses the LibriSpeech dataset [14], specifically the train-clean-100 training set, the dev validation set, and the test set, along with the WHAM noise dataset [15], to create a multi-channel speech dataset. The LibriSpeech dataset consists of clean speech recordings from a single microphone. To meet the conditions for multi-channel audio, this experiment uses the open-source library Pyroomacoustics to generate multi-channel speech in an indoor environment and simulate a microphone array to collect the audio. In the simulation, the room height is set between 3m to 4.5m, and the width and length are randomly selected between 5m to 10m. The reverberation time of the room is set to not exceed 0.5 s and not less than 0.1 s.

The microphone array configuration consists of a linear array of four omnidirectional microphones with a sampling rate of 16kHz. The microphones are evenly spaced, with a 0.1m interval between each. The configuration of the sound sources involves selecting two speakers speaking simultaneously inside the room. Similar to previous setups, it is assumed that both speakers are on the same horizontal plane as the microphone array and are located on one side of the linear microphone array, with distances from the microphone center ranging between 1.5m to 3m. The noise is added to the mixed speech with a random Signal-to-Noise Ratio (SNR) ranging from 0 dB to 20 dB. Finally, the duration of each speech recording is controlled to be 3 s, equivalent to 48,000 samples.

The information about the obtained training set, validation set, and test set is shown in Table 2.

Table 2. Multi-Channel Mixed Speech Dataset Information

Dataset	Number of Samples	Number of Speakers	Duration (Hours)
Training	13,900	251	58
Validation	3,000	40	11
Testing	3,000	40	11

(4) Experimental Setup

Table 3. TSS-UConvNet Hyperparameter Settings

Parameter Symbol	Value	Description
C_ε	512	Number of channels in time-domain encoder
K_ε	40	Receptive field of time-domain encoder
C_M	128	Number of channels in 2D convolution
NFFt	64	Number of points in STFT transformation
C	128	Number of input channels in U-ConvBlock
C_U	512	Number of expanded channels in U-ConvBlock
K_U	10	Receptive field of depth-wise separable convolution
S_U	5	Stride of depth-wise separable convolution
Q	4	Number of up-sampling/down-sampling operations
B	16	Number of U-ConvBlocks
U	2	Up-sampling factor

The experimental environment is set up on a deep learning server. The server is equipped with an Nvidia RTX 3090 graphics card with 24GB of VRAM and an Intel(R) Xeon(R) Platinum 8358P CPU. The deep learning framework used is Pytorch, which is currently the most popular framework.

During the comparative experiments, two benchmark models were selected: one is a single-channel speech separation model based on the U-shaped convolutional block network called SuDoRM-RF, and the other is a multi-channel speech separation model, FasNet [56]. The experimental configuration for the comparative models uses the best settings reported in their respective literature.

For each TSS-UConvNet model and its variants, the training settings are a batch size of 16 and 100 epochs. Validation is performed after each training epoch using the validation set. The Adam optimizer is used with an initial learning rate of 0.001,

which is halved if there is no decrease in the loss function on the validation set for 5 consecutive epochs. Finally, the model with the smallest loss on the validation set is used to test the test set to determine the separation effectiveness of the model.

Unless otherwise specified in the following text, the default hyperparameters for the models are as outlined in Table 3.

5 Experimental Results and Analysis

Experiment 1: In order to explore the impact of various spatial features on model performance, an ablation study was conducted. This involved maintaining all other aspects constant and sequentially changing the input spatial features. This allowed for an analysis of how spatial features—namely Angular Features (AF), Cosine Inter-channel Phase Difference (cosIPD), Sine Inter-channel Phase Difference (sinIPD), and Multi-Channel Speech (MCS)—affect the model’s effectiveness.

Table 4. Results of Directional Feature Ablation Experiment

Feature Configuration	SI-SDRi (dB) (0°, 15°]	SI-SDRi (dB) (15°, 45°]	SI-SDRi (dB) (45°, 90°]	SI-SDRi (dB) (90°, 180°]	AVG
+ AF	11.06	15.45	17.38	19.09	16.66
+ AF, cosIPD	16.36	21.56	22.35	23.89	21.97
+ AF, cosIPD, sinIPD	16.45	22.58	23.27	24.10	22.66
+ AF, cosIPD, sinIPD, MCS	18.17	23.32	23.67	25.71	23.59

As shown in Table 4, the TSS-UConvNet model that only includes the target angle feature AF and the temporal feature W achieves an improvement of nearly 17dB in SI-SDR (Scale-Invariant Source-to-Distortion Ratio). Meanwhile, the cosine inter-channel phase difference (cosIPD) feature contributes an additional improvement of 5.31dB in SI-SDR, especially when the distance between speakers is within a small angular range, contributing approximately 5.3dB. In contrast, the sine inter-channel phase difference (sinIPD) feature provides a minimal increase in SI-SDR of less than 1dB, possibly because the cosIPD and sinIPD features both originate from the inter-channel phase difference (IPD) and share overlapping spatial information. Lastly, the multi-channel convolution and MCS features learned by the two-dimensional convolution layer contribute only 0.93dB improvement, which is not particularly significant.

Experiment 2: To explore the impact of U-shaped convolutional blocks on the model’s performance, two sets of control experiments were set up. In the first set, the U-shaped convolutional block network in the TSS-UConvNet model’s separator was replaced with a temporal convolution network (TCN), resulting in the multi-channel target direction speech separation model TSS-TCN, which was then compared with the TSS-UConvNet model to investigate the effectiveness and performance of U-shaped convolutional networks in multi-channel target direction separation models. The second set of experiments

was conducted to explore the impact of the number of stacked U-shaped convolutional blocks (B) on the separation effectiveness, with B set to 4, 8, 16, and 32 for comparison.

As illustrated in Table 5, it is evident that the U-shaped convolutional block network is more suitable than the temporal convolution network (TCN) for acting as a separator in multi-channel target direction speech separation models, demonstrating the efficiency of U-shaped convolutional blocks in learning temporal information. Even with a larger number of model parameters, the TSS-TCN model does not enhance speech separation performance as effectively as the TSS-UConvNet, which has fewer parameters.

Table 5. Impact of U-Shaped Convolutional Blocks on Model Performance

Model	SI-SDRi (dB)	SDRi (dB)	PESQi	Parameter Size (MB)
TSS-TCN	17.64	5.61	0.09	8.5
TSS-UConvNet (B = 32)	23.11	5.94	0.23	4.9
TSS-UConvNet (B = 16)	23.59	6.76	0.29	2.6
TSS-UConvNet (B = 8)	21.91	5.68	0.22	1.4
TSS-UConvNet (B = 4)	20.36	4.32	0.14	0.8

The comparison of the number of stacked U-shaped convolutional blocks (B) reveals that even with only 4 layers, the improvement in SI-SDR reaches 20.36 dB, SDR increases by 4.32 dB, and the PESQ score rises by 0.14. At this configuration, the parameter size of the TSS-UConvNet is only 0.8 MB, further demonstrating the efficiency of the U-shaped convolutional block network. Notably, when B is increased to 32, there is a decline in speech separation metrics, which might be attributed to the network's excessive depth leading to a loss of spatial feature information. Therefore, to balance the model's parameter size and performance, the default setting for B in this paper is 16.

Experiment 3: This experiment is a comparative study, benchmarking against the single-channel speech separation model SuDoRM-RF and the multi-channel speech separation model FasNet mentioned earlier. In this experiment, the number of U-shaped convolutional blocks in the target speech separation model TSS-UConvNet is set to 8.

Table 6. Comparison of Speech Enhancement Performance Across Models

Model	SI-SDRi (dB)	SDRi (dB)	PESQi	Parameter Size (MB)
SuDoRM-Rf	16.03	0.23	0.02	2.6
FasNet	18.09	7.62	0.15	1.5
TSS-UConv (B = 8)	21.91	5.68	0.22	1.4

From Table 6, it is evident that the proposed target direction speech separation model, TSS-UConvNet (B = 8), outperforms the single-channel speech separation

model SuDoRM-Rf across all evaluation metrics. This superiority primarily stems from the spatial information provided by multi-channel audio, which plays a decisive role. Compared to the multi-channel speech separation model FasNet, even though FasNet shows higher improvement in Signal-to-Noise Ratio (SDR), the enhancements in Scale-Invariant Signal-to-Distortion Ratio (SI-SDR) and Perceptual Evaluation of Speech Quality (PESQ) are not as substantial as those achieved by the TSS-UConvNet model. Specifically, the TSS-UConvNet model exhibits a 3.82 dB higher SI-SDR and a 0.07 higher PESQ score than FasNet.

6 Conclusion

This paper first introduced a multi-channel target direction speech separation model and then proposed an efficient U-shaped convolutional block network for learning temporal information. By integrating this network with prior knowledge of speaker directions, the U-shaped convolutional block network was extended to the proposed multi-channel target direction speech separation model, TSS-UConvNet. To validate the effectiveness of the proposed model, a multi-channel mixed speech simulation dataset was constructed for testing, and the impact of various spatial features on speech separation performance was explored. The experiments demonstrated that the U-shaped convolutional block network is more effective in learning temporal information for multi-channel speech separation compared to the Temporal Convolution Network (TCN). A comprehensive comparative analysis confirmed the significant speech separation effectiveness of the proposed multi-channel target direction speech separation model, TSS-UConvNet. Our model has some limitations while it is applied in certain scenarios. Since the network is based on prior knowledge of speaker directions.

References

1. Tesch, K., Gerkmann, T.: Nonlinear spatial filtering in multichannel speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **29**, 1795–1805 (2021)
2. Heymann, J., Drude, L., Haeb-Umbach, R.: Neural network based spectral mask estimation for acoustic beamforming. In: *IEEE International Conference on Acoustics*. IEEE (2016)
3. Xiao, X., Xu, C., Zhang, Z., et al.: A study of learning based beamforming methods for speech recognition. In: *CHiME 2016 Workshop*, pp. 26–31 (2016)
4. Ochiai, T., Delcroix, M., Ikeshita, R., et al.: Beam-TasNet: time-domain audio separation network meets frequency-domain beamformer. In: *ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 6384–6388 (2020)
5. Heymann, J., Drude, L., Chinaev, A., et al.: BLSTM supported GEV beamformer front-end for the 3rd CHiME challenge. In: *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, pp. 444–451 (2015)
6. Chen, H., Yi, Y., Feng, D., et al.: Beam-Guided TasNet: An Iterative Speech Separation Framework with Multi-Channel Output. *arXiv preprint [arXiv:2102.02998](https://arxiv.org/abs/2102.02998)* (2021)
7. Luo, Y., Han, C., Mesgarani, N., et al.: FaSNet: LOW-latency adaptive beamforming for multi-microphone audio processing. In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, pp. 260–267 (2019)

8. Luo, Y., Mesgarani, N.: Implicit Filter-and-sum Network for Multi-channel Speech Separation (2020)
9. Luo, Y., Chen, Z., Mesgarani, N., et al.: End-to-end microphone permutation and number invariant multi-channel speech separation. In: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 6394–6398 (2020)
10. Wang, Z.Q., Wang, D.L.: Combining spectral and spatial features for deep learning based blind speaker separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **27**(2), 457–468 (2018)
11. Gu, R., Chen, L., Zhang, S.X., et al.: Neural spatial filter: target speaker speech separation assisted with directional information. In: Interspeech, pp. 4290–4294 (2019)
12. Gu, R., Zou, Y.: Temporal-Spatial Neural Filter: Direction Informed End-to-end Multi-channel Target Speech Separation. arXiv preprint [arXiv:2001.00391](https://arxiv.org/abs/2001.00391) (2020)
13. Tzinis, E., Wang, Z., Smaragdis, P.: Sudo rm-rf: Efficient networks for universal audio source separation. In: 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, pp. 1–6 (2020)
14. Panayotov, V., Chen, G., Povey, D., et al.: Librispeech: an asr corpus based on public domain audio books. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp. 5206–5210 (2015)
15. Wichern, G., Antognini, J., Flynn, M., et al.: Wham!: Extending Speech Separation to Noisy Environments. arXiv preprint [arXiv:1907.01160](https://arxiv.org/abs/1907.01160) (2019)



Secure Cloud-Edge Collaborative Task Scheduling Framework Across Data Centers

Ming Wan, Yuxiang Qiu, Yuan Zhu^(✉), and Lan Gan

Nari Information and Communication Technology Co., Ltd., Nanjing, China
{wanming,qiuyuxiang,zhuyuan,ganlan}@sgepri.sgcc.com.cn

Abstract. The growing prevalence of edge computing and its more profound incorporation into cloud computing has given rise to the advent of the multi-center scenarios. This presents a significant challenge in the context of task scheduling. However, the existing task scheduling systems either designs for single cloud/edge center or several edge servers. Few systems support cross-center cloud-edge collaborative scheduling. In this work, we propose a secure cloud-edge collaborative task scheduling system that can generate scheduling plans across multiple data centers. The system supports time-sharing scheduling, multitask scheduling and task offloading between centers at the same time. It generates scheduling plans with the priority-based time-sharing policy, and adopts the deep-Q-network-based multitask policy and the dynamic task offloading policy. The experimental results indicate that the scheduling delay increases linearly with the number of concurrent submitted tasks. The scheduling delay is little affected by the number of data centers.

Keywords: Edge Computing · Task Scheduling · Deep Q-Network · Priority-based Scheduling · IoT Security

1 Introduction

With the quick development of cloud computing and edge computing technology, cloud data centers and edge servers are breaking their boundaries and integrating with each other. Cloud-edge collaborative computing is becoming the new trend in the field of edge computing. It can improve the computing efficiency of real-world applications like health care [1, 13], power system information management [29] and manufacturing [3]. The cloud-edge collaborative task scheduling system, as the underlying basic functional platform, is facing increasingly complex and diverse task scheduling requirements. It becomes an urgent research problem of how to improve the quality of service (QoS) of tasks as much as possible on the basis of the efficient collaboration between cloud and edges, maximizing the usage of the computing resources, and meeting the diversified needs of tasks.

For example, in the power information system, the video monitoring application for transmission lines is a typical application that requires collaborative scheduling of both cloud and edge. The application needs to simultaneously launch two tasks. One of the task runs at the edge side that is close to the monitor, executing real-time video stream compression and key object detection algorithms. The other task runs in the central cloud which is responsible for filtering monitoring results and store key results in the database.

The widely adopted cloud computing scheduling system Kubernetes [2] and its derivative edge computing scheduling system KubeEdge [22] are mainly designed for single data center environments but do not support cross-center scheduling of tasks. The academic prototyping cloud-edge scheduling systems [5, 11, 28] support cross-center scheduling but they seldom consider the priority-based time-sharing scheduling requirements needed in real-world scenarios.

In this work, we propose and implement a secure cross-center cloud-edge collaborative task scheduling system. The system can generate cloud-edge collaborative scheduling plans across multiple data centers. We propose a priority-based time-sharing scheduling method. The scheduler also adopts the deep Q-network-based multitask scheduling policy and the existing task offloading scheduling method simultaneously to generate high quality scheduling plans. The scheduling system also adopts TLS transmission to guarantee the scheduling security. The experimental results indicate that the scheduling delay are almost unaffected by the number of data centers. The scheduling delay increases linearly with the number of tasks.

2 Related Work

2.1 Task Scheduling in Cloud-Edge Collaborative Computing

Depending on the scheduling goals, resource schedule methods for edge computing focuses on two key problems [14]: resource allocation and task offloading. Resource allocation in cloud-edge computing is to assign tasks to clouds and edge servers to maximize QoS or minimize penalty. The goal of task offloading [15] is to adjust the execution location of tasks if the computation resources are limited in some servers and offloading heavy tasks from these servers to clouds that have rich computation resources.

The resource allocation methods for edge computing need to consider the computation, storage, and communication resources of both edge servers and cloud servers during the task scheduling [4, 9]. The communication resource scheduling involves bandwidth occupation, transmission power consumption [6, 7] and channels [8]. Xu et al. [25] considers the computation resource allocation to reduce the execution time of tasks. The storage resources can also be properly allocated to cache frequently-used data to reduce the execution costs on accessing data from remote clouds [12, 24, 26]. Xu et al. [24] propose the online dynamic service caching algorithms to reduce data transmission. Yu et al. [26] considers the sharing of computation results during task offloading to reduce the overall execution delay.

The existing task offloading can be further categorized into full offloading [10, 17, 19, 29] and partial offloading [20, 21, 23, 27]. The full offloading methods work in a binary manner. If a task is decided to offload, it is offloaded on the whole to a remote server. Materwala et al. [10] focus on optimizing energy consumption of task offloading while Zhang et al. [29] uses two-mode switching to get high quality offloading decision even at the early-stage of offloading.

On the contrary, if a task can be divided into smaller sub-tasks, the partial offloading methods can partially offload some sub-tasks to reduce overheads brought by offloading. Wu et al. [21] implements secure task offloading with the help of blockchain. Yu et al. [27] proposes a partial task offloading method for UAV-assisted edge computing scenario.

However, the existing cloud-edge task scheduling methods typically optimize execution efficiency-related metrics such as task failure rate, task delay, and resource utilization. They rarely consider task priority and time-sharing scheduling which are common business-related scheduling factors in real-world practical power applications. Therefore, a practical cloud-edge collaborative scheduling system cannot just adopt a single scheduling method. The system should comprehensively adopt scheduling strategies at different levels such as time-sharing scheduling, multi-objective task orchestration, and task offloading.

2.2 Resource Management Systems for Edge Computing

Kubernetes [18] is an open source system for automated deployment of cloud-native applications, supporting the scheduling of cluster resources by lightweight containers. It provides a complete open source solution for container orchestration management. Kubernetes is originally designed for the cloud environment, its core components deployed on each computing nodes are still heavy for edge servers. KubeEdge [22] is an open source edge-side containerized resource scheduling system that extends native containerized application orchestration and management to edge devices. It is built on top of Kubernetes to provide compatible APIs. However, KubeEdge redevelops the node agent, and significantly optimizes the edge components to take up much less resources. It also optimizes message channels between devices and edge servers with various IoT communication protocols like MQTT.

Besides industry-level systems, DeEdge [11] provides an edge computing test bed that can simulate the communication latency in real-world edge computing scenarios. ENTS [28] is an academic prototyping edge-native system. ENTS extends Kubernetes with advanced scheduling strategy that takes the inner dependencies, data locality and network resource limitation into account. FLEDGE [5] adopts Virtual Kubelets to reduce the resource consumption of the scheduling system, and it can run on resource limited devices like Raspberry Pi 3.

Table 1 compares the scheduling features of the existing systems. There are almost no systems focused on coordinating computing tasks across multiple cloud-edge centers. Currently, research on collaborative multi-center task scheduling systems for the cloud-edge environment is still in its early stages.

Our system can schedule tasks across multiple cloud-edge data centers and also support time-sharing, multitask scheduling and task offloading at the same time.

Table 1. Comparison of Cloud-Edge Resource Management Systems

System	Scheduling Scope	Time-sharing Scheduling	Multitask Scheduling	Task Offloading
Kubernetes	Single Cloud Center	Yes	Yes	No
KubeEdge	Single Edge Center	Yes	Yes	No
DeEdge	Edge Servers	No	Yes	Yes
ENTS	Edge Servers	No	Yes	No
FLEDGE	Edge Centers	Yes	Yes	No
Our	Cloud/Edge Centers	Yes	Yes	Yes

3 System Design

In this paper, we design a cloud-edge collaborative task scheduling system that can allocate tasks among multiple cloud centers and multiple edge servers. The system adopts a container-based resource abstraction and generates schedule decision with three steps.

3.1 Container-Based Resource Abstraction

We adopt the container as the unit of resource allocation for computation and memory storage, which is widely supported in cloud-edge collaborative scheduling [16]. Each container requires a fixed number of CPU cores and memory space. The container runtime daemon (such as Docker or podman) requests the resources from the operating system. If the current available resources cannot meet the requirement of a task, the container daemon fails to launch the container and the system regards the task as failed.

The system uses containers to represent the execution unit of tasks. Each execution of a task corresponds to a container. The system also includes the following fields to describe the task scheduling requirement:

- The priority level of a task indicates the latency sensitivity of the task. The supported priority levels in the systems include the real-time task, the quasi-real-time task, and the batch task. Real-time tasks are very sensitive to the scheduling latency, and it should be executed as soon as possible. On the contrary, batch tasks have relaxed deadlines. The execution of batch tasks can be delayed until midnight or even for several days.
- The preference location of a task implies the location of the data that the task processes.

- The deadline of a task is the time point that the task must be finished. If the task is not finished at the time of deadline, the task is regarded as failed.
- For each task, a resource requirement vector is used to represent the maximum demand for multiple resources of the task, including resources (e.g., number of CPU cores, the memory space, the disk space, and the network bandwidth).

3.2 System Architecture

The core components of the system are shown in Fig. 1 with color background. The interaction interface component includes task submission interface, task execution status monitoring interface, the task execution history interface and the management interface. Those interfaces expose APIs to users and users can submit, execute and monitor tasks of the system.

The cloud-edge resource collaborative scheduler component is the core of the system. It has a three-layer architecture.

- The first layer is the scheduling service that consists of single task scheduling service and multitask scheduling service. The single task scheduling service consists of three parts: the task feature profile, the time-sharing task scheduling, and the task offloading strategy. The multitask scheduling service consists of a task performance prediction module and a multitask scheduling strategy module.
- The second layer consists of two schedulers. The center task execution scheduler runs in the cloud center. It uses an event listener to trigger task scheduling and calls the scheduling services to make scheduling decisions. The local task scheduler runs in each cloud center or edge server to execute tasks with containers. It is also responsible for track the execution status of tasks.
- The third layer is the resource management module, which consists of three parts: resource pool, resource meta-information manager, and the resource usage monitoring module.

The platform support component includes four modules: service monitoring, log service, configuration management, and scheduling solution quality monitoring. The service monitoring module tracks the status of each module in the scheduler component. The log service stores all the logs generated by different modules in a single repository. The scheduling plan quality monitoring module evaluates the quality of the scheduling plans generated by the scheduler according to the task execution logs.

3.3 System Deployment

Figure 2 shows how the cloud-edge collaborative scheduling system. The entire system is deployed over multiple data centers, including both cloud data

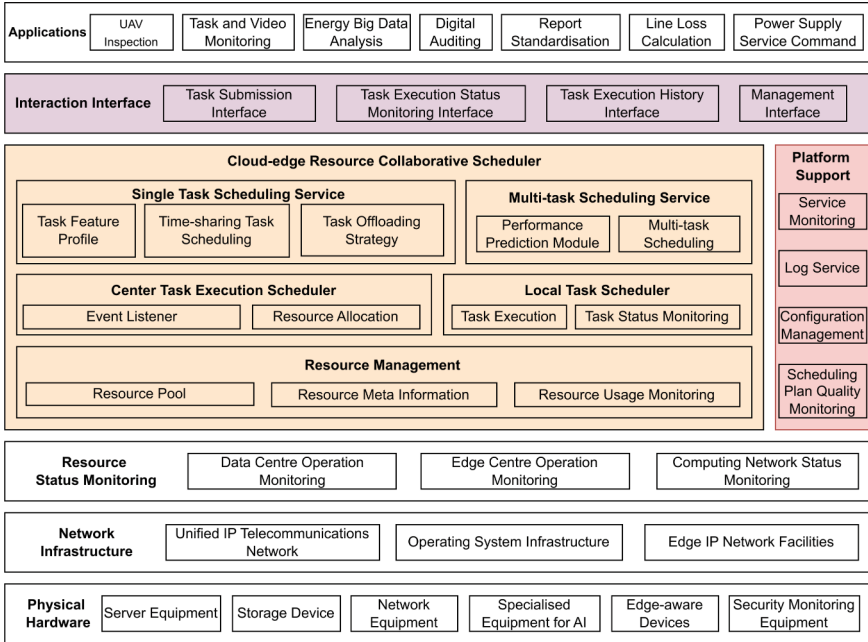


Fig. 1. Architecture of the proposed cloud-edge collaborative task scheduling system.

centers and edge data centers. A cluster resource management framework (e.g., Kubernetes or KubeEdge) is run within each data center. The system schedules user-submitted computation tasks to the appropriate data center for execution.

The system starts a central scheduler in a given cloud server. The central scheduler is responsible for receiving computational tasks from users, making task scheduling decisions and sending the tasks to specific data centers for execution. The central scheduler runs all the system components shown in Fig. 1 except the local task scheduler.

For each cloud/edge data center, the system runs a local scheduler which will execute the local task scheduler module of the system. The central scheduler and the local schedulers are connected to each other through the network. After making decisions, the central scheduler sends task description to local schedulers. Every local scheduler communicates with the corresponding cluster resource manager within the cloud/edge data center. The cluster resource manager actually starts and manages the execution of tasks. The local scheduler of the system tracks the execution of the tasks via the resource manager and reports the status to the central scheduler periodically.

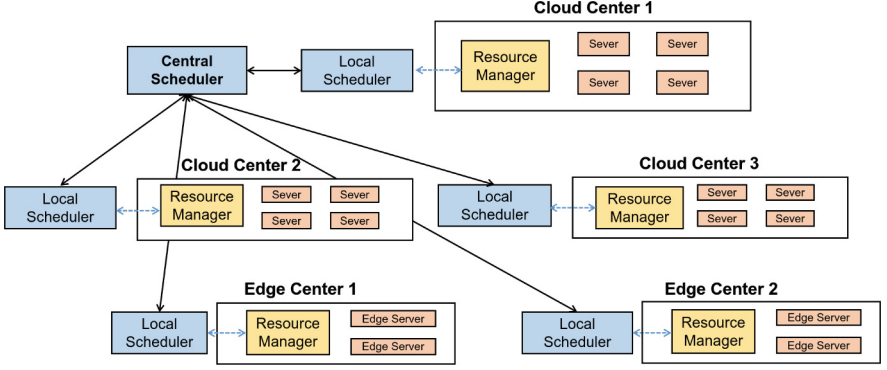


Fig. 2. Scheduling components deployment across multiple data centers. Each center is deployed among a local scheduler.

4 Cross-Center Collaborative Task Scheduling

The proposed collaborative task scheduling system can schedule tasks across multiple data centers including both cloud and edge centers. It proposes a cloud-edge collaborative scheduling process and introduces a time-sharing scheduling method based on priority queues, which supports delaying some tasks to off-peak periods based on the priority information of the computing tasks.

4.1 Scheduling Workflow

Figure 3 shows the scheduling workflow of the system. The arrows in the figure indicates the order of the scheduling steps in our system. The system continuously receives tasks from the job clients.

When a computing task is reached, its first handed over to the task feature extraction module of the single-task scheduling service to generate a task portrait for that computing task. The task portrait contains task priority, task scheduling preference, task request resource request, and generates feature information such as task history resource actual occupancy and task history average execution time based on the task execution history. The task request and its feature portrait will be handed over to the time-phased task scheduling module, which assigns expected time slices to the task and puts it in the priority queue.

The multitask scheduling services triggers scheduling with time slices (i.e., fixed time intervals). In each time slice, the scheduling service retrieves all tasks waiting for scheduling execution in that time slice. It generates a candidate scheduling plan using a deep Q network (i.e., assigning different tasks to appropriate cloud or edge data centers). For a candidate scheduling plan, the scheduling service calls the performance estimation module to predict the execution quality of the execution plan (e.g., task failure rate) and employs existing gradient descent-based scheduling optimization methods to optimize the candidate plan. The process is repeated until the plan reaches a local optimum.

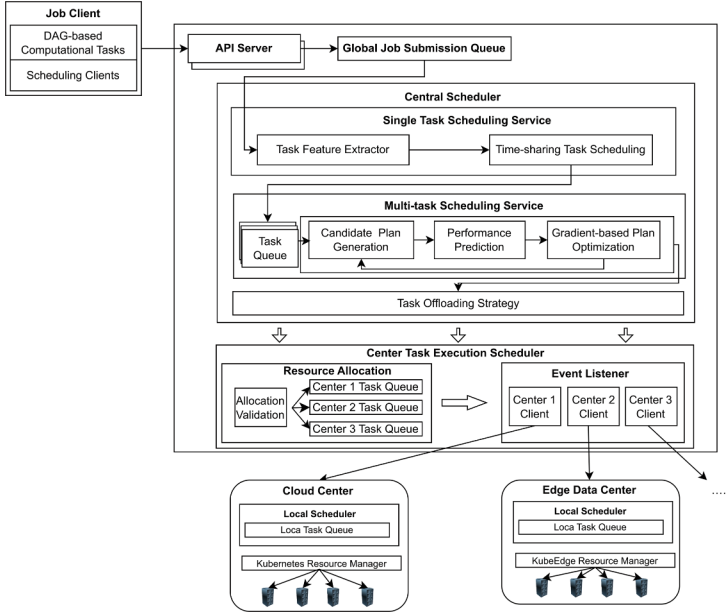


Fig. 3. Scheduling workflow of the cloud-edge collaborative task scheduling framework. The central scheduler uses two services and a policy to generate the scheduling decision.

Because the multitask scheduling service does not consider the resource limitation of each data center when optimizing the scheduling scheme, there is a possibility of task overload, i.e., the resources requested by the assigned tasks of a cloud/edge data center exceeds the upper limit of the currently available resources of that data center. For the tasks that may be overloaded and lead to failure, the system will fine tune the task scheduling plan with the task offloading strategy and transfer those tasks to data centers that still have available resources and meet the data affinity requirements.

The fine-tuned scheduling plan is the execution plan for this time slice. The scheduling plan will be submitted to the center task execution scheduler, which will further verify the validity of the tasks and assign them to the task queues of different data centers. The tasks in the task queue will be pushed down to the local executor modules deployed in different data centers through the client connection of the event listener module. The local executor will create a task execution object for each task and submit it to the cluster's resource manager (e.g. Kubernetes or KubeEdge) to initiate the execution process of the task.

To ensure scheduling safety, the network connections between the central scheduler and the local schedulers of each data center uses TLS encryption, ensuring the reliability and confidentiality of scheduling plan transmission, and guaranteeing the security of the scheduling plan. The central scheduler generates and distributes TLS certificates to all local schedulers. Each local scheduler actively establishes a connection with the central scheduler in a TLS encrypted

manner and maintains this connection during the whole process. The scheduling instructions sent from the central scheduler to the local schedulers, as well as the task execution status and resource usage statistics reported by the local schedulers to the central scheduler, are protected by TLS during transmission. No third party can access or tamper with the aforementioned critical scheduling information between central scheduler and the local schedulers, guaranteeing the security of the scheduling instructions and the scheduling process.

4.2 Priority-Queue-Based Time-Sharing Scheduling

In cloud-edge co-scheduling environments, task queues are usually in a ‘first-in-first-out’ (FIFO) mode, meaning that tasks are executed sequentially in the order in which they arrive. However, this simple scheduling strategy may not be able to satisfy the needs of all tasks, especially those with high real-time requirements. In FIFO mode, if a task waits too long in the queue, even if its execution time is short, the task may time out due to the long waiting time, which is unacceptable for business scenarios that require fast response.

Based on this problem, this framework proposes an intelligent time-sharing scheduling strategy based on priority queuing for task profiling. The basic structure is shown in Fig. 4. The core idea of this strategy is to differentially priorities tasks according to their different real-time requirements. This priority ranking not only takes into account the arrival time of the tasks, but also other factors such as the task’s preservation resource requirements, quality of service (QoS) guarantee requirements, and the task’s due time. Specifically, the task queue in each data center will no longer simply execute tasks in ‘first-come, first-served’ order. Instead, tasks will be reordered according to their priority. This priority is dynamically determined based on the characteristics of the task and business requirements. For example, tasks with shorter due times and high QoS requirements are given a higher priority and moved forward in the queue to be executed faster.

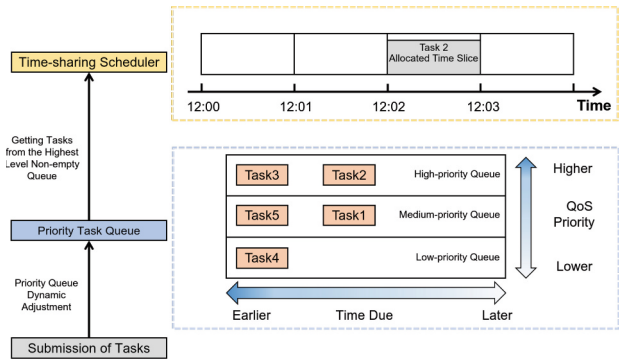


Fig. 4. Time-sharing task scheduling is implemented by the priority queues.

In this model, there exists a task portrait management module in the system, which is able to collect and analyze the characteristic information of tasks, such as task type, resource demand, execution time, etc., and the intelligent time-sharing system dynamically adjusts the priority of tasks accordingly. In this way, the data center can more accurately predict the execution of tasks and make more reasonable scheduling decisions. We assume a continuous time slice t with a slice length of 1 min. At the beginning of each time slice, the system dynamically adjusts the priority of the tasks in the task priority queue according to the computational task portrait, and then allocates computational resources to the tasks based on the order of the tasks in the queue, and allocates a time slice of execution as much as possible for the tasks in the queue.

The intelligent time-sharing scheduling strategy based on priority queues provides a more flexible and efficient task allocation method for cloud-edge collaborative scheduling by taking into account the real-time requirements and business characteristics of tasks. This design aims to optimize resource allocation and improve the efficiency and responsiveness of task execution. By prioritizing tasks with higher real-time requirements and moving tasks with lower real-time requirements to be executed when resources are idle, this strategy reduces the risk of task timeouts and ensures that critical tasks can be completed in a timely manner. At the same time, this approach provides data centers with greater flexibility to dynamically adjust to task-specific requirements and business priorities.

5 Experiments

To evaluate the performance of the proposed scheduling system, we simulate an edge computing scenario with a cluster consisting of four nodes. Each node in the cluster is configured with four CPU cores, 4 GB of memory capacity, and 40 GB of HDD hard disk storage. Each node in the cluster uses the openEuler 22.03 LTS Linux operating system. We set up a Kubernetes cluster with the version v1.20.2 in each node to simulate a data center. The nodes are connected via 10 Gbps virtual Ethernet with the latency less than 1 ms. The scheduling system is implemented with the Python language and executed with the Python interpreter version 3.9.9. Network communication between the central scheduler and the local schedulers is performed with ZeroMQ.

5.1 Scheduling Delay

This experiment focuses on evaluate the time required by the central scheduler to generate scheduling plans for tasks (i.e., the scheduling latency of tasks). The delay describes the computation costs of the central scheduler spent on the scheduling policies. The delay indicates the scalability of the central scheduler when many tasks arrive at the same time. We submit different numbers of tasks to the central scheduler at the same time and measure the time taken from the submission of the first task to the generation of scheduling plans for all tasks. Since we are interested in measuring the overheads related to the scheduler

itself, we submitted tasks with the same priority and resource requirements. The scheduling latency of the central scheduler is evaluated for different number of data centers in the experiments, and the experimental results are shown in Fig. 5.

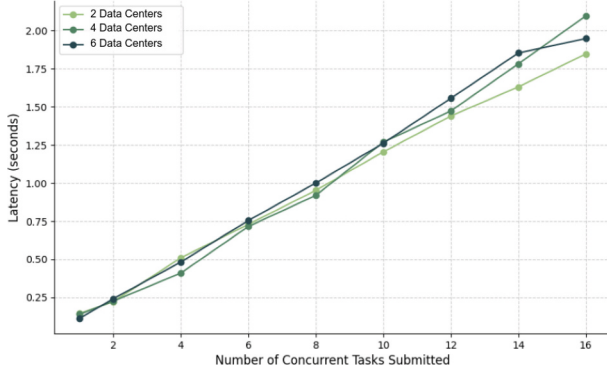


Fig. 5. Scheduling delay of the central scheduler grows linearly with the number of concurrent submitted tasks. The scheduling delay is slightly affected by the number of data centers.

The results indicate that the scheduling delay grows linearly with the growth of the number of tasks submitted at the same time. When 16 tasks are submitted at the same time, the tasks can be scheduled within 2.1 s. The proposed cloud-edge collaborative scheduling system has high execution efficiency. At the same time, the scheduling delay under different numbers of data centers is very close, indicating that the scheduling overhead is less affected by the number of data centers. It shows that the scheduling methods adopted by the system has good scalability.

5.2 Scheduling Quality

In order to measure the quality of the scheduling plans generated by the scheduler of the proposed system, two data centers (one cloud center and one edge center) are deployed in the experiment to simulate the cloud-edge co-scheduling scenario. The scheduling process mainly considers the limitation of the number of CPU cores, and each data center has 10 CPU cores for scheduling.

The computation workloads simulate the collaborative task scheduling requirements from a provincial power company. We submit 21 tasks simultaneously in the experiment, including 2 high-priority tasks, 8 medium-priority tasks, and 11 low-priority tasks. A high-priority task has an estimated execution time of 20 s, occupies 10 CPUs, and has a deadline of 500 s after the current time. A medium-priority task has an estimated execution time of 30 s, occupies 5 CPUs, and has a deadline of 400 s after the current time. A low-priority task

has an estimated execution time of 40s, and has a deadline of 45s after the current time, with each of the 10 tasks occupies 2 CPUs, and 1 task occupies 15 CPUs.

In the experiment, a task is considered as a timeout task if the execution of the task ended longer than the task deadline. If a task is a timeout task, it is also considered as a failure task. If a data center receives more tasks than its available resources, the latter submitted task is also considered to be a failure task. We measure the timeout and failure rates of tasks with different scheduling policies. The two metrics are related to the task failure (either caused by timeout or out of resource limitation). Users are usually more concerned with the successful execution of tasks, followed again by the efficiency of execution. Table 2 list the experimental results.

The proposed system adopts three scheduling strategies simultaneously (time-sharing scheduling + multitask scheduling + task offloading). The experimental results show that the adoption of the three scheduling strategies can effectively reduce the timeout rate and the failure rate of the tasks. If the system only adopts the time-sharing scheduling strategy without multitask scheduling, it would lead to a higher task failure ratio. The tasks that fail in experiments were usually low-priority tasks with high computational resource requirements. These tasks were usually pushed back by the time-sharing scheduling policy, but there were usually not enough computational resources available to complete them when they are scheduled for execution.

Table 2. Scheduling Quality under Different Policies

Scheduling Policy	Timeout Rate	Failure Rate
Random Allocation	47.6%	52.4%
Time-Sharing + Random	28.6%	33.3%
Time-Sharing + Multi-task	0%	4.8%
Time-Sharing + Multi-task + Task Offloading	0%	4.8%

6 Conclusion

Aiming at the cross-center collaborative scheduling of cloud and edge computing, we propose a cloud-edge collaborative scheduling system in this work. The system adopts a container-based resource abstraction. It can generate cross-center cloud-edge collaborative scheduling plans according to the priority-based time-sharing scheduling policy, the deep Q network-based multitask scheduling policy, and the task offloading policy. In the future, we consider adapting the framework to more distributed computing environments like the wireless edge computing environment by taking network metrics during the scheduling. We also consider adopt AI-driven optimization techniques to improve the quality of scheduling policies in the system.

Acknowledgments. This work was supported by the Key Technologies and Applications of Resource Collaborative Scheduling for Cloud-Edge-Network Integrated System Project of State Grid Corporation of China under Grant no.5700-202318292A-1-1-ZN.

References

1. Anil, B., Dayananda, P., Nethravathi, B., Raisinghani, M.S.: Efficient local cloud-based solution for liver cancer detection using deep learning. *Int. J. Cloud Appl. Comput. (IJCAC)* **12**(1), 1–13 (2022). <https://doi.org/10.4018/IJCAC.2022010109>
2. Carrión, C.: Kubernetes scheduling: taxonomy, ongoing issues and challenges. *ACM Comput. Surv.* **55**(7), 138:1–138:37 (2022). <https://doi.org/10.1145/3539606>
3. Chen, Z., Zhang, L., Wang, X., Wang, K.: Cloud-edge collaboration task scheduling in cloud manufacturing: an attention-based deep reinforcement learning approach. *Comput. Ind. Eng.* **177**, 109053 (2023)
4. Fan, W., Liu, X., Yuan, H., Li, N., Liu, Y.: Time-slotted task offloading and resource allocation for cloud-edge-end cooperative computing networks. *IEEE Trans. Mob. Comput.* **23**(8), 8225–8241 (2024). <https://doi.org/10.1109/TMC.2024.3349551>
5. Goethals, T., De Turck, F., Volckaert, B.: Extending kubernetes clusters to low-resource edge devices using virtual kubelets. *IEEE Trans. Cloud Comput.* **10**(4), 2623–2636 (2022). <https://doi.org/10.1109/tcc.2020.3033807>
6. Khalili, A., Zarandi, S., Rasti, M.: Joint resource allocation and offloading decision in mobile edge computing. *IEEE Commun. Lett.* **23**(4), 684–687 (2019). <https://doi.org/10.1109/lcomm.2019.2897008>
7. Kuang, Z., Li, L., Gao, J., Zhao, L., Liu, A.: Partial offloading scheduling and power allocation for mobile edge computing systems. *IEEE Internet Things J.* **6**(4), 6774–6785 (2019). <https://doi.org/10.1109/jiot.2019.2911455>
8. Li, L., Zhang, X., Liu, K., Jiang, F., Peng, J.: An energy-aware task offloading mechanism in multiuser mobile-edge cloud computing. *Mob. Inf. Syst.* **2018**, 1–12 (2018). <https://doi.org/10.1155/2018/7646705>
9. Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: a survey. *IEEE Commun. Surv. Tutorials* **23**(4), 2131–2165 (2021)
10. Materwala, H., Ismail, L., Shubair, R.M., Buyya, R.: Energy-SLA-aware genetic algorithm for edge-cloud integrated computation offloading in vehicular networks. *Futur. Gener. Comput. Syst.* **135**, 205–222 (2022). <https://doi.org/10.1016/j.future.2022.04.009>
11. Meng, J., Tan, H., Xu, C., Cao, W., Liu, L., Li, B.: Dedas: online task dispatching and scheduling with bandwidth constraint in edge computing. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE (2019). <https://doi.org/10.1109/infocom.2019.8737577>
12. Nikoloudakis, Y., et al.: Edge caching architecture for media delivery over P2P networks. In: *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. vol. 37, pp. 1–5. IEEE (2018). <https://doi.org/10.1109/camad.2018.8514935>
13. Pathoe, K., Rawat, D., Mishra, A., Arya, V., Rafsanjani, M.K., Gupta, A.K.: A cloud-based predictive model for the detection of breast cancer. *Int. J. Cloud Appl. Comput. (IJCAC)* **12**(1), 1–12 (2022). <https://doi.org/10.4018/IJCAC.310041>

14. Raeisi-Varzaneh, M., Dakkak, O., Habbal, A., Kim, B.S.: Resource scheduling in edge computing: architecture, taxonomy, open issues and future research directions. *IEEE Access* **11**, 25329–25350 (2023). <https://doi.org/10.1109/access.2023.3256522>
15. Ren, J., et al.: Collaborative task offloading and resource scheduling framework for heterogeneous edge computing. *Wireless Netw.* **30**(5), 3897–3909 (2024)
16. Tang, B., Luo, J., Obaidat, M.S., Vijayakumar, P.: Container-based task scheduling in cloud-edge collaborative environment using priority-aware greedy strategy. *Clust. Comput.* **26**(6), 3689–3705 (2023)
17. Tang, M., Wong, V.W.: Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Trans. Mob. Comput.* **21**(6), 1985–1997 (2022). <https://doi.org/10.1109/tmc.2020.3036871>
18. The Linux Foundation : Kubernetes. <https://kubernetes.io/>. Accessed 15 Aug 2024
19. Tong, Z., Ye, F., Mei, J., Liu, B., Li, K.: A novel task offloading algorithm based on an integrated trust mechanism in mobile edge computing. *J. Parallel Distrib. Comput.* **169**, 185–198 (2022). <https://doi.org/10.1016/j.jpdc.2022.07.006>
20. Wang, J., Zhang, J., Liu, L., Zheng, X., Wang, H., Gao, Z.: Utility maximization for splittable task offloading in IoT edge network. *Comput. Netw.* **214**, 109164 (2022). <https://doi.org/10.1016/j.comnet.2022.109164>
21. Wu, H., Wolter, K., Jiao, P., Deng, Y., Zhao, Y., Xu, M.: EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing. *IEEE Internet Things J.* **8**(4), 2163–2176 (2021). <https://doi.org/10.1109/jiot.2020.3033521>
22. Xiong, Y., Sun, Y., Xing, L., Huang, Y.: Extend cloud to edge with kubeedge. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 373–377 (2018). <https://doi.org/10.1109/SEC.2018.00048>
23. Xu, F., Xie, Y., Sun, Y., Qin, Z., Li, G., Zhang, Z.: Two-stage computing offloading algorithm in cloud-edge collaborative scenarios based on game theory. *Comput. Electr. Eng.* **97**, 107624 (2022). <https://doi.org/10.1016/j.compeleceng.2021.107624>
24. Xu, J., Chen, L., Zhou, P.: Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. IEEE (2018). <https://doi.org/10.1109/infocom.2018.8485977>
25. Xu, X., et al.: A computation offloading method over big data for IoT-enabled cloud-edge computing. *Futur. Gener. Comput. Syst.* **95**, 522–533 (2019). <https://doi.org/10.1016/j.future.2018.12.055>
26. Yu, S., Langar, R., Fu, X., Wang, L., Han, Z.: Computation offloading with data caching enhancement for mobile edge computing. *IEEE Trans. Veh. Technol.* **67**(11), 11098–11112 (2018). <https://doi.org/10.1109/tvt.2018.2869144>
27. Yu, X.Y., Niu, W.J., Zhu, Y., Zhu, H.B.: UAV-assisted cooperative offloading energy efficiency system for mobile edge computing. *Digital Commun. Netw.* **10**(1), 16–24 (2024). <https://doi.org/10.1016/j.dcan.2022.03.005>
28. Zhang, M., Cao, J., Yang, L., Zhang, L., Sahni, Y., Jiang, S.: ENTS: an edge-native task scheduling system for collaborative edge computing. In: 2022 IEEE/ACM 7th Symposium on Edge Computing (SEC). IEEE (2022). <https://doi.org/10.1109/sec54971.2022.00019>
29. Zhang, X., Duan, J., Yan, M., Lyu, S.: Task Offloading with Dual-Mode Switching in Multi-access Edge Computing, pp. 155–168. Springer Nature Singapore (2024). https://doi.org/10.1007/978-981-97-1332-5_13



Research on Security Sharing Mechanism of Cyber Threat Intelligence Based on Consortium Blockchain

Fei Wang^(✉)

Gansu University of Political Science and Law, Cyberspace Security Academy, Lanzhou City,
Gansu Province, China
297419859@qq.com

Abstract. In view of this, this paper adopts the consortium blockchain as the underlying framework, and proposes a network threat intelligence security sharing mechanism based on the consortium blockchain, which can avoid the security problems of the centralized sharing mechanism. This mechanism includes two sub-mechanisms: consensus mechanism improvement and sensitive intelligence sharing strategy, which respectively solve the data security problem during sharing and the privacy protection problem during sensitive intelligence sharing, and further improve the security of network threat intelligence sharing in the alliance blockchain. The main work of this paper is as follows:

- (1) Sort out the overall design of the sharing mechanism, including the underlying basic architecture, the composition of nodes in the network, the threat intelligence sharing process, data security and privacy protection. In terms of data security and privacy protection, while using PKI technology and digital certificates, the consensus mechanism is improved and a sensitive intelligence sharing strategy is proposed to ensure the data security and privacy protection of the system from the two levels of data storage and system access.
- (2) In view of the problem that the traditional alliance chain consensus algorithm PBFT needs to be in a trusted environment to ensure consensus security, node reliability is introduced to reflect the behavior of nodes in the consensus, and the node selection mechanism of the PBFT algorithm is improved. Selecting appropriate nodes to participate in the consensus makes PBFT have certain fault tolerance performance in a trustless environment and improves data security.
- (3) A multi-channel-based sensitive intelligence sharing strategy is proposed. This strategy designs intelligence sharing methods in four scenarios based on the Traffic Light Protocol (TLP), which can be implemented through the channel function in the Hyperledger Fabric framework. Data is isolated, and users can choose different channels to share sensitive data according to the type of sensitive data, thus protecting the privacy of users in the network.

Keywords: Threat Intelligence · Sharing · Blockchain · Consensus Mechanism · Channel

1 Design of Network Threat Intelligence Security Sharing Mechanism Based on Alliance Blockchain

1.1 Overall Architecture Design

1.1.1 Basic Architecture

The basic architecture of the network threat intelligence sharing mechanism based on alliance chain proposed in this article can be divided into application interaction layer and blockchain layer, as shown in Fig. 1.

The application interaction layer mainly encapsulates some basic functions that provide users with access, such as common basic functions in shared networks such as data upload, data query, and data download. These functions can be encapsulated in the form of standard transactions, which then become a transaction sent to the requesting node. After completing the request, the node stores the data on the blockchain, and the blockchain layer mainly includes the P2P network and blockchain, where the transmission and processing of transactions are mainly completed in the P2P network. If you want to package the transmitted transactions into blocks for storage, you need to make these transactions pass the consensus protocol, then package these transactions into the block, and then upload them to the blockchain.

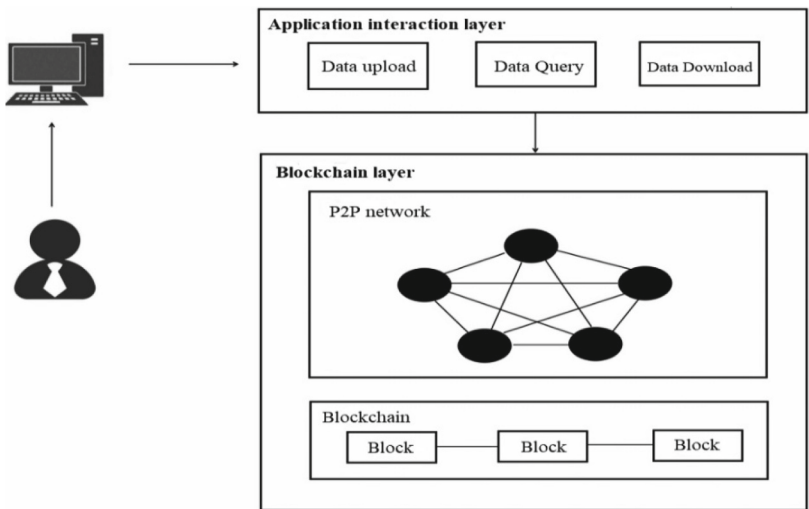


Fig. 1. Basic Architecture of the Scheme

1.1.2 Composition of Nodes Within the Network

According to the degree of trust of nodes in the consensus, various operations of nodes in the blockchain network will be restricted. According to the different executable operations, they can be divided into different levels.

1.2 Design of Threat Intelligence Sharing Process

In the blockchain based threat intelligence sharing mechanism, every shared threat intelligence data can be regarded as initiating a transaction, and the standard transaction process for each transaction is as follows:

- (1) Transaction endorsement. If the client wants to share a piece of intelligence data, the data will be encapsulated into a transaction form by the client, and then the client will send the transaction to the endorsement node in the network, requesting to upload the transaction to the blockchain. After receiving the request, the endorsement node will first read the current blockchain ledger locally, sign the blockchain status obtained after reading the ledger with the blockchain status after joining the transaction, and finally return the signature to the client.
- (2) Organize as blocks. After receiving the data returned by the endorsement node, the client program will package the data, and then all organizational nodes in the network will receive the broadcast of the data. After the organization node receives the data, it starts executing the already set pluggable consensus protocol, and transactions through the consensus protocol will be packaged into a block.
- (3) Confirmation and submission. The specific approach is to first verify the accuracy and legality of the transaction information in the block, and then obtain the blockchain status obtained by the endorsement node after reading the blockchain ledger when submitting the transaction. Compare whether this status is consistent with the current status. After these verifications are completed, the block can be added to the tail of the blockchain, and the status of the ledger will also be updated accordingly. Finally, the submitting node will reply with a message to notify the client that the submission was successful. The consensus is completed, and all nodes within the blockchain have reached a consensus on the uplink of the block, ensuring the synchronization of the blockchain ledger.

1.3 Consensus Mechanism Design

In the article, the classic consensus algorithm of the alliance blockchain, the Practical Byzantine Fault Tolerant Protocol (PBFT), is essentially a voting-based consensus mechanism.

The consensus improvement scheme based on node reliability proposed in this chapter combines node reliability evaluation with PBFT. The nodes in the entire network can be divided into ordinary node groups, voting node groups, candidate node groups, and alternative main node groups based on their reliability. The permissions of each group are different. At the same time, the concept of a consensus set is proposed, which only includes nodes from voting node groups, candidate node groups, and main node groups. Only nodes in the consensus set can participate in the consensus process. The overall process of the solution is shown in Fig. 2.

1.3.1 Design of Node Decision Evaluation Based on Entropy Weighted TOPSIS

The article based on the entropy weighted TOPSIS method regarding whether there is malicious behavior among nodes in the consensus. In the PBFT consensus process, all

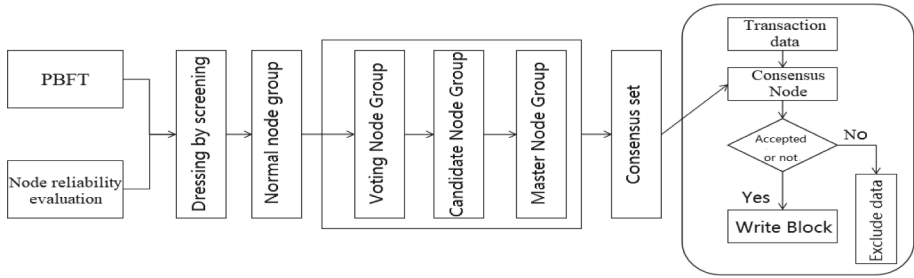


Fig. 2. Overall Process of the Scheme

consensus nodes will provide feedback on their decision opinions to the main node, whether they agree or disagree with the consensus. After obtaining widespread recognition, the data can be linked through the consensus. At least 66% of nodes in the PBFT consensus need to agree in order for the data to be linked normally. Nodes with the same decision opinion can be regarded as a set, The current node reliability and integration of each node are brought into the entropy weighted TOPSIS model to calculate the node decision score. The scores of the same set are summarized and compared, and the set with the highest total score is the final decision result. This article uses the number 1 to indicate “agree” and the number 0 to indicate “disagree”. The group decision-making evaluation process is shown in Fig. 3.

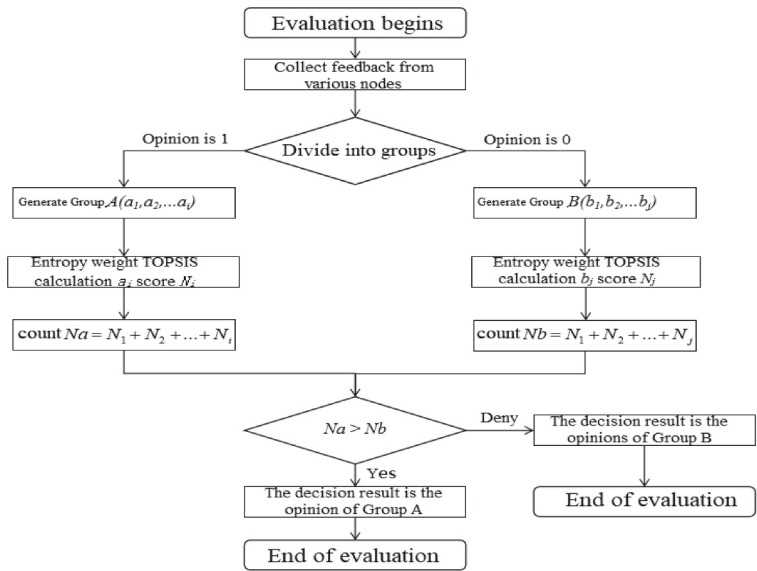


Fig. 3. Group Decision Evaluation Process

1.4 Design of Sensitive Intelligence Sharing Strategy

When users want to share a sensitive piece of intelligence, due to privacy considerations, the sharer may want to choose different sharing ranges based on different levels of sensitivity to prevent the leakage of sensitive intelligence. This requires us to distinguish between the data to be shared.

In response to the above issues, this article divides intelligence sharing methods in different scenarios based on the Traffic Light Protocol (TLP). Traffic Light Protocol (TLP) is a protocol specifically designed to regulate information sharing methods. The original design intention of TLP is to ensure that sensitive information can be shared with an appropriate audience, while not revealing the privacy of the sharer. Choose four colors to represent information sharing methods in different demand scenarios based on different TLP user needs.

- (1) Red (TLP: RED): When the information belongs to a highly sensitive data type, the sender hopes that the dissemination of the information should not be understood by multiple users, and this information can be set to the TLP: RED attribute. At the same time, the recipient of the information shall not share this information with anyone not specified by the sender of the TLP: RED information, and shall only use it for their own use.
- (2) Amber (TLP: AMBER): When the information belongs to a more sensitive data type, the sender wants the information to be spread between multiple users or the organization, and this information can be set to the TLP: AMBER attribute. Meanwhile, information recipients generally only share TLP: AMBER information with members of their own organization or users designated by the information sender.
- (3) Green (TLP: GREEN): When the information belongs to a generally sensitive data type, it can be shared with all users in the sharing range, and this information can be set as the TLP: GREEN attribute. The information recipient can share TLP: GREEN information with all organizations and individuals within the sharing scope, but cannot disclose the sharing channels and related information to the entire society.
- (4) White (TLP: WHITE): When the information belongs to a non-sensitive data type, the sharing of the information does not pose any privacy related security issues. The information can be set to the TLP: WHITE attribute and published on a publicly accessible platform. This public platform generally does not impose excessive restrictions on the user's identity or restrict their access to intelligence.

2 A Consensus Improvement Scheme for PBFT Based on Node Reliability

This chapter proposes a PBFT consensus improvement scheme based on node reliability, mainly focusing on improving the node selection part of PBFT. Under this scheme, as long as the normal contribution of nodes exceeds 50% of the total number of nodes and continues to contribute normally, it can be ensured that malicious behavior of nodes can be identified, and node reliability penalties can be imposed on malicious behavior. Nodes in the consensus network are divided into different weight levels based on node reliability. Nodes with lower levels are prohibited from participating in the consensus process, and only nodes with higher levels are selected to participate in the consensus, increasing the fault tolerance performance of the consensus.

2.1 Node Reliability

2.1.1 Related Definitions

Node reliability is used to reflect the behavior of nodes. Nodes that normally contribute to consensus algorithms will receive a node reliability reward, which generally has high reliability. Nodes with malicious behavior will be punished by node reliability, which generally has a lower reliability.

If there are a total of i nodes in the intelligence sharing system, the reliability C_i of the nodes is used to reflect the reliability level of node i , with a value range of $[0,1]$. The larger the C_i , the more likely it is to be a reliable normal node, while conversely, it is more likely to be a Byzantine node. The initial node reliability of the node is C_{init} , with a default value of 0.5. According to the different behaviors of nodes, they can be divided into three types: normal, cautious malicious nodes, and reckless malicious nodes.

Introduce node integral N to reflect the contribution of nodes and adjust the amplitude of changes in node reliability, with a range of $[0,1000]$, where N_i represents the integral of node i . The initial trust level of the node is N_{init} , with a default value of 500. As the score increases, the magnitude of changes in node behavior will be greater, meaning that the reliability of rewards will change more, and correspondingly, the reliability of penalties will also change more, which will reduce the reliability of nodes more than ordinary nodes after being punished. The different behaviors of nodes can lead to corresponding changes in points. The intelligence value can be divided according to the threat intelligence classification in Sect. 2.1.2, and the higher the value of the threat intelligence, the more points will be awarded.

2.1.2 Evaluation and Update Rules

The reliability of a node has an initial value. After completing a consensus each time, it will determine whether it is malicious behavior based on the behavior made by the node during the consensus process. Normal behavior will be rewarded with reliability and points, while malicious behavior will be punished with reliability and points. When participating in consensus for the first time, the reliability C_i of node i is expressed as:

$$C_i = \begin{cases} \max(\min(C_{init} + n_i \times f, 1), 0), & \text{Normal contribution of nodes} \\ \max(\min(C_{init} - n_i \times f \times p, 1), 0), & \text{Node has malicious behavior} \end{cases} \quad (2.1)$$

When not participating in consensus for the first time, the reliability C_i of node i is expressed as:

$$C_i = \begin{cases} \max(\min(T_i + n_i \times f, 1), 0), & \text{Normal contribution of nodes} \\ \max(\min(T_i - n_i \times f \times p, 1), 0), & \text{Node has malicious behavior} \end{cases} \quad (2.2)$$

Among them, $\max()$ represents the maximum value, $\min()$ represents the minimum value, C_{init} is the initial value of the node, T_i is the current node reliability, f is the reliability reward, p is the penalty coefficient, and n_i is the node change rate.

When participating in consensus for the first time, the integral N_i of node i is expressed as:

$$N_i = \begin{cases} \max(\min(N_{init} + y, 1000), 0), & \text{Normal contribution of nodes} \\ \max(\min(N_{init} - y \times p, 1000), 0), & \text{Node has malicious behavior} \end{cases} \quad (2.3)$$

When not participating in consensus for the first time, the integral N_i of node i is expressed as:

$$N_i = \begin{cases} \max(\min(G_i + y, 1000), 0), & \text{Normal contribution of nodes} \\ \max(\min(G_i - y \times p, 1000), 0), & \text{Node has malicious behavior} \end{cases} \quad (2.4)$$

where $\max()$ represents the maximum value, $\min()$ represents the minimum value, N_{init} is the initial node integral, G_i is the current node integral, y is the integral reward, and p is the penalty coefficient.

The node change rate n_i plays a role in regulating the magnitude of node reliability changes. Nodes with higher scores will have a higher degree of reliability variation compared to other nodes after normal contributions. Conversely, if there is malicious behavior, they will receive a heavier node reliability penalty compared to other nodes; After a node with a lower integral contributes normally, it will have a lower range of node reliability changes compared to other nodes. Conversely, if there is malicious behavior, it will receive a lighter node reliability penalty than other nodes. Once the node integral drops to 0, the reliability of the node will no longer change. The node change rate n_i is expressed as:

$$n_i = \frac{N_i}{\bar{N}} \quad (2.5)$$

N_i represents the integral of node i , and $\bar{N}$ represents the average integral of nodes within the entire network.

2.1.3 Evaluation and Update Implementation

After each time the data needs to be linked for consensus, the group decision results are obtained through the group decision function. Then, the decisions made by each node are compared with the group decision results. If they are consistent, a reward mechanism is executed, and if they are inconsistent, a punishment mechanism is executed.

Next, begin the process of evaluating and updating node reliability. This process will first obtain the decision opinion value d_i of node i , with the decision opinion values generally “agree” being 1 and “disagree” being 0. Then, the final group decision opinion result for the data that needs to be linked in this consensus will be obtained through the group decision function. There are generally three situations:

- (1) If the two values are equal, the node reliability score reward process should be carried out, and the range of node reliability should be controlled between 0 and 1, and the score range should be controlled between 0 and 1000.
- (2) If the two values are not equal, the node reliability and integral penalty process should be carried out, and the range of node reliability should be controlled between 0 and 1, and the range of integral should be controlled between 0 and 1000.
- (3) If there is no final conclusion in group decision-making, no action will be taken.

2.2 Evaluation of Group Decision Results Based on Entropy Weighted TOPSIS

In Sect. 2.1, node reliability will be rewarded or punished for normal contributions or malicious behavior, so it is important to determine how a node has malicious behavior. In the consensus process, group decision-making may sometimes fail to reach consensus due to the presence of malicious nodes, and normal contribution nodes and malicious nodes will form their own opinion sets. Therefore, a rule is needed to determine the decision scores of each set, forming a competition idea between nodes similar to the PoW algorithm, determining the final consensus set group decision result by comparing the decision scores between sets.

2.2.1 Overview of Entropy Weighted TOPSIS Evaluation Principles

Overview of entropy weight method. When constructing an indicator evaluation system, when our evaluation results involve two or more indicators, it is necessary to convert the data of the two indicators into data under the same dimension, and the weights of different indicators often vary. The entropy weight method determines the weight of indicators entirely through sample data calculation, which is more in line with the requirements of data analysis in this article compared to manual active weighting.

There are three main factors to consider when choosing the TOPSIS method: firstly, both evaluation indicators are positive indicators. The purpose of this article is to encourage all nodes in the system to have a higher node reliability and integral value, and nodes with lower values should align with nodes with higher values. Secondly, there are always normal contribution nodes and malicious nodes in the system. This article intends to consider the combination of the optimal values of the indicators in the evaluation set as the ideal state for optimizing decision-making opinions. Thirdly, TOPSIS is more in line with the competitive situation. Even if the numerical difference is not significant compared to individual normal contribution nodes, they will ultimately receive a lower evaluation value or even return to 0 due to the large distance from the optimal solution.

2.2.2 Steps for Calculating Node Decision Scores

- (1) The standardized processing of raw data includes two indicators in the consensus group decision-making indicator system. One is the current reliability of the node, which represents the reliability level of the node, with a range of [0,1]; One is the current integral value of the node, which represents the degree of contribution of the node. The higher the contribution, the greater the reliability reward, and the value range is [0,1000]. The operation between different dimensions is meaningless, which requires processing these two types of data to make them data that can be operated on in the same dimension. Due to the fact that both indicators in this article are positive, the data is generally dimensionless processed using formula 2.6, and the processed results are then used to construct a normalized matrix:

$$x_{ij} = \frac{v_{ij} - \min_{i=1,2,\dots,m} v_{ij}}{\max_{i=1,2,\dots,m} v_{ij} - \min_{i=1,2,\dots,m} v_{ij}} \quad (2.6)$$

Among them, v_{ij} and x_{ij} represent the original and standardized values of the i -th node on the j -th indicator, respectively.

- (2) The entropy weight method calculates the objective weight of indicators. For indicators with rich information, the greater the information entropy, the greater the weight to be given.

The first step is to calculate the proportion of the standardized values of each indicator through the proportion transformation of the standardized data in the standardization matrix $X = (x_{ij})_{m \times n}$. Calculate the contribution degree P_{ij} of the indicator using formula 2.7:

$$P_{ij} = \frac{x_{ij}}{\sum_{i=1}^m x_{ij}} \quad (2.7)$$

Step 2: use formula 2.8 to calculate the entropy weight e_j of each indicator:

$$e_j = -\frac{1}{\ln n} \sum_{i=1}^m P_{ij} \ln P_{ij} \quad (2.8)$$

Among them, when defined as $P_{ij} = 0$, $P_{ij} \ln P_{ij} = 0$.

Step 3: use formula 2.9 to calculate the entropy weight w_j of each indicator:

$$w_j = \frac{1 - e_j}{n - \sum_{j=1}^n e_j}, j = 1, 2, \dots, n \quad (2.9)$$

Step 4: use formula 2.10 to calculate the relative importance of indicators:

$$S_i = \sum_{j=1}^n w_j P_{ij} \quad (2.10)$$

- (3) Use TOPSIS for comprehensive evaluation.

The first step is to construct a standardized data weighted decision matrix, multiply the dimensionless normalized matrix $X = (x_{ij})_{m \times n}$ by the indicator weight vector $w^0(w_1, w_2, \dots, w_n)$, as shown in formula 11, to obtain the standardized data decision matrix: $Y = (y_{ij})_{m \times n}$

$$Y = (y_{ij})_{m \times n} = (w_i^0 x_{ij}) \quad (2.11)$$

The second step is to determine the positive and negative ideal solutions. The positive ideal solution A^* is an optimal solution, which is the optimal value of the indicator; The negative ideal solution A^- is the worst solution, that is, the worst difference. The optimal and worst solutions are determined by formula 12 and 13:

$$A^* (y_1^*, y_2^*, \dots, y_j^*, \dots, y_n^*) = \left(\max_{i=1,2,\dots,m} y_{ij} | j \in J \right) \quad (2.12)$$

$$A^- (y_1^-, y_2^-, \dots, y_j^-, \dots, y_n^-) = \left(\min_{i=1,2,\dots,m} y_{ij} | j \in J \right) \quad (2.13)$$

Among them, $J = \{j | j \in \{1, 2, \dots, n\}\}$

Step 3: Calculate the Euclidean distances S_i^+ and S_i^- from the standardized vectors of the indicators to the optimal and worst solutions, respectively:

$$S_i^+ = \sqrt{\sum_{j=1}^n (y_{ij} - y_j^*)^2} \quad (2.14)$$

$$S_i^- = \sqrt{\sum_{j=1}^n (y_{ij} - y_j^*)^2} \quad (2.15)$$

Step 4: calculate the final decision opinion score C_i . The larger the C_i value, the higher the decision opinion score of the evaluation node; The smaller the C_i value, the lower the decision opinion score of the evaluation node.

$$C_i = \frac{S_i^-}{(S_i^+ + S_i^-)}, i = 1, 2, \dots, m \quad (2.16)$$

2.2.3 Implementation of Group Decision Evaluation

After receiving feedback from each node, the group decision evaluation function integrates different opinion nodes into different groups, calculates the decision scores of each group, and compares the decision scores of each group. The decision opinion of the group with the highest score is selected as the group decision opinion. Firstly, it is necessary to define the variable names of the group decision evaluation algorithm, as shown in Table 1.

Table 1. Variable Names in Group Decision Evaluation

Variable name	Describe
i	Node number
j	Opinion is 1 node count
k	Opinion is 0 node count
Na	Opinion is 1 node, total evaluation score
Nb	Opinion is 0 node, total evaluation score
di	Decision results of node i
A[]	Opinion is the group of node 1
B[]	Opinion is the group of node 0

2.3 Node Partitioning Mechanism

The definition of C_i represents the reliability of nodes, and each node i can be divided into four levels based on C_i : A, B, C, and D, corresponding to four types of node groups: ordinary node group, voting node group, candidate node group, and main node group.

When the reliability of a node is greater than C_{good} , i.e. the credit level of $C_i > C_{good}$ node is A, this type of node is qualified to serve as the master and slave nodes in the consensus mechanism; The credit level of nodes with reliability in the range $C_{normal} \leq C_i < C_{good}$ is B, and this type of node can substitute as the main node and serve as the subordinate node of the consensus mechanism when the A type node cannot serve as the main node; When the reliability is $C_{bad} \leq C_i < C_{normal}$, the node's credit level is C, and this type of node can serve as a subordinate node in the consensus mechanism; When the reliability of a node is $C_i \leq C_{bad}$ and $C_i < 0.4$, the credit level of the node is D, and this type of node cannot participate in the consensus mechanism. The nodes participating in the PBFT consensus can be considered as having a total of $3f + 1$ nodes. Each time the consensus is completed, the nodes are graded according to the partitioning mechanism. And after obtaining the thresholds for each level, the node credit rating is performed, as shown in Fig. 4.

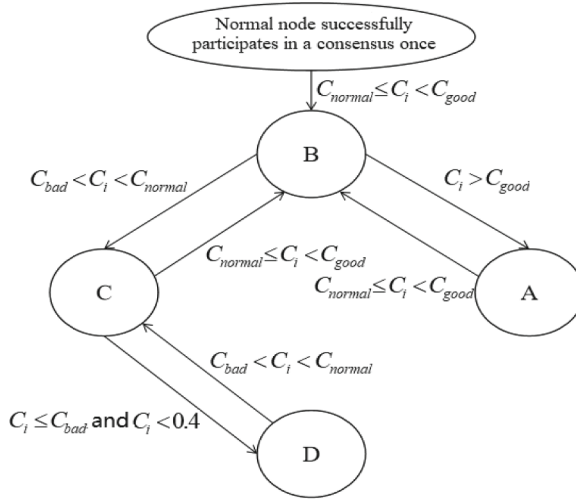


Fig. 4. Node Credit Rating Division

The division of node permissions based on node reliability can urge nodes to perform normal behavior in consensus, and as the number of consensus rounds increases, malicious nodes in the system can also be gradually removed from the consensus set, allowing only nodes with higher reliability to participate in the consensus process. This will make the consensus process only contain as few Byzantine nodes as possible, and most nodes are normal nodes who will make correct decisions.

2.4 Consensus Process

Based on the node partitioning mechanism, master and slave nodes are selected. The selection of slave nodes comes from the nodes in the voting node group, and the selection of master nodes comes from the nodes in the master node group and node candidate

group. The master and slave nodes together form the replica node group participating in this consensus. After the node selection is completed, the consensus process of PBFT begins. A complete consensus in this scheme generally needs to be completed in four steps, as shown in Fig. 5. The specific steps for consensus are as follows:

Step 1: After the client requests data to be uploaded, the master node will generate a pre prepared certificate upon receiving the client's request, which includes the following information: blocks to be verified, timestamp, master node signature, etc. This certificate will be broadcasted by the master node to all slave nodes participating in the consensus and enter a pre prepared state.

Step 2: After receiving the pre prepared certificate from the master node broadcast in step 1, the slave node will forward the received certificate to other slave nodes and enter the prepared state.

Step 3: After receiving certificates from other slave nodes in step 2, the slave node will perform a verification program to verify all data in the certificate. Finally, the verification result will be sent to the other slave nodes, whether it passes or does not pass the verification. As long as a total of $2f + 1$ verified replies are received from the node, it will enter the submission state and broadcast it to the entire network.

Step 4: If a total of $2f$ nodes enter the submission state, it means that the data has been approved by most nodes in the network and can be uploaded to the blockchain. Each node will update the data to the local blockchain ledger, thus achieving consistency across the entire network.

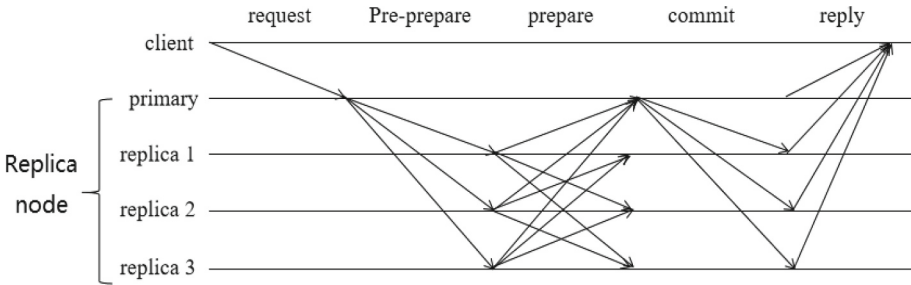


Fig. 5. Consensus Process of Improvement Plan

2.5 Experiment and Analysis

2.5.1 Node Reliability

Assuming that the attacker imitates the behavior of normal contributors, in order to disguise themselves as ordinary users, the attacker needs to reach a consensus normally, and these normal behaviors are used as the cost of executing malicious behaviors to maintain the node reliability required for the attacker to continue attacking. Therefore, we conducted simulations to demonstrate the impact of the attacker's costs under malicious

behavior in cycle t . In this experiment, three types of nodes were simulated: normal contribution nodes without any malicious behavior, namely $t = 0$; Be cautious of malicious nodes, conduct a malicious attack every 10 rounds, and contribute normally to disguise oneself during the rest of the time, namely $t = 10$; Recklessly malicious nodes, conduct malicious attacks every two rounds, and contribute normally at the rest of the time, namely $t = 2$. In terms of the selection of initial values, through multiple experimental comparisons, the reliability changed is set to 0.01. The range of reliability changes for nodes under this value is moderate, avoiding the node from approaching peak 1 too quickly or being excluded from the consensus set prematurely due to misoperation due to large changes. This is in line with the design expectations and scenario needs of this article. According to the value of the penalty coefficient p , the changing trends of the two types of malicious nodes are also different. In order to be closer to the real data, some Gaussian noise ($m = 0$, $s.d = 0.01$) is added. The results are shown in Figs. 6, 7 and 8.

In the figure, when the p -value is 6, although the reckless malicious node quickly drops to 0, the cautious malicious node is not recognized, and the node reliability gradually approaches 1 after multiple rounds of consensus; When the p -value is 16, the reliability of reckless malicious nodes quickly decreases to 0, while cautious malicious nodes gradually approach 0 after a certain number of rounds; When the p -value is 26, both reckless and cautious nodes quickly decrease to 0. The experimental results show that in the case of high p -values, this method can effectively identify two types of malicious nodes and make their node reliability quickly approach 0.

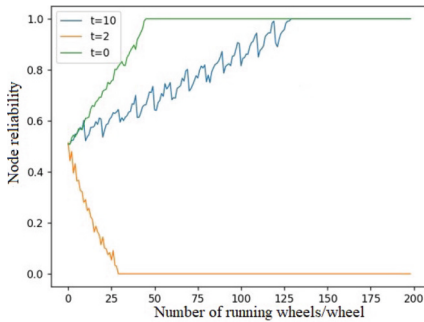


Fig. 6. Penalty coefficient equal to 6 ($p = 6$)

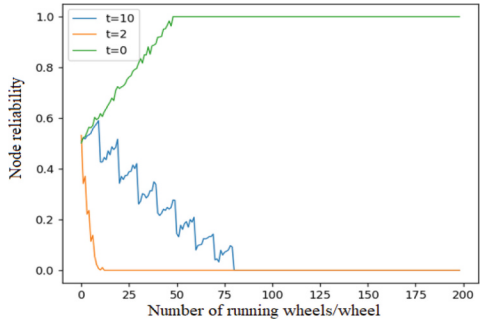


Fig. 7. Penalty coefficient equal to 16 ($p = 16$)

2.5.2 Consensus Fault-Tolerant Performance and Security Analysis

In response to the problem of low fault tolerance in the PBFT algorithm, once the number of Byzantine nodes exceeds 33%, consensus will not be able to pass normally. So uses entropy weighted TOPSIS to calculate the decision scores of normal and malicious nodes, so that malicious behavior can be identified and node reliability penalties can be imposed on malicious nodes. Finally, voting node groups are gradually eliminated based

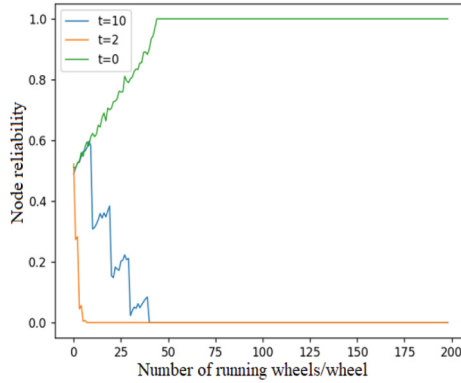


Fig. 8. Penalty coefficient equals 26 ($p = 26$)

on the node selection mechanism. Through simulation experiments, it has been proven that the system can effectively exclude malicious nodes from the consensus set.

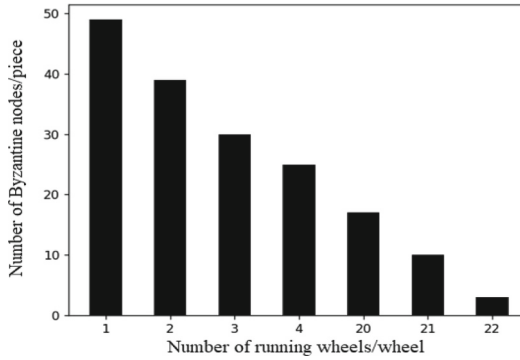


Fig. 9. Number of Byzantine nodes in the improved scheme

As shown in Fig. 9, it can be seen that as the number of consensus rounds continues to increase, this scheme will gradually exclude Byzantine nodes from the consensus set. Moreover, this scheme can not only quickly eliminate reckless malicious nodes without disguise, but also identify cautious malicious nodes disguised as normal nodes for malicious behavior after running a certain number of consensus rounds, significantly improving the fault tolerance performance of the consensus algorithm.

3 Sensitive Information Sharing Strategy based on Multi-channel

This chapter is mainly to solve the privacy protection problem when sharing sensitive intelligence, when users sharing more sensitive information, they often need privacy protection. Hoping that the information they share is only shared within a limited scope,

and other users outside the scope cannot see this information, so as to protect their privacy. Therefore, this chapter designs a sharing strategy for different sensitive intelligence data needs based on the Traffic Light Protocol (TLP), which can adopt the corresponding sharing method according to the specific privacy protection needs of users, and finally use the channel function in the Hyperledger Fabric framework to implement the method.

3.1 Multi Channel

3.1.1 Definition of Multichannel

A channel is a confidential virtual data link created by nodes in a network, which is a logical concept and no entity. Each channel is not always open to all members of the network, and the peers in each channel are set. Each channel shares a ledger unique to its own channel, which means that only peer nodes in the channel can share information, and the ledger of peer nodes in different channels is also different, and a peer node can also exist in multiple channels at the same time.

3.1.2 Multi Channel Design

Channels in Hyperledger specify only individual nodes or are fully exposed depending on your needs. If a channel is fully public, the ledger of that channel can be backed up for nodes across the network. If the a private channel, the creator often designates some nodes as members of the channel when creating the channel. Each channel is created differently and has different functions, which can be divided into system channels and user channels. When each blockchain system is started, the system automatically creates a system channel, which is used to record the configuration information of each channel. User channels can be created by users independently, mainly for the user's personal needs to create, such as only want to share data with individual users, you can create a channel as long as you and the target user.

3.1.3 Interaction Between Channels

When the alliance chain network is started, the two system channels config and global will be automatically created, and there is no need to manually create them, At that time the client can interact through the system based on the framework of this article.

After the network starts, the department or company head uses the command to create the channel. The channel can be seen as a private chain, and the creator does not want the information shared in the channel to be exposed to the entire network, only for data sharing between peer1 and peer2, without being discovered by other nodes. The procedure for creating a channel is as follows:

- (1) The channel creator needs to send a request to create a channel, and the system will encapsulate the request as a transaction in a config channel, send it to the orderer for sorting and packaging.
- (2) The Order node first sorts the transactions, then packages the sorted transactions into blocks and writes them to the config channel, and finally broadcasts the results to peers across the network.

- (3) After each peer in the blockchain network receives the transaction, only peer1 and peer2 read the channel information in the transaction and write it to the world state. Once these operations are done, the first chunk is created for the channel, and once the creation is complete, the channel is started.
- (4) A new transaction is added to the global channel, which is packaged into a block and added to the channel, which mainly records the relevant information of the newly added block of the config channel.

After the company or department head creates the channel, the administrator of the channel is the channel creator, and the administrator can create the smart contract chain-code used by the channel through the command, which mainly contains the functions of data query and addition, as well as the functions added by the contract editor himself. Once the deployment is complete, the company or department head invokes the chain-code's data addition function to store the newly obtained network threat intelligence into the blockchain.

3.2 Smart Contract Design

Smart contracts, also known as chain codes, can be deployed on nodes to perform basic operations on data. In this article, smart contracts are mainly used to complete data upload and query functions. Below, the design of the intelligence upload module and intelligence query module in this chain code will be introduced.

3.2.1 Design of Information Upload Module

This part of the contract mainly implements the operation of users uploading intelligence to the blockchain. When users request to upload intelligence, they need to pass the intelligence to the CreateAsset function according to the specified parameters. In this article, four parameters are mainly designed for each intelligence, namely ID, CLI, Owner, and Color. The ID represents the unique identifier of the intelligence in the database, CLI represents the content of the network threat intelligence, Owner represents the owner of the intelligence, and Color represents the sensitivity level of the intelligence. This method checks the number of parameters. If the number of parameters is inconsistent, the execution is refused and an error message is returned. If the number of parameters is consistent, the PutState method is executed to upload the data to the blockchain.

3.2.2 Design of Intelligence Query Module

This part of the contract mainly implements the operation of users querying intelligence. When a user requests to query a certain intelligence, the ID of the intelligence to be queried is passed as a parameter to the ReadAsset function. This function will use the GetState method to search in the blockchain based on the intelligence ID. If the intelligence is found, it will return all information containing four parameters. If the intelligence cannot be found, it will return a query failure prompt.

3.3 Implementation of Shared Policy Function

The sharing strategy designed in this article can be achieved by completing three parts: building a Fabric testing network, configuring channels, and deploying smart contracts. The following will provide a detailed introduction to the implementation of these three parts.

3.3.1 Building a Fabric Testing Network

To build a Fabric testing network, it is necessary to establish a total of three testing organizations, namely Org1, Org2, and Org3, as well as an Order node responsible for sorting transactions. Each organization needs to include a Peer node and a client node. The Peer node is responsible for storing blockchain data and related transactions, and the client node is responsible for calling chain codes to complete some basic operations. The information of each organization is shown in Table 2.

Table 2. Organization Information in the Network

Organization ID	Number of client nodes	Organization Description
Org1	1	Organizations with sensitive information
Org2	1	Organizations cooperating with Org1
Org3	1	Common organizations in the network

After designing the information of the three organizations, you can see the cryptogen program provided by fabric to generate the encryption materials of the Peer and Order nodes in the three organizations. These encryption materials mainly include a set of certificates and keys, which mainly include the identity of the representative nodes and signature and unsigning functions for identity authentication in transactions. The cryptogen program mainly relies on the configuration file crypto config with the suffix yaml to generate certificates for various nodes in the system. It mainly creates three organizations and a sorting node, with each organization having a Peer node and a client node. The Template in the figure represents the Peer node and users represents the client node.

3.3.2 Channel Configuration

In order to achieve the isolation of sensitive and non sensitive intelligence, at least two mutually independent channels need to be configured. Among them, organization Org1 has a relatively sensitive network threat intelligence that needs to be shared only with cooperative organization Org2. Therefore, a sensitive intelligence channel needs to be established between Org1 and Org2 to share sensitive intelligence. Only the organization nodes inside the channel can access data, while organizations outside the channel cannot access this intelligence; The public data sharing channel includes all organizations to achieve global data sharing. The channel configuration information within the network is shown in Table 3.

Table 3. Channel Configuration Information

Channel name	Channel identification	Organization in channel
Sensitive channel	Tlp-red	Org1,Org2
General sensitive information channel	Tlp-amber	Org1,Org3
Public information sharing channel	Tlp-green	Org1,Org2,Org3

The configtxgen program provided by Fabric can be used to generate initialization files for Orders and channels, and all transaction related configurations can be implemented through this program. The configtxgen program mainly relies on the configuration file configtx.yaml to configure the above information. When creating a channel, the template in the file is read and a channel file is generated. This template information mainly includes sorting nodes and organizational members within the channel.

3.3.3 Chain Code Deployment

Any organization within the channel can instantiate the chain code through its client nodes within the channel. After instantiation, the client nodes can initialize the chain code. After completing the above operations, a fabric testing network is built, and the client nodes of each organization can call the methods in several types of smart contracts designed in Sect. 3.2.

3.4 System Testing

This section mainly tests the sensitive intelligence sharing channel tlp-red, which has been divided into member organizations, based on the deployed Fabric testing system to test whether the channel achieves data isolation function during data exchange between organizations.

3.4.1 Introduction to Testing Environment

The testing environment of this article is mainly completed within a virtual machine. The system is Ubuntu 20.0.4, and the hardware environment of the virtual machine is shown in Table 4:

Table 4. Hardware Environment

system	Ubuntu 20.0.4
Memory	2GB
Hard disk capacity	20GB
Number of processors	1

3.4.2 Functional Testing

- (1) Organization Org1 has highly sensitive network threat intelligence and does not want to share it with other organizations within the network. It only wants to share it with Organization Org2. The client node of Org1 uploads these two sensitive intelligence pieces in the tlp-red channel of only Organization Org1 and Organization Org2, encapsulating the content, intelligence owner, and intelligence sensitivity level of the threat report into contract parameters, The data addition module CreatAsset function in the chain code is called to add these two pieces of information to the channel ledger. The call result is shown in Fig. 10.

```
[root@localhost1 test-network]# peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $(PWD)/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C tlp-red -n basic1 --peerAddresses localhost:7051 --tlsRootCertFiles $(PWD)/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles $(PWD)/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function": "CreateAsset", "Args": ["asset1", "attackIP=192.168.1.2", "peer1", "red"]}'
2022-05-12 21:35:46.554 CST [chaincodeCmd] chaincodeInvokeQuery -> INFO 001 Chaincode invoke successful. result: status:200
[root@localhost1 test-network]# peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $(PWD)/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C tlp-red -n basic1 --peerAddresses localhost:7051 --tlsRootCertFiles $(PWD)/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles $(PWD)/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function": "CreateAsset", "Args": ["asset2", "attackMAC=01-19-ff-3c-7e-7d", "peer1", "red"]}'
2022-05-12 21:36:56.810 CST [chaincodeCmd] chaincodeInvokeQuery -> INFO 001 Chaincode invoke successful. result: status:200
```

Fig. 10. Information addition result

- (2) Due to the fact that Org2 and Org1 are on the same channel tlp-red, they can share their ledger on the same channel. At this point, Org2 can call the GetAllAssets function of the data query module in the chain code to query all sensitive intelligence uploaded by Org1 to the channel ledger. If successful, the content of two threat intelligence messages will be returned. The query results are shown in Fig. 11.

```
[root@localhost1 test-network]# peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile $(PWD)/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C tlp-red -n basic1 -c '{"function": "GetAllAssets", "Args": []}'
[{"ID": "asset1", "CTI": "attackIP=192.168.1.2", "Owner": "peer1", "Color": "red"}, {"ID": "asset2", "CTI": "attackMAC=01-19-ff-3c-7e-7d", "Owner": "peer1", "Color": "red"}]
```

Fig. 11. Organizational Org2 Intelligence Query Results

- (3) Due to the fact that organization Org3 is not in the tlp-red channel, the ReadAsset function of the data query module in the link code cannot find the threat intelligence labeled asset1 that organization Org1 just added in the tlp-red channel, nor can it view the ledger of the tlp-red channel. The query results are shown in Fig. 12.

From this result, it can be seen that both querying the data labeled aeest1 uploaded by Org1 and querying the ledger of channel tlp-red indicate that it cannot be viewed. Through the function of channels, data isolation between sensitive and non sensitive intelligence can be achieved. Even in distributed ledgers such as blockchain, some data can be prevented from being viewed by organizations outside the channel, thus protecting user privacy.

```
[root@localhost1 test-network]# peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --t
ls --cafile $(PWD)/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example
.com-cert.pem -C tlp:red -n basic1 -c '{"function": "ReadAsset", "Args": ["asset1"]}'
Error: error getting endorser client for query: endorser client failed to connect to localhost:11051: failed to create new
connection: connection error: desc = "transport: error while dialing: dial tcp [::1]:11051: connect: connection refused"
[root@localhost1 test-network]# peer channel getinfo -c tlp:red
Error: error getting endorser client for channel: endorser client failed to connect to localhost:11051: failed to create ne
w connection: connection error: desc = "transport: error while dialing: dial tcp [::1]:11051: connect: connection refused"
```

Fig. 12. Organizational Org3 Intelligence Query Results

Acknowledgments. The work is supported by Innovation Fund Project for College Teachers in Gansu Province in 2025 (Title: Research on the governance of network public opinion reversal based on blockchain technology), Key Project of Teaching Reform Research at the School Level of Gansu University of Political Science and Law in 2024 (NO.GZJG2024-A04, Title: Research and Implementation of CSCW Cloud Storage Collective Lesson Preparation System Based on Blockchain Technology), 2024 Gansu University of Political Science and Law “Three in One Education” Research Project (NO. GZSQYR-37, Title: Research and Implementation of CSCW Network Teaching Platform Based on WEB in the Direction of Network Education). 2024 Lanzhou Philosophy and Social Sciences Planning Project (No. 24-B45, Research on Network Public Opinion Cloud Governance of Major Public Events in Lanzhou).

References

1. Chen, P., Desmet, L., Huygens, C.: A study on advanced persistent threats. In: IFIP International Conference on Communications and Multimedia Security. Springer, Berlin, Heidelberg, pp. 63–72 (2014)
2. Piper, S.: Definitive Guide to Next Generation Threat Protection. CyberEdge Group, LLC (2013)
3. Choo, K.K.R., Smith, R.G., McCusker, R., et al.: Future directions in technology-enabled crime: 2007–09. Canberra: Australian Institute of Criminology (2007)
4. FireEye, Inc. Advanced Targeted Attacks – How to Protect Against the Next Generation of Cyber Attacks (2012)
5. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research* **1**(1), 80 (2011)
6. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2009)
7. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media Inc, USA (2015)
8. Gong, S., Lee, C.: BLOCIS: blockchain-based cyber threat intelligence sharing framework for sybil-resistance. *Electronics* **9**(3) (2020)



Towards Automated Decoding of Vehicle CAN Data Using Deep Learning

Ankang Jiao¹✉, Wendi Li², Honglong Zhu¹, and Hao Han¹

¹ College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
jak866@nuaa.edu.cn

² The National Key Lab for Novel Software Technology, Nanjing University, Nanjing 210023, China

Abstract. Controller Area Network (CAN) is a ubiquitous broadcast protocol facilitating real-time communication among electronic control units (ECUs) in vehicles. Despite its importance, the proprietary nature of CAN message formats across different vehicle models and manufacturers poses significant challenges for signal interpretation and data analysis. This paper introduces AutoDBC, a novel end-to-end CAN-Bus Data Domain Intelligent Segmentation algorithm designed to address the limitations of current manual and automated CAN data reverse engineering methods. The AutoDBC framework innovatively extracts features at three levels-bit, byte, and message-to capture the nuanced characteristics of CAN data, accounting for the mixed endianness commonly found in automotive communications. Unlike existing approaches that rely solely on bit-level features such as bit flipping rates, AutoDBC considers the spatial-temporal relationships across bytes and messages, providing a more comprehensive understanding of the data. At the core of AutoDBC is a custom deep learning model that enhances the precision of signal boundary identification. Trained on a diverse dataset of real-world CAN bus data collected from various personal vehicles, this model demonstrates a remarkable improvement in accuracy compared to state-of-the-art (SOTA) methods. Our evaluation shows that AutoDBC achieves a 10–30% increase in precision for signal boundary and signal field identification, a significant leap forward in the field of automotive data analysis.

Keywords: CAN · Signal Boundary Identification · Signal Field Identification · Intelligent Algorithm

1 Introduction

The Controller Area Network (CAN) is a broadcast protocol commonly used for real-time communication among electronic control units (ECUs) in vehicles. With rapid advancements in automotive technology, the global market for CAN-based automotive data is projected to grow significantly, with an estimated value between 450 and 750 billion USD by 2030 [1]. By decoding CAN data, engineers and researchers can perform

diagnostics, troubleshoot issues, analyze security vulnerabilities, and customize after-market modifications. However, the format of CAN data is often proprietary to Original Equipment Manufacturers (OEMs) [2]. Even within a single vehicle make, different models or versions can have varying CAN message formats. This lack of standardization poses a significant challenge to applying a universal approach for interpreting CAN data, as shown in Fig. 1.

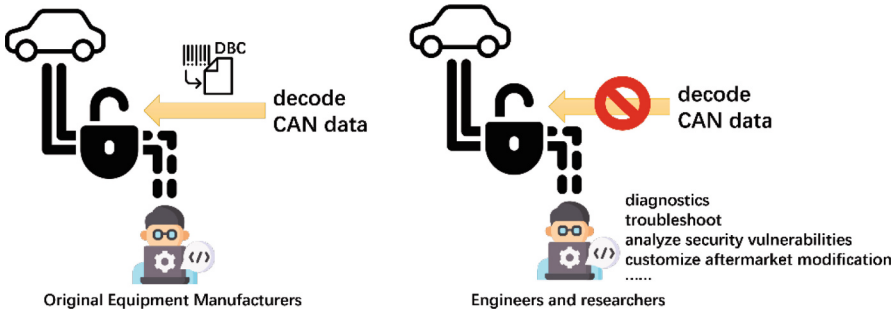


Fig. 1. Challenges of interpreting CAN data across different vehicle models.

Currently, reverse engineering is the primary technique used to decipher the meaning of messages on the CAN bus. This process typically involves capturing and analyzing CAN traffic, correlating messages with known vehicle behaviors or sensor readings, and conducting experiments to understand the functionality of each message [3, 4]. However, manual reverse engineering is labor-intensive, time-consuming, and requires significant expertise, making it difficult to scale.

In response, considerable research has been done on automating CAN bus reverse engineering. Notable methods include the application of machine learning for CAN data feature identification [5], the introduction of CAN data segmentation algorithms [6], and the use of bit-flipping counts for labeling CAN signals [7]. Although these approaches represent important advances, they predominantly rely on bit-level features, such as bit-flipping rates, without fully capturing the spatial-temporal relationships across bytes and messages. Furthermore, many studies assume a uniform byte order for all signals within a message, which is often not true in real-world CAN data, where mixed-endian formats are common. Consequently, existing methods face limitations in accurately identifying signal boundaries and fields.

To address these challenges, we propose AutoDBC, an end-to-end CAN data domain segmentation framework that incorporates multi-level feature extraction across bit-, byte-, and message-level representations. AutoDBC leverages a custom deep learning model to improve the accuracy of signal boundary identification by accounting for mixed byte orders. The model is trained on real CAN data collected from multiple personal vehicles, enabling it to generalize across different vehicle models. Our results demonstrate a 10–30% improvement in accuracy compared to state-of-the-art (SOTA) methods for signal boundary and field identification.

Main Contributions:

- We analyze the effect of mixed byte orders on signal boundary identification and propose a novel feature extraction method that captures bit-, byte-, and message-level characteristics of CAN data.
- We introduce a custom deep learning model that achieves higher accuracy than existing methods in signal boundary identification, using CAN data collected from various vehicles in random driving conditions.
- We implement AutoDBC in a real-world setting, using CAN data collected from multiple personal vehicles over extended periods. Our experimental evaluation shows that AutoDBC is robust across different vehicle models, achieving consistent and accurate signal boundary identification under diverse driving conditions. The identification accuracy outperforms SOTA methods.

The rest of this paper is structured as follows: Sect. 2 provides the necessary background on CAN message formats and DBC files. Section 3 presents the detailed design of the AutoDBC algorithm. Section 4 discusses the evaluation of AutoDBC using real CAN data, and Sect. 5 concludes the paper with a summary of our findings and future directions.

2 Background

This section introduces the fundamental structure of CAN messages and the critical role of DBC files in interpreting raw CAN data. While the DBC file provides a standardized way to map raw CAN data to meaningful signals, the proprietary nature of these files poses challenges for aftermarket applications and third-party developers. In the following sections, we present AutoDBC, a novel framework that automates CAN data segmentation and signal boundary identification, even without DBC files.

2.1 CAN Message Format

The CAN 2.0 specification defines both the physical and data link layers, corresponding to Layers 1 and 2 in the Open Systems Interconnection (OSI) model [8]. As illustrated in Fig. 2, the standard CAN message format consists of several key fields. The Start of Frame (SOF) is used for synchronization purposes, followed by the Arbitration ID (AID), which serves as the message’s unique identifier and determines its priority on the bus. The Remote Transmission Request (RTR) flag differentiates data messages from remote messages.

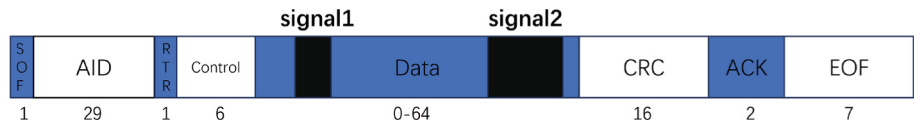


Fig. 2. CAN frame/message format defined in CAN Spec 2.0.

The DBC file also specifies the minimum and maximum possible values for each signal, along with the expected receiving ECU (or node) that processes the message. This detailed structure allows engineers to reverse-engineer CAN data and make sense of the proprietary information contained within the message payloads.

While the DBC file format is standardized, each vehicle manufacturer creates proprietary DBC files specific to their models. These files are often not publicly available, creating barriers for third-party developers aiming to analyze, modify, or customize vehicle data, especially in aftermarket applications.

The boundaries of each signal within the message are determined by the start bit and signal length, as described in the DBC file. There are two common byte orders: big-endian, where the most significant byte (MSB) is stored first, and little-endian, where the least significant byte (LSB) is stored first. Mixed-endian formats, where different signals within the same message use different byte orders, further complicate the reverse-engineering process. In the absence of DBC files, reverse-engineering techniques must be employed to deduce the signal boundaries, data encoding, and physical meanings of the signals.

3 System Design

3.1 Problem Definition

We denote the data field of message m as V_m (Eq. 1) and the corresponding 64-bit matrix format (also known as the CAN Communication Matrix) as M_m (Eq. 2).

$$V_M = [v_0, v_1, v_2, \dots, v_{62}, v_{63}] \quad (1)$$

$$M_m = \begin{bmatrix} b_{00} & b_{01} & \dots & b_{07} \\ b_{10} & b_{11} & \dots & b_{17} \\ \vdots & \vdots & \ddots & \vdots \\ b_{70} & b_{71} & \dots & b_{77} \end{bmatrix} \quad (2)$$

M_m represents a byte per row, comprising a total of 8 bytes, which equates to 64 bits. For each b_{ij} ($i, j \in [0, 7]$) in M_m , we have $f(b_{ij}) = v_x$ such that $x = i \times 8 + j$.

Suppose message m contains N signals. Let S_n ($n \in [0, N]$) denote the n -th signal within the message. $Start_n$ indicates the starting bit position of this signal in V_m . The pair $(Start_n^x, Start_n^y)$ represents the starting bit position of this signal in M_m . The variable $length_n$ represents the length of this signal, while the pair (End_n^x, End_n^y) represents the ending bit position of this signal. The byte order is denoted as F_n .

As illustrated in Fig. 4, when F_n represents big-endian, $Start_n$ corresponds to $(Start_n^x, Start_n^y)$, defining the Most Significant Bit (MSB) boundary of the signal. Here, $Start_n^x = \lfloor Start_n/8 \rfloor$, and $Start_n^y = 7 - (Start_n - Start_n \cdot 8)$. The matrix traverses the signal length in a zigzag pattern from the start bit position downwards and to the right, with the last bit representing the signal's ending bit. When F_n denotes little-endian, $Start_n$ corresponds to (End_n^x, End_n^y) , defining the Least Significant Bit (LSB) boundary of the signal. In this case, the matrix traverses the signal length in an inverted zigzag

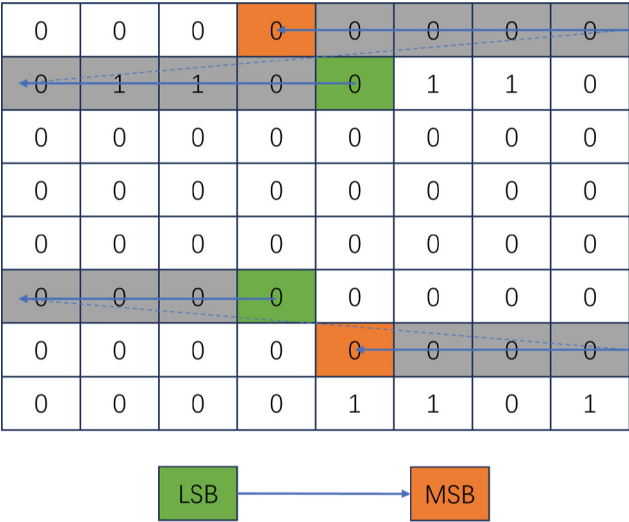


Fig. 4. Mixed byte orders of signals within a single CAN message

pattern from the signal’s termination bit downwards and to the left, with the last bit being the signal’s starting bit.

In the absence of a corresponding DBC file for a given trace consisting of messages with the same ID, our primary goal is to identify each signal’s boundaries within the message of the given trace, regardless of byte order. We refer to this problem as Signal Boundary Identification, is currently focused on the Least Significant Bit (LSB) due to the sparse and less prominent features of the Most Significant Bit (MSB). The next goal is to infer the MSB and byte order of each signal to determine the signal field, which we define as Signal Field Identification. The challenge lies in the fact that each signal within the message may have a different byte order, making the identification process complex.

3.2 Overall Workflow

The overall workflow of AutoDBC is illustrated in Fig. 5. Our approach begins by inputting a sequence of raw CAN messages. A data preprocessing module is used to denoise the dataset and remove featureless traces, mitigating class imbalance. Next, messages are grouped by their MIDs. In the feature extraction and inference module, the proposed deep learning model first extracts bit-level, byte-level, and message-level features, and then infers the boundary of each signal within messages of the same group, specifically focusing on the LSB boundary. In the end, the signal speculation algorithm determines each signal’s byte order and MSB to identify the signal field. The entire approach process is performed end-to-end in an automated manner.

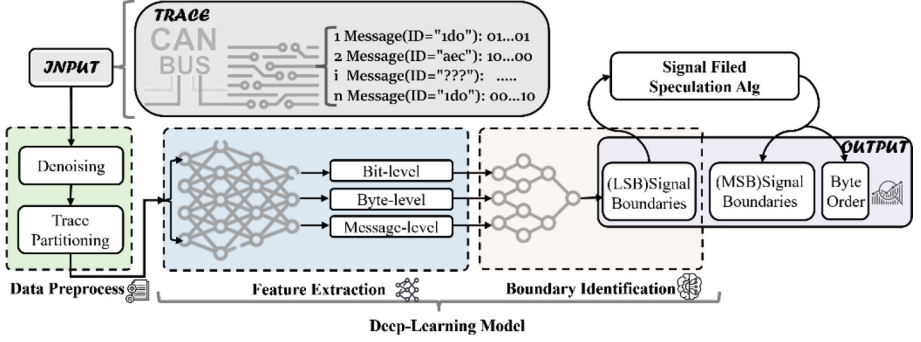


Fig. 5. Overall workflow of AutoDBC

3.3 Data Pre-processing

Denoising

The dataset used in this study is collected from real-world vehicle operations, where certain vehicle operations are rarely triggered by users. These operation traces lack discernible features that can be analyzed and identified. To prevent these featureless data from affecting the model's learning, we employ multiple denoising techniques, as described below:

Static Bit Filtering: Certain signals, such as those related to rarely triggered events (e.g., airbag deployment, emergency call signals), remain static for most of the dataset. Static bits do not contribute useful information for signal boundary identification and can distort the feature extraction process. Therefore, we apply a bit-flipping rate analysis to filter out signals where the bit-flip rate is zero across the entire trace. This process involves calculating the bit-flip rate R_b for each bit b as:

$$R_b = \frac{\sum_{i=1}^N 1(x_i \neq x_{i-1})}{N}$$

where x_i is the value of bit b at time i , and N is the total number of messages in the trace. If $R_b = 0$, the bit is considered static and removed from further analysis.

Low Variance Filtering: In addition to static bits, some signals may have very low variance, meaning they change infrequently and provide little useful information for signal boundary detection. We compute the variance for each signal across the dataset and apply a threshold-based filter to remove signals with a variance below a defined threshold:

$$\text{Var}(x) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$$

where x_i is the value of the signal at time i and $\bar{x}$ is the mean of the signal. Low-variance signals are considered noise and are discarded.

Outlier Detection and Removal: Outliers can occur in CAN data due to sensor malfunctions, communication errors, or other anomalous events. We apply the Z-score statistical method to detect and remove outliers. The Z-score method standardizes the signal values and removes values that deviate beyond a set threshold (e.g., $Z > 3$):

$$Z = \frac{x_i - \mu}{\sigma}$$

where μ and σ are the mean and standard deviation of the signal. The IQR method removes values that fall outside the range of $Q1 - 1.5 \times IQR$ to $Q3 + 1.5 \times IQR$, where $Q1$ and $Q3$ are the first and third quartiles, respectively.

Trace Partitioning

After denoising the data, the next step is to partition the CAN traces into manageable segments for model training and evaluation. The partitioning step not only improves the computational efficiency but also enhances the model's ability to generalize, as it learns from smaller, more focused sequences of CAN messages.

Time-Series Partitioning: In detail, to ensure that partitioning maintains the temporal dependencies between messages, we divide each trace with the same MID into sub-traces, with each sub-trace containing 1000 messages. Each sub-trace preserves the sequence of messages, allowing the model to capture temporal relationships between signals in the CAN data. This is critical for accurately identifying signal boundaries that may be influenced by previous messages in the sequence, while also preventing data overfitting by balancing the number of examples seen by the model.

3.4 Feature Extracting

Model Inputs

After the preprocessing and denoising stages, AutoDBC ensures that the raw CAN data is clean, relevant, and transformed into structured formats suitable for deep learning-based feature extraction and signal boundary identification. For each partitioned trace of CAN messages, we represent the data as a sequence of 64-bit vectors, where each vector corresponds to the data field of one CAN message. Based on these, we defined two inputs for AutoDBC as below:

Input 1 (Matrix Inputs): Each message is transformed into a binary matrix M_m of dimensions 8×8 , where each row corresponds to one byte of the CAN message:

$$M_m = \begin{bmatrix} b_{00} & b_{01} & \cdots & b_{07} \\ b_{10} & b_{11} & \cdots & b_{17} \\ \vdots & \vdots & \ddots & \vdots \\ b_{70} & b_{71} & \cdots & b_{77} \end{bmatrix}$$

Here, $b_{ij} \in \{0, 1\}$ represents the j -th bit in the i -th byte of the message. This matrix serves as the bit-level (details as **Bit-Level Features**) input and byte-level (details as **Byte-Level Features**) input for the model, allowing the convolutional layers to operate directly on the raw bit information.

Input 2 (Vector Inputs): At the message level (details as **Message-Level Inputs**), each CAN message is treated as part of a larger trace, with messages grouped together by their MIDs. These grouped message sequences are fed into the model as a time-series of 64-bit vectors, where each vector represents one CAN message. For example, a sequence of 1000 messages is represented as a matrix $\mathbf{T} \in \mathbb{R}^{1000 \times 64}$, where each row corresponds to a single message:

$$\mathbf{T} = \begin{bmatrix} m_1 & m_2 & \cdots & m_{64} \\ m_1 & m_2 & \cdots & m_{64} \\ \vdots & \vdots & \ddots & \vdots \\ m_1 & m_2 & \cdots & m_{64} \end{bmatrix}$$

This section describes the process of extracting features from the clean, partitioned CAN message traces at different levels: bit-level, byte-level, and message-level, as well as how these features are utilized by the deep learning identification model for effective signal boundary identification.

Bit-level Features

The bit-level extracting focuses on the smallest granularity of the CAN message, treating each individual bit as an essential component for feature extraction.

During the bit-level feature extracting, the bit-level input (**Input 1**) is processed using 3x3 convolutional kernels to capture the local spatial interactions between adjacent bits, focusing on the patterns like bit flipping, which often indicates signal boundaries. These kernels slide across the matrix to generate feature maps for each input message.

Byte-level Features

While bit-level features provide fine-grained detail, byte-level features are critical for capturing the interactions and dependencies between groups of bits, particularly for signals that span multiple bytes in the CAN data field.

At the byte level, the 64-bit message vector is treated as eight 8-bit bytes, and relationships between these bytes are analyzed to capture signal boundaries and patterns that occur across multiple bytes. Mixed-endian byte orderings present in CAN data further complicate this process, making byte-level features critical for accurate signal boundary identification.

The byte-level input is represented by the same M_m matrix (**Input 1**), but the convolutional kernels are larger, typically 8×8 , to capture cross-byte dependencies. These kernels are applied across entire bytes of data, allowing the model to learn higher-order features that span the bit boundaries within a byte and across adjacent bytes.

Byte-level features help the model detect situations where signals are split across bytes, with different parts of the signal using different byte orders (big-endian or little-endian). This capability is crucial for accurately interpreting real-world CAN data, where mixed-endian formats are common.

Message-Level Inputs

Message-level inputs (**Input 2**) account for the temporal nature of CAN communication, where signals are transmitted in a sequence of messages over time. Since many

signals exhibit periodic behavior or have dependencies on previous messages, extracting message-level features allows the model to capture such long-range dependencies.

In the deep learning model, these time-series inputs are processed using convolutional filters with sizes such as 3×3 , which operate across consecutive messages to capture temporal relationships. This approach enhances the model's ability to predict the signal boundaries in real-time communication, even when signals exhibit periodicity or depend on contextual information from earlier messages.

3.5 Boundary Identification

After processing the bit-, byte-, and message-level inputs, the three sets of features are combined to form a comprehensive representation of the CAN message data. The feature maps from each input type are concatenated along the channel dimension, resulting in a multi-channel input that is fed into deeper layers of the CNN.

The final input representation is structured as follows:

$$\text{Input} = [x_{\text{bit}}, x_{\text{byte}}, x_{\text{message}}]$$

where x_{bit} , x_{byte} , and x_{message} are the feature maps extracted from the bit-level, byte-level, and message-level inputs, respectively. This multi-channel input ensures that the model has access to all levels of granularity in the CAN data, improving the accuracy of signal boundary identification.

The combination of bit-, byte-, and message-level features allows the model to capture both local and global patterns, making it robust to variations in data across different vehicle models and driving conditions. By leveraging this multi-level trace features, AutoDBC uses neural network and sigmoid classifier to identify signal boundaries effectively with high accuracy.

3.6 Model Structure

To effectively capture the spatial-temporal relationships in the data field of CAN frames/messages, we adopt a custom Convolutional Neural Network (CNN) model that includes convolutional and pooling layers, coupled with Residual connections to preserve input information and facilitate gradient propagation. Additionally, batch normalization and dropout techniques are applied to accelerate convergence. The model structure is depicted in Fig. 6.

In the AutoDBC model, three sets of convolutional layers target the Bit, Byte, and Message levels, respectively. A 3×3 kernel is used to extract bit-level features, an 8×8 kernel captures byte-level relationships, and another 3×3 kernel extracts message-level features within the ID trace.

We denote the convolution kernels' filter size (fs) for bit-level feature extraction as fs_1 , byte-level feature extraction as fs_2 , and message-level feature extraction as fs_3 . The feature data obtained from Input 1 after bit-level and byte-level feature extraction are denoted as x_1 and x_2 , respectively, both having the same shape. x_1 and x_2 are residually connected to Input 1 to refine the feature extraction data. The feature data obtained from the message-level feature extraction of Input 2 is denoted as x_3 . x_3 is residually

connected to Input 2 data to further refine the extracted features. After applying residual connections, pooling layers process x_1 , x_2 , and x_3 to average features at the same bit level. The specific operation is defined in Eq. 3:

$$\text{Pool}(X) = \begin{cases} \text{Mean}(X \cdot \text{Reshape}(l \times 8 \times 8, \text{shape}), \text{axis} = 0), \\ \quad \text{shape} = \text{shape 1} \\ \text{Mean}(X \cdot \text{Reshape}(l \times 8 \times 8, \text{shape}), \text{axis} = 0), \\ \quad \text{shape} = \text{shape 2} \\ \text{Mean}(X \cdot \text{Reshape}(l \times 64, \text{shape}), \text{axis} = 0), \\ \quad \text{shape} = \text{shape 3} \end{cases} \quad (3)$$

Here, the Reshape (a, b) function converts the data from shape b to shape a . The Mean ($x, \text{axis} = 0$) function calculates the mean of x along the 0 axis to compress x .

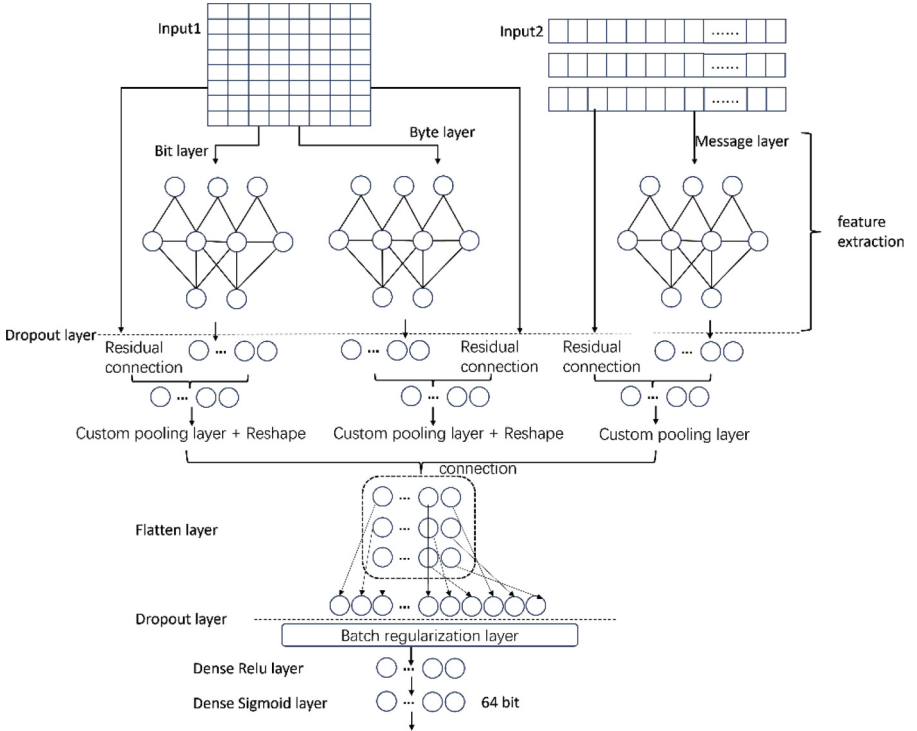


Fig. 6. Deep learning network architecture of AutoDBC for signal boundary identification

After pooling, x_1 , x_2 , and x_3 are combined into new feature data. This data is regularized through a batch normalization layer to prevent overfitting. The data is then compressed into a vector with a length of 64 by the perceptron, and finally, a 64-bit multi-label classification is performed by the perceptron classifier.

3.7 Model Structure

The Least Significant Bit (LSB) is the lowest bit of each signal and contains detailed bit features that assist in identifying the signal's lower boundary, such as bit flip rates, which are analyzed through Signal Boundary Identification. In contrast, the Most Significant Bit (MSB) is the highest bit of each signal and is typically set to 0, exerting minimal influence on signal boundary identification. Therefore, we define the correct signal field as the scanning region where one LSB is accurately identified, with no additional LSBs detected while scanning the message data in the LM direction until the bit flip rate of one bit reaches 0. The LM direction is characterized by a decreasing bit flip rate from the LSB, which indicates the signal byte order. Based on the identified LSB, speculated MSB, and determined byte order, the signal field can be ascertained (Signal Field Identification). As outlined in Algorithm 1, the process begins with the initialization of key parameters, including the bit flip rates for the signal array and the position of the subsequent LSB. The process then progresses through the big-endian scanning method (described in Algorithm 2) and the little-endian scanning method (detailed in Algorithm 3).

Algorithm 1 Find MSB from LSB

```

1: Input: Signal array signal [], LSB position LSB_pos
2: Output: MSB position
3:   rates  $\leftarrow$  CALCULATEBITFLIPRATES(signal[])
4:   next_LSB  $\leftarrow$  GETNEXTLSB(LSB_pos) ▷ Determine next LSB
   position
5:   MSB_pos_big  $\leftarrow$  LSB_pos, MSB_pos_little  $\leftarrow$  LSB_pos
6:   BIGENDIANSCAN (signal [], rates, LSB_pos, next_LSB)
7:   LITTLEENDIANSCAN (signal [], rates, LSB_pos, next_LSB)
8:   return max (MSB_pos_big, MSB_pos_little)

```

Algorithm 2 Big-Endian MSB Calculation

```

1: procedure BIGENDIANSCAN (signal [], rates, LSB_pos, next_LSB)
2:   MSB_pos_big  $\leftarrow$  LSB_pos
3:   while MSB_pos_big > 0 and MSB_pos_big  $\neq$  next_LSB do
4:     next_pos_big  $\leftarrow$  CALCULATENEXTPOSITION(signal[], MSB_pos_big, big-
   endian)
5:     if rates [next_pos_big] < rates [MSB_pos_big] and rates [next_pos_big] = 0 then
6:       MSB_pos_big  $\leftarrow$  next_pos_big
7:     break
8:   end if
9:   MSB_pos_big  $\leftarrow$  next_pos_big
10: end while
11: end procedure

```

Algorithm 3 Little-Endian MSB Calculation

```

1: procedure LITTLEENDIANSCAN (signal [], rates, LSB_pos, next_LSB)
2:   MSB_pos_little  $\leftarrow$  LSB_pos
3:   while MSB_pos_little < length(signal[]) - 1 and MSB_pos_little  $\neq$  next_LSB do
4:     next_pos_little  $\leftarrow$  CALCULATENEXTPOSITION(signal[], MSB_pos_little, little-
endian)
5:     if rates [next_pos_little] < rates [MSB_pos_little] and rates [next_pos_little] = 0
then
6:       MSB_pos_little  $\leftarrow$  next_pos_little
7:     break
8:     end if
9:     MSB_pos_little  $\leftarrow$  next_pos_little
10:  end while
11:  end procedure

```

4 Implementation and Evaluation

4.1 Implementation

We implemented AutoDBC in Python 3.7 using the following libraries: pandas 1.3.5, numpy 1.19.3, keras 2.2.4, tensorflow 1.13.1, and scikit-learn 1.0.2. For comparison, we re-implemented three state-of-the-art (SOTA) methods-READ [9], LibreCAN [10], and CAN-D [11]-under the same experimental settings.

Datasets and Ground-Truth. Most existing studies require data collection from controlled experiments with specific vehicle driving states [12, 13] for model training and testing. In our study, we relaxed this requirement by using CANedge1 to randomly collect raw CAN data from three personal vehicles over different time periods across two weeks, as shown in Fig. 7. The statistics of the collected data, amounting to a substantial 7.9 gigabytes, are shown in Table 1, where we used the Honda and Cadillac datasets for training and the Chevy dataset for testing (each trace contains 1,000 messages). We utilized DBC files from CommaAI’s OpenDBC project for labeling. **All code and data are available on GitHub:** <https://github.com/ZH21/AutoDBC.git>.

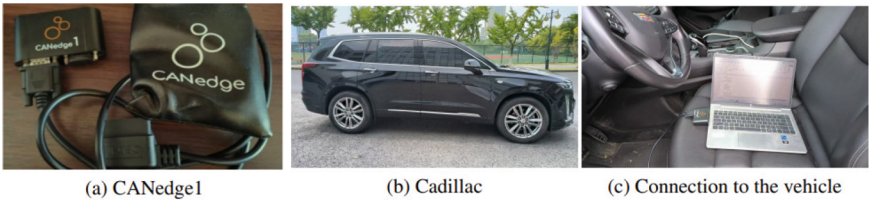


Fig. 7. Experiments were conducted with CANedge1 and real vehicles.

Model Parameters. The optimal parameters for our model are as follows: the CNN filter sizes (fs) for fs_1 , fs_2 and fs_3 are all set to 15. The optimal training parameters are

Table 1. Statistics of Collected Raw CAN Data

Vehicle	Datasets	Message Count	Traces
Honda Civic (2015)	7	1,509,141	1,492
Chevy Equinox (2014)	5	1,512,802	1,479
Cadillac XT6 (2019)	21	1,521,400	1,473

a learning rate (lr) of 0.0001, a batch size (bs) of 32, and a dropout rate (dr) of 0.1. The optimizer used is Adam. Both $input_1$ and $input_2$ have a shape of $bs \times 8000 \times 8 \times 15$, while $input_3$ (corresponding to Input 2) has a shape of $bs \times 1000 \times 64 \times 15$.

Model Inference Details. The original data for $input_1$, $input_2$, and $input_3$ are connected through residual connections with their extracted features x_1 , x_2 , and x_3 , respectively. The data for $input_1$ and $input_2$ are reshaped to $bs \times 1000 \times 8 \times 8 \times 15$ and averaged along axis 1 (with the message as the basic unit) to aggregate the features of each message across the entire sub-trace, resulting in feature shapes of $bs \times 1 \times 8 \times 8 \times 15$. Each feature is then unified into the shape $bs \times 64 \times 15$. The three features are combined into a single feature of shape $bs \times 64 \times 45$, which is then flattened into a one-dimensional vector. This vector passes through a batch normalization layer, followed by a perceptron layer with a ReLU activation function, which compresses it to $bs \times 64$. Finally, a Sigmoid perceptron classifier performs multi-label classification for the 64-bit outputs.

Research Questions The following questions will be addressed in subsequent sections:

- **Q1:** Do different signal byte orders need to be considered in signal boundary identification tasks? This question examines whether the use of big-endian and little-endian sequences is consistent across different messages.
- **Q2:** What is the performance of AutoDBC in LSB signal boundary recognition? This involves evaluating the performance of AutoDBC in comparison to other algorithms in the task of signal boundary identification.
- **Q3:** How effective is AutoDBC in signal field identification? This question assesses the performance of AutoDBC when paired with the signal field speculation algorithm, and evaluates how the identified signal boundaries influence signal field speculation compared to other algorithms.

4.2 DBC Files Analysis - Q1

To investigate the impact of signal byte order on the identification of CAN bus signal boundaries in the automotive industry, we analyzed the signal byte order across all DBC files from the CommaAI open-source DBC repository, as illustrated in Fig. 8. Figure 8 (a) demonstrates that both little-endian and big-endian signals can appear within the same DBC file. Relying solely on a single byte order for signal boundary identification may significantly compromise accuracy across different vehicles. Figure 8 (b) indicates that while big-endian signals are approximately three times more prevalent than little-endian signals, the latter still represent a substantial number (7724 signals) and should not be disregarded in signal boundary identification research.

To address Research Question 1, it is essential to consider the effect of byte order in algorithms aimed at achieving universal and accurate signal boundary identification.

4.3 Performance of Signal Boundary Identification - Q2

We evaluated the performance of READ, LibreCAN (Phase0), CAN-D (Step1), and our proposed AutoDBC using Precision, Recall, F1 score, and Accuracy (consistent with related work). LibreCAN (Phase0) and CAN-D (Step1) are the signal boundary identification modules of their respective approaches. The results are shown in Table 2.

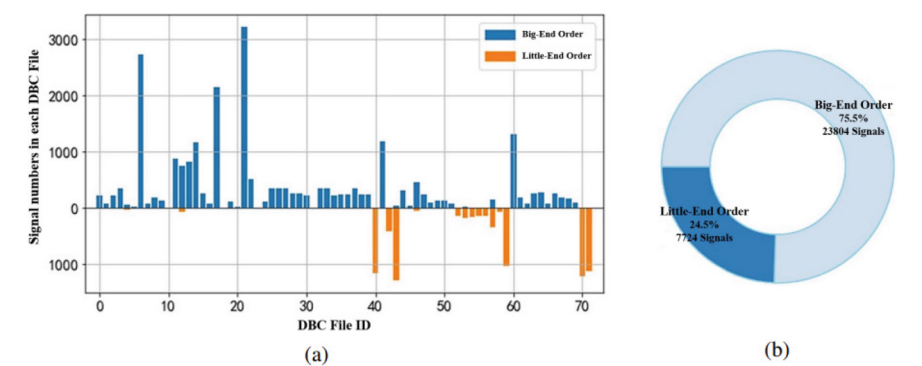


Fig. 8. Signal Byte Order Statistics of CommaAI OpenDBC Repository 72 DBC Files

Method	Precision	Recall	F1-score	Accuracy
READ [9]	0.37	0.62	0.47	0.96
LibreCAN [10]	0.50	0.43	0.46	0.97
CAN-D [11]	0.51	0.62	0.56	0.97
AutoDBC	0.63	0.69	0.66	0.98

The performance of the four methods is ranked as follows: AutoDBC > CAN – D(Step1) > LibreCAN(Phase0) ≈ READ. Furthermore, we observed that the F1 scores for all four methods are relatively low. This may be attributed to two factors: 1) The ground truth data obtained from CommaAI’s DBC files corresponds to older vehicle models, which may not be entirely accurate; 2) The dataset used for testing was collected from real-world driving scenarios, where the features are less distinct and comprehensive compared to those collected under controlled conditions [9, 11].

Regarding Research Question 2, AutoDBC has demonstrated strong capabilities in signal boundary identification.

4.4 Performance of Signal Field Identification -Q3

Table 3 presents the results of signal field identification for READ, LibreCAN (Phase0), CAN-D (Step1, Step2), and AutoDBC. In the table, the number of identified signal fields correctly is as CE. The number of signal fields identified by a model corresponds to the number of recognized LSB boundaries (as TE). The number of DBC signal fields equals the number of actual signals recorded in the DBC file (as TDBC). CE/TE represents the precision of the model in identifying signal fields; the larger the value, the more accurate the model is in recognizing the correct signal fields. CE/TDBC indicates the capability to correctly detect signal fields; the larger the value, the more accurately the signal fields can be retrieved.

Table 3. Signal field Identification Test Results

Model	Correctly Identified LSB Signal Boundaries	Correctly Identified Fields (CE)	Total Identified Fields (TE)	DBC Signal Fields (TDBC)	CE/TE	CE/TDBC
READ [9]	94	88	252	151	34.92%	58.28%
LibreCAN [10]	65	63	130	151	48.46%	41.72%
CAN-D [11]	94	93	178	151	52.25%	61.59%
AutoDBC	104	102	166	151	61.45%	67.55%

The results show that the capability of the four methods to correctly identify signal fields is ranked as follows: AutoDBC > CAN – D (Step1, Step2) > LibreCAN (Phase0) > READ. This ranking suggests that AutoDBC performs more accurately in identifying signal fields compared to the other methods, thereby addressing Research Question 3.

4.5 Case Study

We qualitatively analyze the AutoDBC signal field identification results compared with CAN-D (Step1, Step2), which has a stronger ability to identify the positions of the signal than READ and LibreCAN. We take Trace (Message_ID = 0x1a1, Name = Accelerator-Pedal) (as Table 4) and Trace (Message_ID = 0x348, Name = EBCMWheelSpdFront) (as Table 5) as the analysis examples, and study the effect difference of results between the two methods in identifying the signal fields. The results are represented in Fig. 9.

In Fig. 9, the Label, AutoDBC and CAN-D (Step1, Step2) bars are composed of 64 small rectangles, which represent the signal domains identified by the DBC detector, AutoDBC and the CAN-D (Step1, Step2), respectively. The red vertical lines indicate the ground truth of signal boundaries. The small dark blue rectangles indicate the non-signal regions in messages, and the other colored rectangles indicate the different signal fields identified by the methods. Among them, Sub-Figure (a) represents the results

Table 4. AcceleratorPedal Message signal information

signal name	start bit	signal length
AcceleratorPedal	55	8

Table 5. EBCMWheelSpdFront Message signal information

signal name	start bit	signal length
FLWheelSpd	7	16
FRWheelSpd	23	16

of the signal field identification of the CAN-D, AutoDBC, Label (Ground Truth) for AcceleratorPedal Message, and Sub-Figure (b) represents the result of the identification results of two methods and Label for EBCMWheelSpdFront Message.

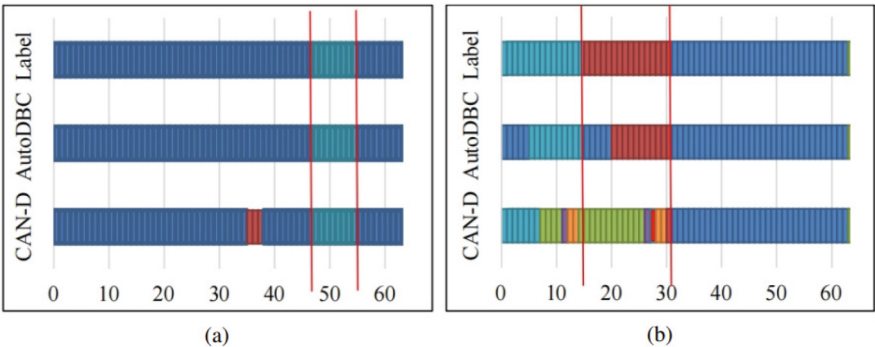


Fig. 9. Qualitative results of signal field identification

From Fig. 9, it can be seen that AutoDBC can identify the signal fields of AcceleratorPedal and EBCMWheelSpdFront Message more accurately, while the CAN-D (Step1, Step2) identifies many redundant CAN-D AutoDBC Label CAN-D AutoDBC Label signals, which is consistent with the analysis of the results of 2. Since the nature of the signals' MSB, both methods are unable to accurately determine it.

However, the results in Fig. 9 still shows that the ability of AutoDBC to identify the signal boundary more accurately greatly improves the ability to identify the signal fields, which is conducive to the subsequent reverse extraction of the signal attributes.

4.6 Related Work

Jaynes et al. [5] were the first to address the automation challenge of CAN bus reverse engineering, designing a machine learning classifier to learn the overall features of CAN

data. Markovitz et al. [6] was the first to design a CAN bus data domain segmentation algorithm and to provide a rough classification of the segmented data domains. Nolan et al. [7] studied labeling algorithms by processing bit flipping counts, establishing this approach as a mainstream research method in CAN signal labeling algorithms in recent years. The READ algorithm [9] was proposed in 2019 to identify the position and type of each signal within each CAN message. This algorithm, based on certain assumptions and artificially defined thresholds, cannot be strictly equivalent to facts, introducing a degree of subjectivity. However, due to its enhanced identification ability, reduced time requirement, and significantly improved signal extraction accuracy compared to previous algorithms, it has become a commonly used reference and comparison algorithm for subsequent research. In addition, Huybrechts et al. [14] proposed two different signal information identification methods based on OBD-II and GPS data, which, to some extent, limited the scalability of the model. Verma et al. [15] introduced ACTT technology, which matches all possible positions of signals within CAN frames with real-time diagnostic data obtained from OBD-II to achieve automatic labeling and translation based on the READ algorithm. [10] proposed LibreCAN, which improves the READ algorithm for signal labeling, converts power signals based on the correlation between CAN data and external data, and uses semi-automatic programs to determine body signals. [16] were the first to introduce clustering in CAN bus reverse engineering to identify CAN signals related to the same vehicle function. [11] proposed the CAN-D algorithm, which for the first time fully parses all information related to CAN signal boundaries, byte order, symbols, and physical meanings. [17] is an automated reverse-engineering system that achieves high-precision slicing and labeling of automotive CAN signals through byte-level clustering and bit-level feature analysis, but it has limitations, including challenges in real-time performance, generalization ability, unknown signal recognition, and computational resource requirements. [18] proposes a method for reverse-engineering automotive CAN messages through marked bits and fuzz testing to assist with vehicle safety and intrusion detection, but it is limited by its reliance on CAN database files and the constraints of data collection. Deep learning frameworks have also been widely applied in other research related to CAN bus [19-21].

5 Conclusion

In conclusion, our paper introduces AutoDBC, a pioneering algorithm for the intelligent segmentation of CAN-Bus data. This algorithm excels at extracting features at the bit-, byte-, and message-level, which is essential for effectively capturing the characteristics of signals with both big-endian and little-endian byte orders. We have developed an innovative deep-learning model specifically tailored for AutoDBC to perform end-to-end identification of signal boundaries and fields, a critical task in CAN data analysis.

Our extensive testing and evaluation on real vehicle data have shown that AutoDBC significantly outperforms current methods. It achieves higher accuracy in signal boundary and signal field identification, a testament to the effectiveness of our approach. This advancement is particularly valuable for applications in automotive diagnostics, monitoring, and the advancement of Advanced Driver-Assistance Systems (ADAS).

Overall, AutoDBC's ability to handle the complexities of CAN-Bus data with precision makes it a promising tool for the automotive industry, enhancing the potential for more accurate and efficient data analysis in vehicle systems.

References

1. Bertongello, M., et al.: Monetizing car data-new service business opportunities to create new customer benefits. McKinsey & Company, pp. 1–60 (2016)
2. Qian, H., Han, H., Zhu, X., Xu, F.: Shufflecan: enabling moving target defense for attack mitigation on automotive can. In: 2023 19th International Conference on Mobility, Sensing and Networking (MSN). IEEE, pp. 351–358 (2023)
3. Quigley, C., Charles, D., McLaughlin, R.: Can bus message electrical signatures for automotive reverse engineering, benchmarking and rogue ecu detection. SAE Technical Paper, Tech. Rep. (2019)
4. Buscemi, A., Turcanu, I., Castignani, G., Panchenko, A., Engel, T., Shin, K.G.: A survey on controller area network reverse engineering. IEEE Communications Surveys & Tutorials **25**(3), 1445–1481 (2023)
5. Jaynes, M., Dantu, R., Varriale, R., Evans, N.: Automating ecu identification for vehicle security. In: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp. 632–635 (2016)
6. Markovitz, M., Wool, A.: Field classification, modeling and anomaly detection in unknown can bus networks. Vehicular Commun. **9**, 43–52 (2017)
7. Nolan, B.C., Graham, S., Mullins, B., Kabban, C.S.: Unsupervised time series extraction from controller area network payloads. In: 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall). IEEE, pp. 1–5 (2018)
8. Szydlowski, C.P.: Can specification 2.0: Protocol and implementations. SAE Technical Paper, Tech. Rep. (1992)
9. Marchetti, M., Stabili, D.: Read: reverse engineering of automotive data frames. IEEE Trans. Inf. Forensics Secur. **14**(4), 1083–1097 (2018)
10. Pesé, M.D., Stacer, T., Campos, C.A., Newberry, E., Chen, D., Shin, K.G.: Librecan: automated can message translator. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 2283–2300 (2019)
11. Verma, M.E., Bridges, R.A., Sosnowski, J.J., Hollifield, S.C., Iannacone, M.D.: Can-d: a modular four-step pipeline for comprehensively decoding controller area network data. IEEE Trans. Veh. Technol. **70**(10), 9685–9700 (2021)
12. Koscher, K., et al.: IEEE symposium on security and privacy. IEEE **2010**, 447–462 (2010)
13. Miller, C., Valasek, C.: Adventures in automotive networks and control units. Def Con **21**(260–264), 15–31 (2013)
14. Huybrechts, T., Vanommeslaeghe, Y., Blontrock, D., Van Barel, G., Hellinckx, P.: Automatic reverse engineering of can bus data using machine learning techniques. In: Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 12th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2017). Springer, pp. 751–761 (2018)
15. Verma, M., Bridges, R., Hollifield, S.: Actt: automotive can tokenization and translation. In: 2018 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE, pp. 278–283 (2018)
16. Ezeobi, U., Olufowobi, H., Young, C., Zambreno, J., Bloom, G.: Reverse engineering controller area network messages using unsupervised machine learning. IEEE Consumer Electronics Magazine **11**(1), 50–56 (2020)

17. Lin, X., et al.: Bycan: reverse engineering controller area network (can) messages from bit to byte level. *IEEE Internet of Things Journal* (2024)
18. Bi, Z., Xu, G., Wang, C., Xu, G., Zhang, S.: A method for translating automotive body-related can messages based on labeled bits. *Appl. Sci.* **13**(3), 1942 (2023)
19. Salunkhe, S.S., Pal, S., Agrawal, A., Rai, R., Mole, S.S., Jos, B.M.: Energy optimization for can bus and media controls in electric vehicles using deep learning algorithms. *The Journal of Supercomputing*, pp. 1–16 (2022)
20. Zhang, G., Li, Y.: Voltage inspector: Sender identification for in-vehicle can bus using voltage slice. *Comput. Secur.* **145**, 104017 (2024)
21. Hossain, M.D., Inoue, H., Ochiai, H., Fall, D., Kadobayashi, Y.: Long short-term memory-based intrusion detection system for in-vehicle controller area network bus. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, pp. 10–17 (2020)



Transfer Learning-Driven Surrogate-Assisted Differential Evolution Algorithm with User Generated Content

Yumeng Peng¹, Lin Bao¹(✉), Xiaoyan Sun², Jun Hao¹, Yu Wang¹, and Hang Yang¹

¹ Jiangsu University of Science and Technology, Zhenjiang 212100, Jiangsu, China
baolin_zj@163.com

² China University of Mining and Technology, Xuzhou 221116, Jiangsu, China

Abstract. For vast amounts of data and diverse user needs, accurately customizing search results and delivering personalized content to enhance user experience and platform profitability remain critical challenges in the field of personalized services. From the perspective of dynamic optimization, this paper regards a personalized search and recommendation task as a dynamic qualitative index optimization problem and proposes a transfer learning-driven surrogate-assisted differential evolution algorithm with user generated contents. The proposed solution begins by preprocessing and vectorizing multi-source heterogeneous user-generated content, where textual data is vectorized using a pre-trained language model. Next, convolutional neural networks (CNNs) and singular value decomposition (SVD) are combined to extract user preference and item attribute features, leveraging transfer learning to use them as fixed feature extractors, thereby enhancing feature extraction efficiency in dynamic environments. Subsequently, an attention factorization machine is employed to interact with these extracted features, constructing a surrogate model based on user preferences to approximate the objective function of the recommendation task. Finally, the surrogate model is integrated with a differential evolution algorithm. Within the differential evolution framework, specific evolutionary operators, strategies, and a dynamic model management mechanism are introduced to adapt the evolutionary search process based on user preferences. New evolutionary individuals are generated iteratively with detailed user preference information, refining the recommendation list and enhancing personalization accuracy in each iteration. Extensive experiments on several practical, commonly used real-world datasets verify the feasibility and effectiveness of the proposed method.

Keywords: personalized recommendation · multi-source heterogeneous data · transfer learning · differential evolution algorithm · surrogate model

1 Introduction

With the rapid development of online services and mobile applications, the internet is inundated with a vast and complex array of multi-source heterogeneous information. This provides users with a rich variety of new information while also increasing the difficulty

of filtering out useful information [1]. Personalized search and recommendation are methods of information filtering designed to help users quickly find content that meets their needs and to assist internet-related companies in improving economic efficiency and service quality [2–7].

In personalized search and recommendation, researchers leverage user-generated content—such as reviews, item tags, and ratings—to extract user interest and preference information. Khan et al. [8] introduced a semantic-based item content embedding model that utilizes convolutional neural networks (CNN) to extract contextual features and non-negative matrix factorization (NMF) to address sparsity and negative value loss. Dezfouli et al. [9] developed the Match Pyramid Recommender System (MPRS), which constructs a matching matrix from users' text reviews and employs an interaction-based approach to compute matching scores for user-item pairs via CNN. Li et al. [10] represented items as word sequences and users' item interactions as sentence sequences, designing a bidirectional Transformer model to encode these sequences and fine-tuning the model for specific recommendation tasks, thereby proposing a method based on language representation learning. While these approaches enhance recommendation accuracy and effectiveness, they fall short in addressing dynamic personalized search and recommendation tasks within big data environments, as they primarily rely on historical user behaviors.

In practical applications, user interactions with platforms will generate new user generated contents, which has a crucial impact on subsequent personalized search and recommendation tasks [11, 12]. Existing researches have shown that a personalized search and recommendation can be viewed as a dynamic qualitative metric optimization problem. By combining surrogate models with evolutionary optimization algorithms, new interaction data can be effectively utilized [13–16].

In addressing the challenges of dynamic personalized search and recommendation in big data environments, this paper proposes a transfer learning-driven surrogate-assisted differential evolution algorithm with user generated contents. First, a natural language model is used to map user text reviews into a high-dimensional semantic space, generating vectorized representations of the reviews. Simultaneously, project attribute information is encoded into vectorized representations. Next, utilizing user preferences and project attribute features, a user preference surrogate model is constructed and trained based on transfer learning techniques, serving as a substitute for user ratings. Then, within the framework of a differential evolution algorithm, appropriate evolutionary operators and strategies are designed to generate new evolved individuals that incorporate user preference information. The surrogate model is used to estimate the fitness of these individuals, guiding the direction of personalized evolutionary search. Finally, a management mechanism for the dynamic model is designed to update and optimize the user preference surrogate model based on newly generated user contents, assisting users in quickly finding satisfactory solutions and efficiently completing personalized search and recommendation tasks. Extensive experiments on real-world general datasets validate the feasibility and effective-ness of the proposed algorithm.

The main contributions of this paper are threefold:

- (1) The problem of personalized search and recommendation is regarded as a dynamic qualitative indicator optimization task, leading to the design of a data-driven

surrogate-assisted differential evolution algorithm. This approach, grounded in evolutionary optimization, integrates deep learning methods and transfer learning techniques within the differential evolution framework;

- (2) To effectively extract and interact user preference features with item attributes, a user preference surrogate model is developed by combining convolutional neural networks with attention-based factorization machines, actively participating in the personalized search process;
- (3) A dynamic model management mechanism is designed to allow the surrogate model to retain effective parameter information from previous optimizations while rapidly updating with new data. This ensures the model adapts to user preferences and behaviors in real-world big data environments, enhancing the efficiency and effectiveness of personalized search and recommendation.

The remainder of this study is organized as follows. Section 2 presents the related work. The proposed algorithm is described in detail in Sect. 3. Section 4 shows the experimental results and analysis. Finally, the conclusion is followed.

2 Related Work

2.1 Data-Driven Optimization Problems

For a personalized search and recommendation problem, its objective function usually cannot be described by a precise mathematical function. Furthermore, the qualitative characteristics of users' preferences and items' attributes are difficult to quantify and these features may exhibit different patterns in different times and situations. To address this challenge, researchers have proposed to combine data-driven surrogate models with evolutionary optimization algorithms to handle the complex dynamic qualitative index optimization problems such as personalized search and recommendation [17].

The mathematical formulation of the dynamic personalized search and recommendation problem can be represented as follows:

$$\begin{cases} \max \sum f^t(x, u) \\ s.t. u \in U, x \in \Omega^t \end{cases} \quad (1)$$

where, $U = \{u_1, u_2, \dots, u_{|U|}\}$ represents the set of users, $|U|$ is the number of users; $\mathbf{x} = (x_1, \dots, x_{|\mathbf{x}|})$ is the feature vector of items, with the feature dimension of $|\mathbf{x}|$; Ω^t denotes the search space at time t ; $t \in 1, \dots, T$ represents discrete time steps and $T > 1$ is the total number of time steps; The objective function $f^t(\mathbf{x}, u)$ represents the user preferences at time t . Since user preferences are dynamic and difficult to fit with mathematical functions, it is necessary to construct a data-driven surrogate model for the personalized search and recommendation problem.

Current research on data-driven surrogate models primarily addresses static objective functions in continuous spaces, with dynamic objective functions receiving limited attention. Notable contributions include Nyikosa et al. [18] introduced an adaptive Bayesian optimization using a time-dependent length scale, Richter et al. [19] proposed Bayesian methods incorporating temporal aspects, and Li et al. [20] presented a hierarchical multi-output Gaussian process for capturing correlations across time. While

these studies emphasize parameter-adaptive strategies to infer adjustments in objective functions through data analysis, they fall short in the context of personalized search and recommendation, which necessitates the effective utilization of multi-source heterogeneous user-generated content to capture high-dimensional features for dynamic optimization.

In this context, this paper will consider the full utilization of a large amount of user-generated content to design a dynamically updatable user preference surrogate model that learns and fits users' preferences based on the information flow across different time periods.

3 Algorithm Framework

The proposed transfer learning-driven surrogate-assisted differential evolution algorithm with user-generated content (SADE-DCAFM) algorithm consists of three main components: User preference surrogate model, Surrogate-assisted differential evolution algorithm and dynamic management mechanism. The basic framework of the proposed SADE-DCAFM is shown in Fig. 1.

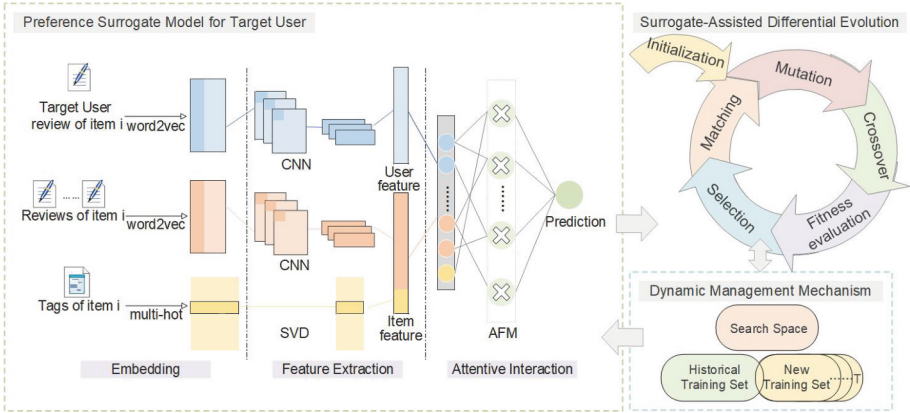


Fig. 1. Diagram of the SADE-DCAFM Algorithm

3.1 Constructing the User Preference Surrogate Model

Vectorization of User-Generated Content

For user text reviews, data preprocessing is performed. This process results in an effective set of words. Subsequently, the Word2Vec model is utilized to convert the words in the effective set into corresponding word vectors, forming a word vector set $\mathbf{W}$, with each word assigned a unique index.

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_i, \dots, \mathbf{w}_N]^T, \mathbf{W} \in \mathbb{R}^{N \times d} \quad (2)$$

where N is the total number of words; d is the word embedding dimension; $\mathbf{w}_i = [w_{i1}, w_{i2}, w_{i3}, \dots, w_{id}]$ is the vectorized representation of the i -th word.

The category tags of item X are encoded in a binary vector $\mathbf{g}$ by Multi-hot.

$$\mathbf{g} = [g_1, g_2, g_3, \dots, g_r] \quad (3)$$

where r is the length of vector $\mathbf{g}$. If item X contains the j -th category tag, then the j -th element $g_j = 1$; otherwise, $g_j = 0$.

User Interest Preference Feature Extraction

Each text comment provided by user u is measured as a sequence of words. Each word is mapped to its corresponding word vector $\mathbf{w}$ through the indexes in the word vector set $\mathbf{W}$. Thus, each text comment of user is represented as the corresponding word vector matrix $\mathbf{S} \in \mathbb{R}^{d \times n}$.

$$\mathbf{S} = \begin{bmatrix} | & | & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_n \\ | & | & | \end{bmatrix} \quad (4)$$

where, d represents the number of matrix rows, corresponding to the word embedding dimension; n represents the number of matrix columns, corresponding to the number of word vectors. If a text comment contains more than n valid words, it is truncated; otherwise, it is zero-padded.

A user interest preference feature extraction model is developed using the word vector matrix $\mathbf{S}$ corresponding to the text comments of user u as input.

The convolution in the model is the local perception of input data at different locations in essence. The convolutional kernel slides along the column dimension of the input data $\mathbf{S}$ with a fixed step to perform the weighted sum in the different local and obtain the output component o_i ,

$$o_i = (\mathbf{S} \otimes \mathbf{E})_i = \sum_j (\mathbf{S}_{[:,i-m+1:i]} \otimes \mathbf{E})_j \quad (5)$$

where $\mathbf{E} \in \mathbb{R}^{d \times m}$ is the convolution kernel; the symbol $\otimes$ denotes the element-wise multiplication; $\mathbf{S}_{[:,i-m+1:i]}$ is a data matrix slice of length h and width m .

The above output component o_i is vertically concatenated to form a feature vector $\mathbf{c} \in \mathbb{R}^{1 \times (n-m+1)}$.

$$\mathbf{c} = [o_1, o_2, \dots, o_{n-m+1}]^T \quad (6)$$

Furtherly, the obtained feature vectors $\mathbf{c}$ from the convolutional are horizontally concatenated to form the feature map matrix $\mathbf{C} \in \mathbb{R}^{l \times (n-m+1)}$.

$$\mathbf{C} = \begin{bmatrix} | & | & | \\ \mathbf{c}_1 & \dots & \mathbf{c}_l \\ | & | & | \end{bmatrix} \quad (7)$$

where, l represents the number of convolution kernels.

In order to obtain a richer feature map representation, the user interest preference feature extraction model based on convolutional neural network utilizes three sets of convolution kernels with different width.

After the convolution operation, a feature vector $\mathbf{c}'$ is obtained to learn the nonlinear decision boundary by using a nonlinear activation function ReLU.

$$\begin{aligned}\alpha(x) &= \max\{0, x\} \\ o_i' &= \alpha(o_i) \\ \mathbf{c}' &= [o_1', o_2', \dots, o_{n-m+1}']^T\end{aligned}\quad (8)$$

Subsequently, the feature vector $\mathbf{c}'$ is transmitted to the pooling layer with the max-pooling to obtain the vector $\mathbf{c}^*$.

$$\begin{aligned}\text{pool}(\mathbf{c}') &: \mathbb{R}^{1 \times (n-m+1)} \rightarrow \mathbb{R} \\ \mathbf{c}^* &= \max\{o_1', o_2', \dots, o_{n-m+1}'\}\end{aligned}\quad (9)$$

Thus, the feature graph matrix with dimensionality reduction $\mathbf{C} \in \mathbb{R}^{l \times 1}$ is acquired.

$$\begin{aligned}\text{pool}(\mathbf{C}) &: \mathbb{R}^{l \times (n-m+1)} \rightarrow \mathbb{R}^{l \times 1} \\ \mathbf{C} &= [\mathbf{c}_1^*, \mathbf{c}_2^*, \dots, \mathbf{c}_l^*]\end{aligned}\quad (10)$$

Finally, the feature map matrix $\mathbf{C}$ is fed into a fully connected layer to obtain the user interest preference features $\mathbf{z}_u$.

$$\mathbf{z}_u = f(\mathbf{C} \times \mathbf{W} + \mathbf{b}) \quad (11)$$

where $\mathbf{W}$ is the weight matrix; $\mathbf{b}$ is the bias vector.

Item Attribute Features Extraction

Items' attribute features are extracted by the text comments and category labels of items. All the text comments of each item are treated as a word sequence, in which each word is mapped to a word vector w via its index in the word vector set $\mathbf{W}$. The corresponding text comments of each item are represented as a word vector matrix $\mathbf{S}$.

$$\mathbf{S} = \begin{bmatrix} | & | & | & | \\ \mathbf{w}_1 & \dots & \mathbf{w}_n \\ | & | & | & | \end{bmatrix} \quad (12)$$

An item text information feature extraction model based on CNN is constructed by using the corresponding word vector matrix $\mathbf{S}$ of item's text comments as the model input. The structure of this model is similar to the user interest preference feature extraction model based on CNN described in Sect. 3.1, and will not be detailed here. The item text information feature representation $\mathbf{z}_x^1$ with indexes is obtained by the trained item text information feature extraction model based on CNN.

Subsequently, the binary vector representations $\mathbf{g} = [g_1, g_2, \dots, g_r]$ of item's category labels are compressed into a k -dimensional space by SVD. The feature vector

representations $\mathbf{G}$ of the category labels of all the items are obtained.

$$\mathbf{G} = \begin{bmatrix} g'_{11} & g'_{12} & \cdots & g'_{1k} \\ g'_{21} & g'_{22} & \cdots & g'_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ g'_{x1} & g'_{x2} & \cdots & g'_{xk} \end{bmatrix} = \mathbf{U} \Sigma_k \mathbf{V}^T \quad (13)$$

Finally, the textual feature vector representation $\mathbf{z}_x^1$ of item $\mathbf{X}$ is concatenated with the feature vector representation $\mathbf{g}'$ of item's category labels. The vectorized representation of item's attribute features $\mathbf{z}_x = [\mathbf{z}_x^1, \mathbf{g}']$ is obtained.

User-Item Feature Interaction

The features $\mathbf{z} = (\mathbf{z}_u, \mathbf{z}_x)$ and user rating are fed into the attention-based Factorization Machine model, which leverages an automatic focusing and weighting mechanism to differentiate the importance of various user-item feature interactions.

The specific calculation is as follows:

- (1) Firstly, in the Pairwise Interaction Layer, the m vectors are expanded into $m(m-1)/2$ interaction vectors, constructing interaction vectors $\mathbf{p}_{ij}$ to encode the relationships between the vectors.

$$\mathbf{p}_{ij} = (\mathbf{v}_i \odot \mathbf{v}_j) z_i z_j \quad (14)$$

where, $\mathbf{v}_i$ and $\mathbf{v}_j$ are the latent vectors of feature i and feature j , respectively; z_i and z_j represent the vector factors corresponding to the i -th and j -th positions.

Subsequently, in the pooling layer incorporating the attention mechanism, the weights of each interaction vector are learned automatically, and the attention weight for each pairwise feature interaction is obtained by calculating the weighted sum.

$$\alpha_{ij} = \frac{\exp(\mathbf{h}^T \text{ReLU}(\mathbf{W} \mathbf{p}_{ij} + \mathbf{b}))}{\sum_{\delta=1}^m \sum_{\beta=\delta+1}^m \exp(\mathbf{h}^T \text{ReLU}(\mathbf{W} \mathbf{p}_{\delta\beta} + \mathbf{b}))} \quad (15)$$

where, α_{ij} denotes the attention weight corresponding to the interaction vector; $\mathbf{h}$ is the weight vector that maps the output value to the attention weight space; $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector of the attention mechanism, respectively.

Finally, the interaction vectors from the pairwise interaction layer are weighted and summed using the attention mechanism to produce the final model prediction.

$$\hat{y}_u(\mathbf{z}) = w_0 + \sum_{i=1}^n w_i z_i + \mathbf{p}^T \sum_{i=1}^n \sum_{j=i+1}^n \alpha_{ij} (\mathbf{v}_i \odot \mathbf{v}_j) z_i z_j \quad (16)$$

where, $\hat{y}_u(\mathbf{z})$ represents the predicted rating by the user preference surrogate model;

3.2 Surrogate-Assisted Differential Evolution Algorithm

The Differential Evolution (DE) algorithm optimizes a population through cooperation and competition among individuals over successive generations, thereby solving optimization problems [21–23]. Within the framework of the DE algorithm, this paper utilizes a data-driven user preference proxy model to estimate the fitness values of evolving individuals.

Through the item attribute features obtained in Sect. 3.1, the genotype of evolutionary individuals in the evolutionary optimization algorithm is initialized as X_i^t ,

$$X_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{iD}^t) \quad (17)$$

In the search space Ω , NP item individuals are randomly selected to form the initial population $X(0)$,

$$X(t) = \{X_1^t, X_2^t, \dots, X_{NP}^t\}, t = 0, 1, \dots, T \quad (18)$$

where t is the population evolutionary algebra; X_i^t denotes the i -th item individual in the population $X(t)$; x_{ij} indicates the j -th gene of the item individual X_i^t .

To avoid item individuals from being repeatedly selected, an item individual is chosen and will be excluded from the search space Ω .

$$\Omega = \bigcup_{i=1}^N X_i, \Omega^t = \Omega / \{X_i^t\} \quad (19)$$

Then, three different evolutionary individuals X_{p1}^t , X_{p2}^t , and X_{p3}^t ($p1 \neq p2 \neq p3$) are randomly selected in the current population $X(t)$. A mutated individual H_i^{t+1} is generated by differential vectors and mutation factors.

$$H_{ij}^{t+1} = X_{p1}^t + F(X_{p2}^t - X_{p3}^t) \quad (20)$$

where $X_{p2}^t - X_{p3}^t$ denotes the differential vector; the mutation factor $F \in [0, 2]$ is used to control the scale of the differential vector; $p1$, $p2$ and $p3$ are different positive integers.

Furtherly, to a trial individual $V_i^{t+1} = (v_{i1}^{t+1}, v_{i2}^{t+1}, \dots, v_{iD}^{t+1})$ is generated by performing the crossover operation of the evolution individual $X_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{iD}^t)$ and its mutated individual $H_i^{t+1} = (h_{i1}^{t+1}, h_{i2}^{t+1}, \dots, h_{iD}^{t+1})$.

$$v_{ij}^{t+1} = \begin{cases} h_{ij}^{t+1}, \text{rand}(ij) \leq CR \text{ or } j = \delta \\ x_{ij}^t, \text{rand}(ij) > CR \text{ and } j \neq \delta \end{cases} \quad (21)$$

where $\text{rand}(ij) \in [0, 1]$ is a random number corresponding to the j -th gene of evolution individual X_i^t ; the crossover factor $CR \in [0, 1]$ controls the evolutionary degree of trial individuals; $\delta \in \{1, 2, \dots, D\}$ is a parameter controlling the mutation position.

According to the user preference surrogate model based on the user interest preferences, the predicted ratings of the user for the trial individual V_i^{t+1} and target individual X_i^t are estimated as the fitness of the evolution individuals. The item individual with higher fitness are selected:

$$U_i^{t+1} = \begin{cases} V_i^{t+1}, f(X_i^t) < f(V_i^{t+1}) \\ X_i^t, f(X_i^t) \geq f(V_i^{t+1}) \end{cases} \quad (22)$$

By a similarity measurement criterion [24, 25], some feasible item individuals X_i^{t+1} similar to new evolutionary individual U_i^{t+1} are selected from the search space Ω to update the next generation population $X(t+1)$.

$$X_i^{t+1} = \begin{cases} \max\left(\frac{\langle V_i^{t+1}, \Omega_{data} \rangle}{\|V_i^{t+1}\| \cdot \|\Omega_{data}\|}\right), U_i^{t+1} = V_i^{t+1} \\ X_i^t, \text{otherwise} \end{cases} \quad (23)$$

$$\Omega^{t+1} = \Omega^t / \{X_i^{t+1}\} \quad (24)$$

Using a trained user preference surrogate model, the interactive Differential Evolution (DE) algorithm is driven to perform mutation, crossover, and selection operations until the termination condition of the algorithm is met (either a satisfactory solution for the user is found or the maximum number of evolutionary iterations is reached).

3.3 Dynamic Management Mechanism

Training and Updating the User Preference Surrogate Model

During the evolutionary iteration of the algorithm, transfer learning technology [26] is employed to enhance the computational efficiency of the user preference surrogate model. A feature extraction model with fixed parameters is trained based on historical user-generated content and applied in the feature extraction step. To dynamically fit the changing interests of users, the feature interaction model is kept dynamically updated to capture the latest interaction relationships between users and items.

Adaptive Training Sample Collection

An adaptive training sample collection strategy is designed to transfer beneficial knowledge from past time periods to the current one while updating the user preference surrogate model with new data.

$$Q^t = \mu Q_{fitness \geq \gamma}^t + \Delta Q^t \quad (25)$$

where Q^t represents the training samples for the current period; ΔQ^t is the new samples; $Q_{fitness \geq \gamma}^t$ represents the historical samples with the fitness greater than threshold value γ ; The adaptive parameter μ is used to regulate the number of $Q_{fitness \geq \gamma}^t$ based on the quantity of ΔQ^t in each period, maintaining a fixed ratio between them.

4 Experimental Design and Analysis

In this section, we evaluate the feasibility and effectiveness of the proposed SADE-DCAFM algorithm by verifying the following four questions:

Question 1: How does the proposed DCAFM model perform compared to other recommended models?

Question 2: Does the combination of user surrogate models and differential evolution algorithms enhance personalized search and recommendation effectiveness?

Question 3: How does the comprehensive performance of the proposed SADE-DCAFM compare with other surrogate-assisted evolutionary optimization algorithms?

Question 4: What impact do different hyperparameters have on the search results and recommendation effectiveness of the proposed SADE-DCAFM?

We conduct experiments on three datasets of real-world public from different domains to verify the comprehensive performance of the proposed algorithm. The statistics of dataset are summarized in Table 1.

Table 1. Table captions should be placed above the tables.

Dataset	#Users	#Items	#Scores
Amazon-Store	1,406,890	430,530	3,205,467
Amazon-Music	478,235	266,414	838,006
Yelp	1,912,494	180,347	7,778,794

We employ Root Mean Square Error (RMSE), Normalized Discounted Cumulative Gain (NDCG), Hits Ratio (HR), and Mean Average Precision (MAP) for evaluation, which are widely used in related researches, to assess the algorithms’ performance in rating prediction accuracy, sorting quality, recommendation hit rate, and relevance.

4.1 Experiment 1 (Answer Question 1)

To assess the performance of the proposed user preference surrogate model, the DCAFM algorithm is compared with several baseline algorithms, including BPR [27], DeepCoNN [28], FMLP [29], JOVA [30], and SSCF [31]. The experimental results of various personalized recommendation algorithms are shown in Table 2 and the best solution is bolded.

From the experimental results, we can observe the following conclusions:

- (1) In general, the proposed DCAFM model demonstrates superior experimental results across most evaluation metrics compared to other baseline models. For instance, in the Amazon-Store dataset, the DCAFM algorithm achieved the best RMSE value of 0.7223, which is lower than the next best SSCF model’s RMSE of 0.8930. Additionally, it obtained the highest HR@20 value of 0.0603, surpassing the next best JOVA algorithm by 4.47%. Furthermore, its NDCG@20 value was the highest at 0.9173, outperforming the next best BPR algorithm by 1.09%.
- (2) Additionally, the experimental results indicate that the DCAFM exhibits more consistent performance and greater stability across different datasets compared to other baseline models. This enhanced stability may be attributed to the design advantages of the DCAFM algorithm itself. The DCAFM makes full use of multi-source heterogeneous user-generated content and incorporates effective feature extraction and feature interaction models, enabling it to more accurately predict user ratings for items, thereby improving recommendation quality and stability.

Table 2. Performance Comparison of Different Algorithm on three Datasets

Dataset		BPR	DeepCoNN	FMLP	JOVA	SSCF	DCAFM
Amazon-Store	RMSE	2.6364	1.7911	1.8572	1.977	0.8930	0.7223
	HR@20	0.0405	0.0427	0.0510	0.0576	0.0261	0.0603
	mAP@20	0.9631	0.4366	0.5607	0.7518	0.1350	0.9162
	NDCG@20	0.9073	0.8792	0.8621	0.8801	0.5531	0.9173
Amazon-Music	RMSE	2.5609	1.1036	1.3320	2.012	0.9430	0.9667
	HR@20	0.0962	0.0636	0.1600	0.0275	0.0371	0.1642
	mAP@20	0.8280	0.5414	0.4600	0.6419	0.2339	0.8371
	NDCG@20	0.8565	0.5419	0.5510	0.8987	0.4142	0.8801
Yelp	RMSE	2.3065	0.8871	1.224	1.8776	1.0202	1.2857
	HR@20	0.0397	0.0200	0.0341	0.0293	0.0321	0.0432
	mAP@20	0.5932	0.3908	0.2100	0.4197	0.2571	0.6265
	NDCG@20	0.8401	0.8533	0.8677	0.8357	0.5417	0.8588

4.2 Experiment 2 (Answer Question 2)

To validate the effectiveness of the proposed SADE-DCAFM algorithm in personalized search and recommendation tasks, comparative experiments were conducted with the DCAFM model without the integration of the differential evolution algorithm. Both the DCAFM and SADE-DCAFM algorithms were independently run 10 times, and the mean values of HR@20, MAP@20, and NDCG@20 were compared to measure the performance of the algorithms. The experimental results are shown in Fig. 2.

From the results of two models across different datasets, it is evident that the evaluation metrics for SADE-DCAFM are higher than those for DCAFM. The performance improvement brought by the combined design shows a similar trend across different datasets. This indicates that the integration of the differential evolution algorithm is beneficial for accomplishing personalized search and recommendation tasks. Compared to DCAFM, SADE-DCAFM can not only update model parameters based on newly added user interaction data and timely perceive dynamic changes in user preferences but also continuously optimize the population set to enhance the recommendation set. Consequently, this significantly improves the overall performance.

4.3 Experiment 3 (Answer Question 3)

To verify the overall performance of the proposed SADE-DCAFM algorithm in handling personalized search and recommendation tasks in dynamic environments, we conducted comparative experiments with other surrogate-assisted evolutionary algorithms (SADE-MLP, RIEDA-MsH [32], AtRIEDA-MsH [33], and SAGA-DCAFM) across various datasets. The experimental results are shown in Table 3.

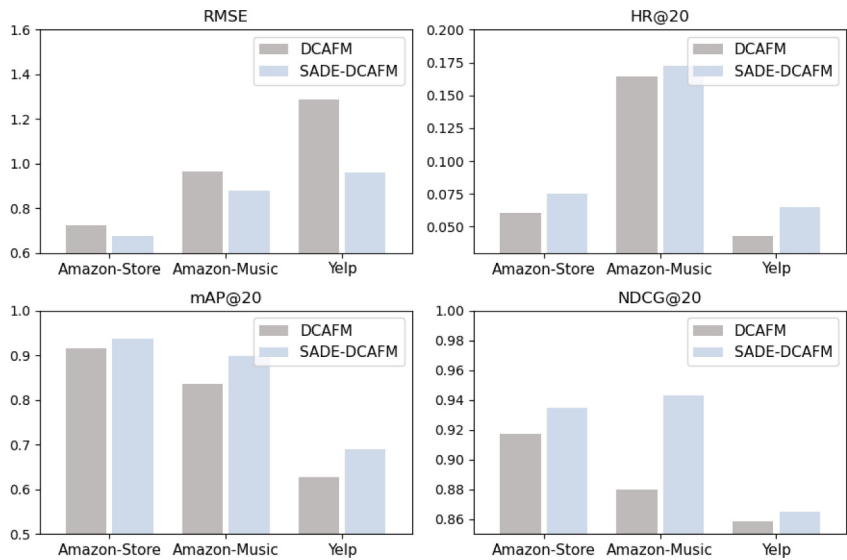


Fig. 2. Comparison Results of DCAFMs and SADE-DCAFMs

Table 3. Performance of Different Interactive Evolutionary Optimization on Three Datasets

Dataset		MLP -IDEC	RIEDA -MsH	AtRIEDA -MsH	DCAFMs -IGA	DCAFMs -IDEC
Amazon-Store	RMSE	1.1212	1.1713	1.1786	2.4176	0.6745
	HR@20	0.0285	0.0361	0.0363	0.0416	0.0753
	mAP@20	0.9193	0.9313	0.9355	0.9202	0.9372
	NDCG@20	0.9155	0.9255	0.9321	0.9226	0.9345
Amazon-Music	RMSE	2.5362	1.4275	1.4301	2.2168	0.8802
	HR@20	0.0743	0.0911	0.0956	0.1098	0.1725
	mAP@20	0.6945	0.7532	0.7919	0.7777	0.8979
	NDCG@20	0.8344	0.8592	0.8760	0.9269	0.9431
Yelp	RMSE	1.6992	1.3422	1.3434	1.6083	0.9610
	HR@20	0.0285	0.0289	0.0290	0.0383	0.0651
	mAP@20	0.6481	0.6855	0.6858	0.6761	0.6890
	NDCG@20	0.8288	0.8305	0.8321	0.8521	0.8647

From results, we can observe the following conclusions:

- (1) By analyzing the experimental results of the proposed SADE-DCAFM algorithm with those of the SADE-MLP algorithm, it can be observed that an excellent user preference surrogate model is crucial for driving the differential evolution optimization algorithm to quickly find user-satisfactory solutions. Compared to the SADE-MLP algorithm, the experimental results of the SADE-DCAFM algorithm show significant performance improvements. This is attributed to the proposed SADE-DCAFM algorithm's construction of an accurate user preference surrogate model, which more efficiently guides the optimization iterations of the evolutionary algorithm.
- (2) The contrast of the experimental results between the proposed SADE-DCAFM algorithm and the SAGA-DCAFM algorithm shows that, in the context of personalized search and recommendation tasks, the differential evolution algorithm offers superior optimization capabilities over genetic evolutionary algorithms. During the personalized search process, the differential evolution algorithm can fully utilize its search and selection capabilities to optimize the item set. This allows the user preference surrogate model to make more targeted recommendations, helping users quickly find satisfactory items.

Overall, the experimental results of the proposed SADE-DCAFM algorithm consistently outperform other surrogate-assisted evolutionary algorithms.

4.4 Experiment 4 (Answer Question 4)

To investigate the optimization performance of the differential evolution algorithm under various parameter configurations, this study utilizes user A26J2TRDPZH7RZ from the Amazon-Store dataset as a case study. The evaluation focuses on the population distribution and average fitness values at different stages from the perspective of this user. The scatter plots presented in Fig. 3 depict the projected positions of individuals in the population within a two-dimensional PCA space, providing an intuitive representation of the population distribution.

The differential evolution algorithm effectively enhances population fitness across various parameter configurations, though the degree of optimization varies depending on the specific settings. A comprehensive comparison of the initial and final population distributions, along with the improvement in average fitness under different configurations, reveals that the parameter configuration $F = 0.6$, $CR = 0.7$ is particularly effective. This configuration not only preserves population diversity but also increases the average fitness value by 0.365, significantly surpassing other parameter settings.

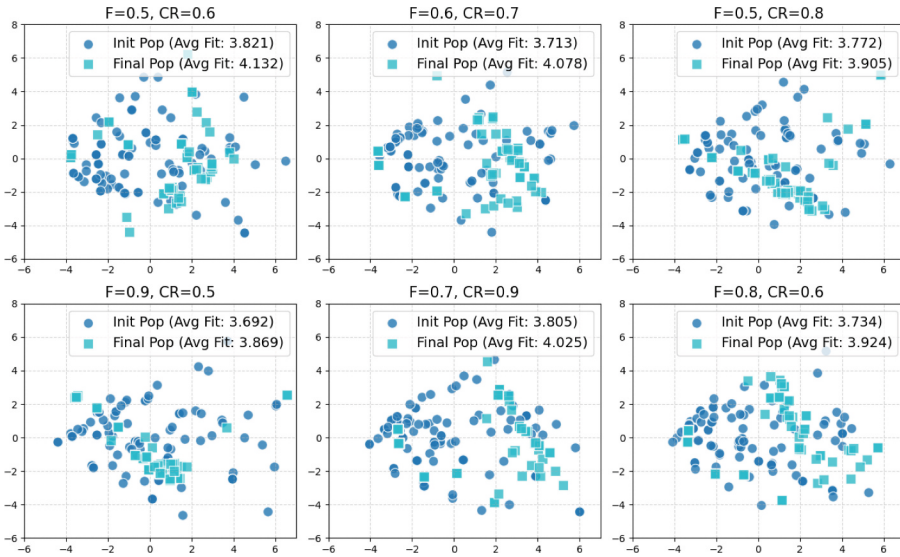


Fig. 3. The Optimization Effect with Different Parameter Configurations

5 Conclusion

This paper addresses the issues of dynamic personalized search and recommendation in a big data environment. It constructs a dynamically updatable user preference surrogate model that extracts user preference information embedded in multi-source heterogeneous user-generated data. Simultaneously, corresponding evolutionary optimization strategies are designed in the framework of differential evolution to generate new individuals that incorporate these user preferences. The user preference surrogate model is then employed to predict the fitness of item individuals, assisting users in gradually identifying satisfactory solutions. This approach will effectively address personalized search and recommendation tasks. However, this study has yet to fully explore the potential of rich multi-source heterogeneous data, such as images, videos, and location information. Future research will thoroughly investigate the impact of these data on personalized search and recommendation methods, and provide recommendations for further enhancing algorithm performance.

Acknowledgement. This work was jointly supported by the National Natural Science Foundation of China under grants No. 61876184 and the Postgraduate Research & Practice Innovation Program of Jiangsu Province under grants No. SJCX24_2494.

References

1. Truong, Q.-T., Lauw, H.: Multimodal review generation for recommender systems. In: The World Wide Web Conference, pp. 1864–1874. ACM, New York (2019). <https://doi.org/10.1145/3308558.3313463>

2. Wang, X., Kadioğlu, S.: Modeling uncertainty to improve personalized recommendations via Bayesian deep learning. *International J. Data Science and Analytics* **16**(2), 191–201 (2023)
3. Liang, J., et al.: MMMLP: multi-modal multilayer perceptron for sequential recommendations. In: *Proceedings of the ACM Web Conference*, pp. 1109–1117 (2023)
4. Jia, N., Chen, J., Wang, R.: An attention-based convolutional neural network for recipe recommendation. *Expert Syst. Appl.* **201**, 116979 (2022)
5. Pan, H., Yang, X.: Intelligent recommendation method integrating knowledge graph and Bayesian network. *Soft. Comput.* **27**(1), 483–492 (2023)
6. Tyagi, G., Ray, S.: A comparative review of convolutional neural networks, long short-term memory, and recurrent neural networks in recommendation systems. In: *International Conference on Artificial Intelligence on Textile and Apparel*, pp. 395–408 (2023)
7. Patel, R., Thakkar, P., Ukani, V.: CNNRec: convolutional Neural Network based recommender systems—a survey. *Eng. Appl. Artif. Intell.* **133**, 108062 (2024). <https://doi.org/10.1016/j.engappai.2024.108062>
8. Khan, Z., Iltaf, N., Afzal, H., Abbas, H.: Enriching non-negative matrix factorization with contextual embeddings for recommender systems. *Neurocomputing* **380**, 246–258 (2020). <https://doi.org/10.1016/j.neucom.2019.09.080>
9. Dezfouli, P.A.B.B., Momtazi, S., Dehghan, M.: Deep neural review text interaction for recommendation systems. *Appl. Soft Comput.* **100**, 106985 (2021)
10. Li, J., et al.: Text is all you need: learning language representations for sequential recommendation. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1258–1267. ACM, New York (2023), <https://doi.org/10.1145/3580305.3599519>
11. Song, Y., et al.: Show me the whole world: towards entire item space exploration for interactive personalized recommendations. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 947–956 (2022)
12. Cheng, D., et al.: Ihgnn: Interactive hypergraph neural network for personalized product search. In: *Proceedings of the ACM Web Conference 2022*, pp. 256–265 (2022)
13. Xie, L., Li, G., Wang, Z., Cui, L., Gong, M.: Surrogate-assisted evolutionary algorithm with model and infill criterion auto-configuration. *IEEE Transactions on Evolutionary Computation* (2023)
14. Liu, J., Wang, Y., Sun, G., Pang, T.: Multisurrogate-assisted ant colony optimization for expensive optimization problems with continuous and categorical variables. *IEEE Trans. Cybernetics* **52**(11), 11348–11361 (2021)
15. Liu, S., Wang, H., Peng, W., Yao, W.: A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification. *IEEE Trans. Evol. Comput.* **26**(5), 1087–1101 (2022)
16. Starodubcev, N.O., Nikitin, N.O., Kalyuzhnaya, A.V.: Surrogate-assisted evolutionary generative design of breakwaters using deep convolutional networks. In: *2022 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8 (2022)
17. Jin, Y., Wang, H., Chugh, T., Guo, D., Miettinen, K.: Data-driven evolutionary optimization: an overview and case studies. *IEEE Trans. Evol. Comput.* **23**(3), 442–458 (2018)
18. Nyikosa, F.M., Osborne, M.A., Roberts, S.J.: Bayesian Optimization for Dynamic Problems. *arXiv preprint arXiv:1803.03432* (2018)
19. Richter, J., Shi, J., Chen, J.-J., Rahnenführer, J., Lang, M.: Model-based optimization with concept drifts. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pp. 877–885 (2020)
20. Li, K., Chen, R., Yao, X.: A data-driven evolutionary transfer optimization for expensive problems in dynamic environments. *IEEE Transactions on Evolutionary Computation* (2023)

21. Li, Y.-L., Zhan, Z.-H., Gong, Y.-J., Chen, W.-N., Zhang, J., Li, Y.: Differential evolution with an evolution path: a DEEP evolutionary algorithm. *IEEE Transactions on Cybernetics* **45**(9), 1798–1810 (2014)
22. Ahmad, M.F., Isa, N.A.M., Lim, W.H., Ang, K.M.: Differential evolution: a recent review based on state-of-the-art works. *Alex. Eng. J.* **61**(5), 3831–3872 (2022)
23. Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A., et al.: Differential evolution: a review of more than two decades of research. *Eng. Appl. Artif. Intell.* **90**, 103479 (2020)
24. Li, B., Han, L.: Distance weighted cosine similarity measure for text classification. In: *Intelligent Data Engineering and Automated Learning—IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20–23, 2013. Proceedings*, pp. 611–618. Springer (2013)
25. Purwadi, J., Delima, R., Wibowo, A., Rumuy, A.: Comparison of the application of weighted cosine similarity and Minkowski distance similarity methods in stroke diagnostic system. *International Journal of Advanced Computer Science & Applications* **14**(12) (2023)
26. Hosna, A., Merry, E., Gyalmo, J., Alom, Z., Aung, Z., Azim, M.A.: Transfer learning: a friendly introduction. *Journal of Big Data* **9**(1), 102 (2022)
27. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian Personalized Ranking from Implicit Feedback. *arXiv preprint [arXiv:1205.2618](https://arxiv.org/abs/1205.2618)* (2012)
28. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 425–434 (2017)
29. Zhou, K., Yu, H., Zhao, W.X., Wen, J.-R.: Filter-enhanced MLP is all you need for sequential recommendation. In: *Proceedings of the ACM Web Conference 2022*, pp. 2388–2399 (2022), <https://doi.org/10.1145/3485447.3512111>
30. Askari, B., Szlichta, J., Salehi-Abari, A.: Joint Variational Autoencoders for Recommendation with Implicit Feedback. *arXiv preprint [arXiv:2008.07577](https://arxiv.org/abs/2008.07577)* (2020)
31. Albora, G., Mori, L.R., Zaccaria, A.: Sapling similarity: a performing and interpretable memory-based tool for recommendation. *Knowl.-Based Syst.* **275**, 110659 (2023)
32. Bao, L., Sun, X., Gong, D., Zhang, Y.: Multisource heterogeneous user-generated contents-driven interactive estimation of distribution algorithms for personalized search. *IEEE Trans. Evol. Comput.* **26**(5), 844–858 (2021)
33. Bao, L., Sun, X., Gong, D., Zhang, Y.: Enhanced restricted Boltzmann machine-driven interactive estimation of distribution algorithms with attention mechanism. *Acta Automatica Sinica* **49**(10), 2188–2200 (2023)

Author Index

B

Bao, Lin [II-201](#)
Bi, Gang [I-189](#)

C

Cao, Jiuwen [I-201](#)
Chang, Yuhao [I-109](#), [I-119](#)
Chen, Geng [II-93](#), [II-106](#)
Chen, Jiagui [I-201](#)
Chen, Long [I-252](#)
Chen, Zaixin [II-77](#), [II-132](#)
Cheng, Jinwei [I-61](#)
Chu, Jie [I-49](#)
Cui, Mingtao [I-109](#)

D

Ding, Jundi [I-158](#)
Ding, Nan [I-169](#)
Dong, Yifan [I-39](#)
Du, Peike [II-77](#)

F

Fan, Bai [I-212](#)
Fan, Jiaxin [I-189](#)
Fang, Jiajia [I-201](#)
Fang, Jinguo [I-119](#)
Feng, Shuyi [II-3](#)
Feng, Xiao [I-99](#), [I-109](#), [I-119](#)
Fu, Jiajia [I-3](#), [I-132](#)

G

Gan, Lan [II-18](#), [II-147](#)
Gao, Haotian [I-158](#)
Gao, Yuxing [I-252](#)

H

Han, Dezhi [I-179](#)
Han, Hao [II-181](#)
Hao, Jun [II-201](#)
Hao, Zhifei [I-99](#)

He, Dacheng [I-179](#), [II-121](#)
He, Mengmeng [I-265](#)
He, Wenchen [I-39](#), [I-89](#)
Hu, Dinghan [I-201](#)
Huang, Donghai [I-3](#), [I-132](#)
Huang, Jain [II-31](#)
Huang, Lvchao [I-119](#)
Huang, Tianshu [I-169](#)
Huo, Yingzi [II-43](#)

J

Jiang, Dongqi [I-189](#)
Jiang, Man [II-121](#)
Jiao, Ankang [II-181](#)

K

Kaihu, Hou [I-212](#)
Kang, Zhongmiao [I-3](#), [I-132](#)
Ke, Meiguo [II-43](#)

L

Li, Biqing [II-132](#)
Li, Haotian [I-142](#)
Li, Kefei [II-43](#)
Li, Wendi [II-181](#)
Li, Xingnan [I-3](#), [I-132](#)
Li, Yanbo [II-31](#)
Li, Yitao [I-49](#), [I-89](#)
Li, Zi [II-66](#)
Liang, Wenjuan [I-132](#)
Liu, Guangwei [I-99](#), [I-119](#)
Liu, Peng [I-228](#), [I-239](#)
Liu, Wanyu [I-61](#)
Liu, Wenbin [I-109](#)
Liu, Xingtong [I-12](#)
Liu, Yuelin [II-66](#)
Liu, Yuxin [II-43](#)
Lu, Chenxin [I-61](#), [I-76](#), [II-3](#)
Lu, Jiangang [I-3](#)
Luo, Wen [I-142](#), [II-54](#)

Lv, Dong I-39, I-89
Lv, Enqi I-252

M

Mao, Yang II-132
Meng, Xiangwei I-179

N

Ning, Shan I-49
Niu, Kunxiang I-142, II-54

P

Peiyao, Gong I-212
Peng, Yumeng II-201
Pengbiao, Chen I-212

Q

Qin, Jiawei I-89
Qiu, Yuxiang II-18, II-147

S

Shen, Ji I-76
Su, Bo I-189
Sui, Mingshuang I-39
Sun, Denghui I-239
Sun, Xiaoyan II-201
Sun, Yong I-109

W

Wan, Ming II-147
Wang, Fei II-161
Wang, Haiyang I-39, I-49
Wang, Kaiqiao I-239
Wang, Liqiang I-39, I-49, I-89
Wang, Weiming II-106
Wang, Yong I-23
Wang, Yu II-201
Wang, Zihan I-89
Wu, Kaijun II-77

Wu, Yanbing II-43
Wu, Zhenhong I-189

X

Xiao, Lijun I-179
Xu, Ke II-121
Xue, Wenhua I-252

Y

Yan, Hanming I-99
Yan, Kai I-189
Yan, Tihua II-77, II-132
Yang, Hang II-201
Yao, Longfei I-142
Yu, Ji I-158
Yuan, Jie I-76, II-3

Z

Zeng, Qingtian II-93, II-106
Zhang, Chun I-239
Zhang, Hang I-99
Zhang, Hongbin II-66
Zhang, Lei II-18
Zhang, Peiyao I-109
Zhang, Ronglin II-43
Zhang, Xiaoyan I-265
Zhang, Yiming II-93
Zhang, Zhen II-3
Zhang, Zhezhaio I-201
Zhao, Dongmei II-66
Zhao, Jingjing I-12
Zhao, Liutao I-169
Zheng, WenFa I-23
Zheng, Yang I-61, I-76, II-3
Zheng, Zhongxiang I-228
Zhou, Qirong I-49
Zhou, Shuhui I-61, I-76
Zhu, Honglong II-181
Zhu, Yuan II-18, II-147