

# Federated Learning for Internet of Things and Big Data

Lead Guest Editor: Jinbo Xiong

Guest Editors: Lei Chen, Narasimha Shashidhar, and Zuobin Ying





---

# **Federated Learning for Internet of Things and Big Data**

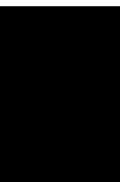
Wireless Communications and Mobile Computing

---

## **Federated Learning for Internet of Things and Big Data**

Lead Guest Editor: Jinbo Xiong

Guest Editors: Lei Chen, Narasimha Shashidhar,  
and Zuobin Ying



---



Copyright © 2022 Hindawi Limited. All rights reserved.

This is a special issue published in “Wireless Communications and Mobile Computing.” All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

# Chief Editor
































Zhipeng Cai , USA

## Associate Editors

Ke Guan , China  
Jaime Lloret , Spain  
Maode Ma , Singapore

## Academic Editors

Muhammad Inam Abbasi, Malaysia  
Ghufran Ahmed , Pakistan  
Hamza Mohammed Ridha Al-Khafaji ,  
Iraq  
Abdullah Alamoodi , Malaysia  
Marica Amadeo, Italy  
Sandhya Aneja, USA  
Mohd Dilshad Ansari, India  
Eva Antonino-Daviu , Spain  
Mehmet Emin Aydin, United Kingdom  
Parameshchhari B. D. , India  
Kalapraveen Bagadi , India  
Ashish Bagwari , India  
Dr. Abdul Basit , Pakistan  
Alessandro Bazzi , Italy  
Zdenek Becvar , Czech Republic  
Nabil Benamar , Morocco  
Olivier Berder, France  
Petros S. Bithas, Greece  
Dario Bruneo , Italy  
Jun Cai, Canada  
Xuesong Cai, Denmark  
Gerardo Canfora , Italy  
Rolando Carrasco, United Kingdom  
Vicente Casares-Giner , Spain  
Brijesh Chaurasia, India  
Lin Chen , France  
Xianfu Chen , Finland  
Hui Cheng , United Kingdom  
Hsin-Hung Cho, Taiwan  
Ernestina Cianca , Italy  
Marta Cimitile , Italy  
Riccardo Colella , Italy  
Mario Collotta , Italy  
Massimo Condoluci , Sweden  
Antonino Crivello , Italy  
Antonio De Domenico , France  
Floriano De Rango , Italy

Antonio De la Oliva , Spain  
Margot Deruyck, Belgium  
Liang Dong , USA  
Praveen Kumar Donta, Austria  
Zhuojun Duan, USA  
Mohammed El-Hajjar , United Kingdom  
Oscar Esparza , Spain  
Maria Fazio , Italy  
Mauro Femminella , Italy  
Manuel Fernandez-Veiga , Spain  
Gianluigi Ferrari , Italy  
Luca Foschini , Italy  
Alexandros G. Fragkiadakis , Greece  
Ivan Ganchev , Bulgaria  
Óscar García, Spain  
Manuel García Sánchez , Spain  
L. J. García Villalba , Spain  
Miguel Garcia-Pineda , Spain  
Piedad Garrido , Spain  
Michele Girolami, Italy  
Mariusz Glabowski , Poland  
Carles Gomez , Spain  
Antonio Guerrieri , Italy  
Barbara Guidi , Italy  
Rami Hamdi, Qatar  
Tao Han, USA  
Sherief Hashima , Egypt  
Mahmoud Hassaballah , Egypt  
Yejun He , China  
Yixin He, China  
Andrej Hrovat , Slovenia  
Chunqiang Hu , China  
Xuexian Hu , China  
Zhenghua Huang , China  
Xiaohong Jiang , Japan  
Vicente Julian , Spain  
Rajesh Kaluri , India  
Dimitrios Katsaros, Greece  
Muhammad Asghar Khan, Pakistan  
Rahim Khan , Pakistan  
Ahmed Khattab, Egypt  
Hasan Ali Khattak, Pakistan  
Mario Kolberg , United Kingdom  
Meet Kumari, India  
Wen-Cheng Lai , Taiwan

Jose M. Lanza-Gutierrez, Spain  
Pavlos I. Lazaridis , United Kingdom  
Kim-Hung Le , Vietnam  
Tuan Anh Le , United Kingdom  
Xianfu Lei, China  
Jianfeng Li , China  
Xiangxue Li , China  
Yaguang Lin , China  
Zhi Lin , China  
Liu Liu , China  
Mingqian Liu , China  
Zhi Liu, Japan  
Miguel López-Benítez , United Kingdom  
Chuanwen Luo , China  
Lu Lv, China  
Basem M. ElHalawany , Egypt  
Imadeldin Mahgoub , USA  
Rajesh Manoharan , India  
Davide Mattera , Italy  
Michael McGuire , Canada  
Weizhi Meng , Denmark  
Klaus Moessner , United Kingdom  
Simone Morosi , Italy  
Amrit Mukherjee, Czech Republic  
Shahid Mumtaz , Portugal  
Giovanni Nardini , Italy  
Tuan M. Nguyen , Vietnam  
Petros Nicolitidis , Greece  
Rajendran Parthiban , Malaysia  
Giovanni Pau , Italy  
Matteo Petracca , Italy  
Marco Picone , Italy  
Daniele Pinchera , Italy  
Giuseppe Piro , Italy  
Javier Prieto , Spain  
Umair Rafique, Finland  
Maheswar Rajagopal , India  
Sujan Rajbhandari , United Kingdom  
Rajib Rana, Australia  
Luca Reggiani , Italy  
Daniel G. Reina , Spain  
Bo Rong , Canada  
Mangal Sain , Republic of Korea  
Praneet Saurabh , India

Hans Schotten, Germany  
Patrick Seeling , USA  
Muhammad Shafiq , China  
Zaffar Ahmed Shaikh , Pakistan  
Vishal Sharma , United Kingdom  
Kaize Shi , Australia  
Chakchai So-In, Thailand  
Enrique Stevens-Navarro , Mexico  
Sangeetha Subbaraj , India  
Tien-Wen Sung, Taiwan  
Suhua Tang , Japan  
Pan Tang , China  
Pierre-Martin Tardif , Canada  
Sreenath Reddy Thummaluru, India  
Tran Trung Duy , Vietnam  
Fan-Hsun Tseng, Taiwan  
S Velliangiri , India  
Quoc-Tuan Vien , United Kingdom  
Enrico M. Vitucci , Italy  
Shaohua Wan , China  
Dawei Wang, China  
Huaqun Wang , China  
Pengfei Wang , China  
Dapeng Wu , China  
Huaming Wu , China  
Ding Xu , China  
YAN YAO , China  
Jie Yang, USA  
Long Yang , China  
Qiang Ye , Canada  
Changyan Yi , China  
Ya-Ju Yu , Taiwan  
Marat V. Yuldashev , Finland  
Sherali Zeadally, USA  
Hong-Hai Zhang, USA  
Jiliang Zhang, China  
Lei Zhang, Spain  
Wence Zhang , China  
Yushu Zhang, China  
Kechen Zheng, China  
Fuhui Zhou , USA  
Meiling Zhu, United Kingdom  
Zhengyu Zhu , China






# Contents

## **Federated Learning Incentive Mechanism Design via Enhanced Shapley Value Method**

Xun Yang , Weijie Tan , Changgen Peng , Shuwen Xiang , and Kun Niu 






Research Article (11 pages), Article ID 9690657, Volume 2022 (2022)

## **Local Epochs Inefficiency Caused by Device Heterogeneity in Federated Learning**

Yan Zeng , Xin Wang , Junfeng Yuan , Jilin Zhang , and Jian Wan 

Research Article (15 pages), Article ID 6887040, Volume 2022 (2022)

## **An IoT Crossdomain Access Decision-Making Method Based on Federated Learning**

Chao Li , Fan Li , Zhiqiang Hao, Lihua Yin , Zhe Sun , and Chonghua Wang 





Research Article (9 pages), Article ID 8005769, Volume 2021 (2021)

## **Multidomain Fusion Data Privacy Security Framework**

Jing Yang, Lianwei Qu , and Yong Wang 



Research Article (26 pages), Article ID 8492223, Volume 2021 (2021)

## **VarDefense: Variance-Based Defense against Poison Attack**

Mingyuan Fan , Xue Du , Ximeng Liu , and Wenzhong Guo 




Research Article (9 pages), Article ID 1974822, Volume 2021 (2021)

## **Enhancing Packet-Level Wi-Fi Device Authentication Protocol Leveraging Channel State Information**

Yubo Song , Bing Chen, Tianqi Wu, Tianyu Zheng, Hongyuan Chen , and Junbo Wang


Research Article (12 pages), Article ID 2993019, Volume 2021 (2021)

## **Signal Modulation Recognition Method Based on Differential Privacy Federated Learning**

Jibo Shi , Lin Qi, Kuixian Li , and Yun Lin 




Research Article (13 pages), Article ID 2537546, Volume 2021 (2021)

## **LstFcFedLear: A LSTM-FC with Vertical Federated Learning Network for Fault Prediction**

Xiangquan Zhang , Zhili Ma, Anmin Wang, Haifeng Mi, and Junjun Hang




Research Article (10 pages), Article ID 2668761, Volume 2021 (2021)

## **A Novel Way to Generate Adversarial Network Traffic Samples against Network Traffic Classification**

Yongjin Hu , Jin Tian , and Jun Ma 




Research Article (12 pages), Article ID 7367107, Volume 2021 (2021)

## **Sentiment Classification Algorithm Based on the Cascade of BERT Model and Adaptive Sentiment Dictionary**

Ruixue Duan , Zhuofan Huang , Yangsen Zhang , Xiulei Liu , and Yue Dang 




Research Article (8 pages), Article ID 8785413, Volume 2021 (2021)

## **UserRBPM: User Retweet Behavior Prediction with Graph Representation Learning**

Huihui Guo , Li Yang , and Zeyu Liu 

Research Article (17 pages), Article ID 4431416, Volume 2021 (2021)

**Privacy-Preserving Federated Learning Framework with General Aggregation and Multiparty Entity Matching**

Zhou Zhou , Youliang Tian , and Changgen Peng 

Research Article (14 pages), Article ID 6692061, Volume 2021 (2021)



## Research Article

# Federated Learning Incentive Mechanism Design via Enhanced Shapley Value Method

Xun Yang <sup>1</sup>, Weijie Tan <sup>2,3,4</sup>, Changgen Peng <sup>2,3</sup>, Shuwen Xiang <sup>1</sup> and Kun Niu <sup>2</sup>

<sup>1</sup>College of Mathematics and Statistics, Guizhou University, Guiyang 550025, China

<sup>2</sup>State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

<sup>3</sup>Guizhou Big Data Academy, Guizhou University, Guiyang 550025, China

<sup>4</sup>Key Laboratory of Advanced Manufacturing Technology, Ministry of Education, Guizhou University, Guiyang 550025, China

Correspondence should be addressed to Weijie Tan; [wjtan@gzu.edu.cn](mailto:wjtan@gzu.edu.cn) and Changgen Peng; [cgpeng@gzu.edu.cn](mailto:cgpeng@gzu.edu.cn)

Received 10 July 2021; Revised 20 October 2021; Accepted 9 December 2021; Published 28 January 2022

Academic Editor: James Ying

Copyright © 2022 Xun Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Federated learning (FL) is an emerging collaborative machine learning method. In FL processing, the data quality shared by users directly affects the accuracy of the federated learning model, and how to encourage more data owners to share data is crucial. In other words, how to design a good incentive mechanism is the key problem in FL. In this paper, we propose an incentive mechanism based on the enhanced Shapley value method for FL. In the proposed mechanism, the enhanced Shapley value method is proposed to measure income distribution, which takes multiple influence factors as weights. The analytic hierarchy process (AHP) is used to find the corresponding weight value of the influence factors. Finally, the numerical experiments are carried to verify the performance of the proposed incentive mechanism. The results show that compared with the Shapley value method considering the single factor, the income distribution of all participants can better reflect multiple factor contribution when using the enhanced Shapley value method.

## 1. Introduction

With the rapid development of the artificial intelligence technology, it is quite common for cross-organizational and cross-institutional data use. How to ensure data privacy and security while realizing data sharing [1, 2] have attracted extensive attention, which is a fundamental challenge in the era of artificial intelligence [3–6]. Federated learning (FL) is a new distributed machine learning method to address the data island problem, and it was first proposed by Google in 2016 [7–9], which can be applied to the Android phone user's local model updating and training between multiple participants or multiple computing nodes. Nowadays, it has been used in all walks of life widely.

In the process of FL, the incentive mechanism is an important research topic [10–12]. To make all data owners contribute their data for model training actively, it is necessary to establish a reasonable incentive mechanism to encourage data owners to share valuable data. A valuable incentive mechanism can enable all participants to continu-

ously and carry out efficient model training and improve the final trained federated model more accuracy. When there is no a meaningful incentive mechanism, all participants do not update data in real time in the process of data federation and do not contribute to FL actively, which will not obtain the accurate training model, and all participants dare not use the trained model. It will also cause a waste of time and economic investment of all participants.

Recently, there are a series of research results about the incentive mechanism of FL. In [11], the authors proposed an incentive scheme for a local, client machine learning model for FL, which solves the problem that participants may opt out of the federated system and provides useless information to the federated system without satisfactory incentives. To facilitate collaborative machine learning among multiple model owners, the work in [12] proposed a hierarchical incentive mechanism framework, which solves the problem of hierarchical structure within federated systems. Zhan et al. [13] designed a deep reinforcement learning-based incentive mechanism, which solves unique

the nonsharing of data and difficult contribution evaluation. Yu et al. [14] proposed a divide budget incentive method, which solves the restriction of temporary mismatch between contribution and reward in FL. From these researches, we note that participant's submodel influences the quality of trained model generated from FL. Generally, the participants have an enthusiasm to cooperate with each other for improving the quality of model. In [15, 16], all participants in FL are considered as having equivalent contributions. However, the contributions of participants have distinct in some cases. For example, the FL model is applied in WeBank, which has cooperated with such financial institutions, insurance companies, and enterprises.

In these existing works, they do not take multiple distribution factors such as cost input, risk factor, and data quality that affect the distribution of benefits. To obtain income distribution fair and just, we proposed an incentive mechanism based on the enhanced Shapley value method for FL. In the proposed method, all federated participants get different federated incomes according the contribution in model training, and there is no free-riding among all participants.

The major contributions of this paper are summarized as the threefold:

- (1) We construct the federated incentive model for multiple participants, introduce the cooperative game to the incentive model in FL, and consider multiple factors that affect the income distribution of participants in the FL system
- (2) To allocate the income for the federated participants more fair, we propose the analytic hierarchy process (AHP) to construct the expert judgment matrix according the multiple contribution factors, which achieves the corresponding weights of all the influencing factors
- (3) The effectiveness of this incentive mechanism is verified through numerical experiments. Experimental results show that the income distribution of all participants can better reflect multiple factor contribution

The rest of this paper is organized as follows: The related work and our main contribution of this paper are introduced in Section 2. The definition of FL, cooperative games, Shapley value, and the AHP are introduced in Section 3. The fair federated model and incentive enhanced Shapley value method are established in Section 4. The rationality of the method is verified by numerical experiments and numerical comparison in Section 5. The conclusion and future work are drawn in Section 6.

## 2. Related Work

The model training accuracy of FL is affected by incentive mechanism, which has been widely concerned at present [15–17]. In [18, 19], the authors proposed an incentive mechanism which combined the contract and reputation can improve the efficiency of federated model training. In [20], an optimization method in the FL wireless networks

was formulated, which captures two trade-offs and obtains the global optimal solution by characterizing the closed-form solutions to all subproblems. In [21, 22], within the Stackelberg framework, a motivation-based interaction model was established between global servers and participating devices to encourage participants taking part in cooperation. Based on fair allocation for wireless network resources, Li et al. [23] proposed a new objective of resource optimization, which promote a fair society from the perspective of fairness, flexibility, and efficiency. For the linear regression model, the work [24] designed an incentive payment structure to encourage agents provide high-quality data, which can describe the impact of data points on the loss function of the model. The game framework was proposed in [25, 26] to solve the efficiency problem of multi-party model.

In the incentive mechanism, the return fairness is a worthy studying problem. The work of [27] studied the dynamics of coalition formation under the bounded rationality condition, which considers the situation that the profit of each team is given by the submodular function, and proposed a profit distribution scheme based on the concept of marginal utility. Xiong et al. [28] proposed an incentive mechanism for the multiattribute user selection, which effectively improves sensing user quality for mobile crowdsensing. The Shapley value method is used to distribute fair the benefits generated in the process of the coalition [29], which is the most widely used method to evaluate fairness [30, 31]. In [32], the authors combined the quality estimation with a monetary incentive and used the Shapley value method to determine the cost of each household, which was based on the effect of goodness in saving and remaining points. In [33], a repertoire of efficient algorithms was proposed to meet the time-consuming challenge in the Shapley value requiring process. Liu et al. [34] proposed a blockchain-based FedCoin peer-to-peer payment incentive mechanism for FL and used the Shapley value method to distribute federated revenue.

In the existing works, they do not take multiple related factors such as cost input, risk factor, and data quality that affect the distribution of benefits. In order to fully consider the impact of multiple factors on the fairness of distribution, an incentive mechanism for the FL system is proposed to achieve fairness and justice, which uses the enhanced Shapley value method. The proposed incentive mechanism introduces the cooperative game to the incentive model and considers multiple factors that affect the income distribution of participants. The analytic hierarchy process (AHP) is used to construct the expert judgment matrix, which achieves the corresponding weights of all the influencing factors.

## 3. Preliminaries

*3.1. Definition of Federated Learning.* Let  $\{1, 2, \dots, N\}$  be defined as  $N$  data owners, and they all hope to merge their respective data  $\{D_1, D_2, \dots, D_N\}$  to train a machine learning model. A model  $M_{\text{SUM}}$  is trained by a traditional method, which puts all data  $D = D_1 \cup D_2 \cup \dots \cup D_N$  together. A model  $M_{\text{FED}}$  can be cooperatively trained by the data owners of

claimed FL system in learning process, which the data  $D_i$  never leaked to other participant via any data owner  $i$  in this process. Moreover, the precision of  $M_{\text{FED}}$ , defined as  $V_{\text{FED}}$ , should be pretty close to the property of  $M_{\text{SUM}}$  and  $V_{\text{SUM}}$ . Generally, if there is a  $\delta \geq 0$  satisfies the condition

$$|V_{\text{FED}} - V_{\text{SUM}}| < \delta, \quad (1)$$

we say that the FL algorithm has  $\delta$ -accuracy loss [4].

**3.2. Cooperative Games.** The  $G(N, \nu)$  is defined cooperative game and satisfies two conditions as follows [29, 35, 36]:

$$\nu(S_1) + \nu(S_2) \leq \nu(S_1 \cup S_2), \quad (2)$$

$$S_1 \cap S_2 = \emptyset, \nu(\emptyset) = 0, \quad (3)$$

where  $N$  is a finite set of participants,  $S_1, S_2 \in 2^N$ ,  $\nu: 2^N \rightarrow R$  is characteristic function,  $2^N$  is the set of all the subsets of  $N$ . Let  $\nu(S)$  is participants' income function,  $\nu(N)$  indicates the coalition income and  $\varphi_i(\nu)$  (defined in formula (6)) is the income of participant  $i$  in  $\nu(N)$ , which satisfy two constraints:

$$\nu(N) = \sum_{i=1}^n \varphi_i(\nu), \quad (4)$$

$$\nu(S) \leq \sum_{i \in S} \varphi_i(\nu), \quad (5)$$

Formula (4) is named as collective rationality, which indicates that the sum of all participants' incomes from the maximum income in  $G(N, \nu)$  should be equivalent to  $\nu(N)$ , and formula (5) is claimed coalition rationality, which shows that the sum of profit allocated to all probable coalitions must be greater than or equivalent to  $\nu(S)$ . When  $|S| = 1$ , the formula (5) is individual rationality.

**3.3. Shapley Value.** Shapley value was presented by Shapley [29, 37] in the cooperative game theory, which can effectively solve the problem of cooperative income distribution and define as the following equation:

$$\varphi_i(\nu) = \sum_{i \in S, S \subset N} w(|S|) [\nu(S) - \nu(S \setminus i)], \quad (6)$$

$$w(|S|) = \frac{(n - |S|)! (|S| - 1)!}{n!}, \quad (7)$$

where  $S \subset N$ ,  $i = 1, 2, \dots, n$ ,  $|S|$  is the quantity of participants in subset  $S$ ,  $w(|S|)$  is weight coefficient,  $\nu(S)$  is the income of subset  $S$  and satisfies Equations (2) and (3), the term  $\nu(S) - \nu(S \setminus i)$  evaluates the marginal contribution of  $i$  to the coalition  $S$ , and  $\nu(S \setminus i)$  shows the benefit of other players in subset  $S$  other than  $i$ .

## 4. Federated Learning Incentive Mechanism

Before establishing the FL incentive model, we make the following assumptions:

- (1) All participants have the ability to pay for FL, and in the process of income distribution, they adopt the best income distribution scheme.
- (2) All participants are willing to participate in the coalition and do not withdraw from each sub-coalition, and all parties are very satisfied with the final income distribution scheme.
- (3) All participants are completely trustworthy, and there is no cheating.
- (4) FL should adopt multiparty agreement to recognize the income distribution scheme to ensure the smooth implementation of the strategy.

**4.1. FL Incentive Model.** The proposed FL incentive model is as shown in Figure 1, and in this model, we assume that there are  $n$  factors and  $n$  participants and establish their federated models. We use different colors to represent the process of different factors for different participants, the direction of model updating, and income distribution in the whole FL process.

*Step 1.* Under the influence of factor 1, factor 2, ..., and factor  $n$  on the federated income, we distribute model 1, model 2, ..., and model  $n$  to participant 1, participant 2, ..., and participant  $n$  and then train the  $n$  submodels respective of data participants. The data of the  $n$  participants do not leave the local, to protect the privacy of the data in their respective databases.

*Step 2.* Every time the model runs, the parameters of the  $n$  submodels are updated, until the end of the model run, and the models are aggregating to get the federated aggregation model we need.

*Step 3.* Every time the model runs, using formula (19), each participant  $i$  will receive the income  $\tilde{\varphi}_i(\nu, t)$  from the federated expected to cost  $b(t)$  in round  $t$ .

*Step 4.* When the model is running finished, the parameters of all submodels are aggregating to get the final aggregated federated model. Finally, the federated income is allocated according to the enhanced Shapley value method, and each participant  $i$  will achieve income  $\varphi_i'(\nu, t)$  from formula (20).

The income distribution of each participant  $i$  is affected by  $n$  factors, which are allocated to rely on the contribution of each participant  $i$  to the whole federation. This design makes participants get the distributed federated benefits more fairly and get an accurate federated model. Because the influencing factors impact income of participants, the enthusiasm of participants will be affected and, finally, impact the accuracy of the federated model. If each participant  $i$  actively cooperates in the federated process and tries to take part in the FL model to update data in the local database in real time, the parameters of the model training will also be updated, and the final federated aggregation model will also be updated. This cycle will continue until the end of the whole model training. Finally, the total federated

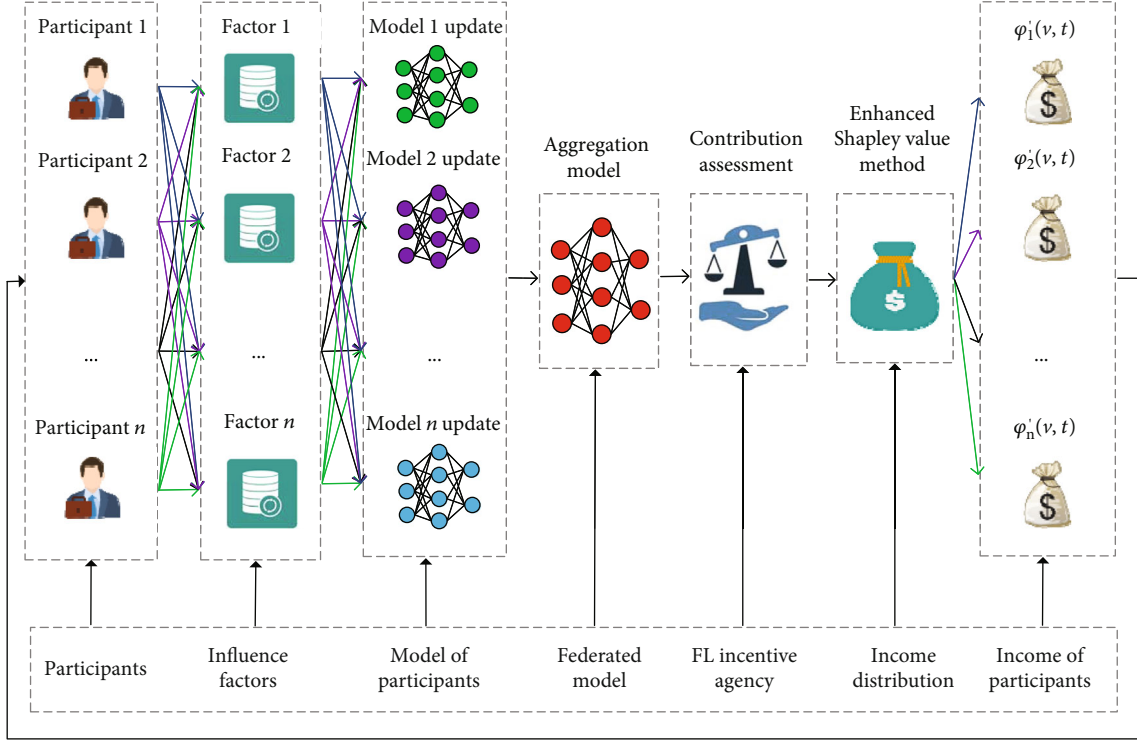


FIGURE 1: Federated learning incentive model.

income generated is distributed according to the enhanced Shapley value method.

**4.2. Contribution Assessment.** To encourage the participants to actively participate in the FL, we set each participant to be treated fairly based on their contribution and meet three fair conditions at the same time:

- (1) *Contribution Fairness.* Each participant's income is related to its contribution to the federation.
- (2) *Expectation Fairness of Risk-Taking.* The expected risk and time risk between each participant are minimized.
- (3) *Expectation Fairness.* The expected and time risks taken from each participant vary as little as possible.

We denote  $c_i(t)$  the cost of participant  $i$  and  $q_i(t)$  the contribution, and  $q_i(t) \geq 0$  includes risk-taking, data quality, effort degree, and cost input. And we shall use the procurement auction method [41] to estimate  $c_i(t)$ . Further, we define the expected risk  $f_i(t)$  and time risk  $g_i(t)$  by the following dynamic system:

$$\begin{aligned} f_i(t+1) &= \max \left[ f_i(t) + c_i(t) - \varphi_i'(v, t), 0 \right], \\ g_i(t+1) &= \max \left[ g_i(t) + \mu_i(t) - \varphi_i'(v, t), 0 \right], \end{aligned} \quad (8)$$

where  $\mu_i(t)$  represents an indicator function is

$$\mu_i(t) = \begin{cases} \tilde{c}_i(t), & f_i(t) > 0, \\ 0, & f_i(t) \leq 0. \end{cases} \quad (9)$$

Here, when  $f_i(t) > 0$ , the time series  $g_i(t)$  will increase, and  $\tilde{c}_i(t)$  shows that participant  $i$  is the average cost of contribution for the federation. So we have a federally maximized  $H$ .

$$H = \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \left[ q_i(t) \varphi_i'(v, t) \right], \quad (10)$$

satisfying fair condition (1), where  $T$  is the total quantity of runs of the model and  $q_i(t)$  represents the contribution made by participant  $i$  to the federation. We should treat all participants fairly to minimize their expected loss and distribution. Therefore, we introduce the Lyapunov function formulated as follows [42]:

$$L(t) = \frac{1}{2} \sum_{i=1}^N \left[ f_i^2(t) + g_i^2(t) \right]. \quad (11)$$

For the convenience of calculation, the root operator is omitted calculation, which will not change the good properties of formula (11). With the change of time, the expected risk deviation of the participants is as follows:

$$\begin{aligned}
\Delta L(t) &= \frac{1}{T} \sum_{t=1}^T [L(t+1) - L(t)] \\
&= \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \left[ \frac{1}{2} f_i^2(t+1) - \frac{1}{2} f_i^2(t) + \frac{1}{2} g_i^2(t+1) - \frac{1}{2} g_i^2(t) \right] \\
&\leq \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \left[ f_i(t) c_i(t) - f_i(t) \varphi_i'(v, t) + \frac{1}{2} c_i^2(t) \right. \\
&\quad - c_i(t) \varphi_i'(v, t) + \frac{1}{2} \varphi_i'^2(v, t) + g_i(t) \mu_i(t) \\
&\quad \left. - g_i(t) \varphi_i'(v, t) + \frac{1}{2} \mu_i^2(t) - \mu_i(t) \varphi_i'(v, t) + \frac{1}{2} \varphi_i'^2(v, t) \right]. \tag{12}
\end{aligned}$$

Since  $\varphi_i'(v, t)$  is a control variable of the income function of the participant  $i$ , we extract the formula containing it from formula (12) and get the following result:

$$\Delta L(t) \leq \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \left\{ \varphi_i'^2(v, t) - \varphi_i'(v, t) [f_i(t) + c_i(t) + g_i(t) + \mu_i(t)] \right\}. \tag{13}$$

The expected loss  $\Delta L(t)$  is the distribution function of  $f_i(t)$  and  $g_i(t)$ . If  $\Delta L(t)$  is minimized and fair conditions (2) and (3) are satisfied, the expected loss will gradually decrease with time. Let

$$F(t) = \varepsilon H - \Delta L(t), \tag{14}$$

is the target number, which indicates the inequality of expected loss and waiting time between participants, and it should be minimized, where  $\varepsilon$  is a federated regularization term to constraint the equilibrium between the two targets. Therefore, the federation is aimed at the following function:

$$F(t) = \frac{1}{T} \sum_{t=1}^T \sum_{i=1}^N \left\{ \varphi_i'(v, t) [\varepsilon q_i(t) + f_i(t) + c_i(t) + g_i(t) + \mu_i(t)] - \varphi_i'^2(v, t) \right\}, \tag{15}$$

and the constraint condition is

$$\sum_{i=1}^N \tilde{\varphi}_i(v, t) \leq b(t), \tilde{\varphi}_i(v, t) \geq 0, \text{ for } \forall i, t, \tag{16}$$

where  $b(t)$  means the cost of participants in the current payment budget, which is determined by their expected loss and the time they spend waiting for the reward to be fully paid, and the  $\tilde{\varphi}_i(t)$  represents the income that participant  $i$  received from the federation in round  $t$ . Let us take the derivative of formula (15) with respect to  $\varphi_i'(v, t)$ , and set the first derivative equal to 0 to obtain  $\varphi_i'(v, t)$ :

$$\varphi_i'(v, t) = \frac{1}{2} [\varepsilon q_i(t) + f_i(t) + c_i(t) + g_i(t) + \mu_i(t)]. \tag{17}$$

Moreover, by using the second derivatives of (15) with respect to  $\varphi_i'(t)$ , we obtain the following result:

$$\frac{d^2}{d\varphi_i'(v, t)} F(t) = -1 < 0. \tag{18}$$

When the federated model runs in the round  $t$ , the contribution of the participants to the federation is  $q_i(t)$ , and the total compensation of the participants  $i$  is  $\varphi_i'(v, t) = 1/2[\varepsilon q_i(t) + f_i(t) + c_i(t) + g_i(t) + \mu_i(t)]$ . If budget  $b(t)$  is not enough to fully pay all participants' compensation in round  $i$ , the federated organization will pay them in installments within a certain period. The federated organization will pay  $\tilde{\varphi}_i(v, t)$  in installments in round  $i$  according to the following equation:

$$\tilde{\varphi}_i(v, t) = \frac{\varphi_i'(v, t)}{\sum_{i=1}^N \varphi_i'(v, t)} b(t). \tag{19}$$

For the income distribution scheme in Figure 1, when  $f_i(t)$  and  $g_i(t)$  are equal to 0, this is a very important condition, and participant  $i$  does not invest extra cost; then,  $\varphi_i'(v, t) = \varepsilon q_i(t)$ . After that, participant  $i$  will use the enhanced Shapley value method to evaluate its contribution based on the federation and allocate the future income. This income allocation method will first consider the participants whose expected risk is not equal to 0 and also consider the contributions of other participants to the federation. According to Figure 1 and the above theoretical analysis, we present the pseudo-code of FL incentive in Algorithm 1.

**4.3. The Enhanced Shapley Value Method.** To ensure that all participants are satisfied with the federated income distribution scheme and make the distribution mechanism play an incentive role, all participants can actively contribute to the federated model. The Shapley value method considers that the contribution degree of each participant as  $1/n$  ignores the influence of other factors. Therefore, it is necessary to consider the effects of multiple factors on federal income distribution, by taking multiple influence factors as weights, and the enhanced Shapley value method is proposed to measure income distribution [28, 40].

$$\varphi_i'(v, t) = \varphi_i(v, t) + v(N) \times \Delta P_i, \Delta P_i = P_i - \frac{1}{n}, i = 1, 2, \dots, n, \tag{20}$$

$$P_i = (\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in})(w_1, w_2, \dots, w_n)^T, i = 1, 2, \dots, n, \tag{21}$$

where  $\varphi_i'(v, t)$  shows the anticipated income for federated model  $i$  after improvement and under a federated game condition in the  $t$  round,  $v(N)$  is the revenue of the grand coalition,  $\varphi_i(v, t)$  is the income distribution of participant  $i$  using the Shapley value method in the  $t$  round of FL. In particular, when  $t = 1$ ,  $\varphi_i'(v, t)$  is calculated using formula 6, and

Input:  $\varepsilon$  and  $b(t)$  are given by the federated system administrator.  $f_i(t)$  and  $g_i(t)$  are the expected risk of all participants and the income of all participants respectively in round  $t$ .

Output:  $\{\tilde{\varphi}_1(v, t), \tilde{\varphi}_2(v, t), \dots, \tilde{\varphi}_N(v, t)\}$

- 1: Initialization  $h(t) \leftarrow 0$ ; To stores the sum of all  $\varphi'_i(v, t)$  values.
- 2: **for**  $i = 1, 2, \dots, N$  **do**
- 3: **if**  $d_i(t) > 0$  **then**;  $d_i(t)$  represents the contribution scale value of participant  $i$  to the federated model.
- 4:     Calculation  $c_i(t)$ ;
- 5:     Calculation  $q_i(t)$ ;
- 6: **else**
- 7:      $c_i(t) = 0$ ;
- 8:      $\varphi'_i(v, t) \leftarrow \frac{1}{2} [\varepsilon q_i(t) + f_i(t) + c_i(t) + g_i(t) + \mu_i(t)]$ ;
- 9:      $h(t) \leftarrow h(t) + \varphi'_i(v, t)$ ;
- 10: **if**  $i = 1, 2, \dots, N$  **then**
- 11:      $\tilde{\varphi}_i(v, t) \leftarrow \frac{\varphi'_i(v, t)}{h(t)} b(t)$
- 12:      $f_i(t+1) = \max [f_i(t) + c_i(t) - \tilde{\varphi}_i(v, t), 0]$ ;
- 13:      $g_i(t+1) = \max [g_i(t) + \mu_i(t) - \tilde{\varphi}_i(v, t), 0]$ ;
- 14: **return**  $\{\tilde{\varphi}_1(v, t), \tilde{\varphi}_2(v, t), \dots, \tilde{\varphi}_N(v, t)\}$

ALGORITHM 1: FL incentive mechanism via enhanced Shapley value method.

$\Delta P_i$  indicates the difference between the comprehensive evaluation factor  $P_i$  and the average factor  $1/n$  introduced by participant  $i$ , and it refers to the comprehensive factor as an incentive factor in the federated distribution of incomes.

$$\Delta P_i = \begin{cases} \text{Participant } i \text{ should be encouraged,} & P_i - \frac{1}{n} > 0, \\ \text{Participant } i \text{ should be punished,} & P_i - \frac{1}{n} < 0. \end{cases} \quad (22)$$

$P_i$  is the sum of the product of the measured values  $\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in}$  of the  $n$  factors and the corresponding factor weight  $w_1, w_2, \dots, w_n$ , and  $\sum_{i=1}^n P_i = 1$ . Thus, we have

$$\begin{cases} \sum_{i=1}^n \Delta P_i = 0, \\ \sum_{i \in S} \Delta P_i \geq \frac{v(S) - \sum_{i \in S} \varphi_i(v, t)}{v(S)}. \end{cases} \quad (23)$$

We solve these weights of  $n$  influence factors  $\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in}$  with the AHP. It is easy to verify that formula (20) satisfies formulas (4) and (5).

*Proof.* According to formulas (4) and (23), formula (20) can be proofed as follows:

$$\begin{aligned} \sum_{i=1}^n \varphi'_i(v, t) &= \sum_{i=1}^n [\varphi_i(v, t) + v(N) \times \Delta P_i] \\ &= \sum_{i=1}^n \varphi_i(v, t) + v(N) \sum_{i=1}^n \Delta P_i \\ &= v(N). \end{aligned} \quad (24)$$

Hence, formula (20) satisfies formula (4). In addition, according to formulas (5) and (23), formula (20) is proofed as the following:

$$\begin{aligned} \sum_{i \in S} \varphi'_i(v, t) &= \sum_{i \in S} [\varphi_i(v, t) + v(S) \times \Delta P_i] \\ &= \sum_{i \in S} \varphi_i(v, t) + v(S) \sum_{i \in S} \Delta P_i \\ &\geq v(S). \end{aligned} \quad (25)$$

Hence, formula (20) satisfies formula (5).  $\square$

To solve the Shapley value correction factor of the federated income distribution, the AHP is used to assign the importance degree according to the scale of 1-9. The explanation of scale 1-9 is presented in Table 1 [39].

In Table 1, index indicates the intensity of importance on an absolute scale. The judgment matrix in this paper is constructed by the joint score of the expert group, and they score according to the rules in Table 1. These experts have a strong background in economics, computer, and mathematics. Hence, according to the joint score of the expert group, the judgment matrix  $A(a_{ij})_{n \times n}$  is obtained.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad (26)$$

and satisfied  $a_{ij} > 0$ ,  $a_{ji} = 1/a_{ij}$ , and  $a_{ii} = 1$ . The matrix  $B(b_{ij})_{n \times n}$  is given by normalizing, and according to each column of the matrix  $A(a_{ij})_{n \times n}$ ,  $b_{ij} = a_{ij} / \sum_{i=1}^n a_{ij}$ ,  $\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{in}$ , comes from each  $i$ -th row element of matrix  $B(b_{ij})_{n \times n}$

TABLE 1: The explanation of index in expert judgment matrix.

Index	Explanation
1	The factors $i$ and $j$ are equally essential
2	The factor $i$ is more essential than $j$ slightly
3	The factor $i$ is more essential than $j$ obviously
4	The factor $i$ is more essential than $j$ strongly
5	The factor $i$ is more essential than $j$ extremely
2, 4, 6, 8	The index between adjacent indexes above
Reciprocal	The value $a_{ij}$ is obtained by factor $i$ and factor $j$ , and the value $a_{ji} = 1/a_{ij}$ is obtained by $j$ comparing factor with factor $i$

and represents the measurement value of  $n$  factors affecting federated revenue;  $\alpha_{ij}$  represents the  $j$ -th factor affecting the individual's federated income, and  $\sum_{i,j=1}^n \alpha_{ij} = 1$ .

$$B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{bmatrix}. \quad (27)$$

Sum matrix  $B(b_{ij})_{n \times n}$  is according to row vector,  $c_i = \sum_{j=1}^n b_{ij}$ , and achieves the vector  $C$ .

$$C = (c_1, c_2, \dots, c_n), \quad (28)$$

and then, normalizing the vector  $C$ ,  $w_i = c_i / \sum_{i=1}^n c_i$ . Consequently, the weight value  $w$  of judgment matrix  $A(a_{ij})_{n \times n}$  is obtained.

$$w = (w_1, w_2, \dots, w_n)^T, \quad \sum_{i=1}^n w_i = 1. \quad (29)$$

Since the judgment matrix  $A(a_{ij})_{n \times n}$  is established based on the experience and estimation of experts, some differences will appear between the judgment matrix  $A$  and the real situation, and it needs to test the consistency of the matrix  $A(a_{ij})_{n \times n}$ . If the matrix  $A(a_{ij})_{n \times n}$  is consistent, its eigenvector  $w = (w_1, w_2, \dots, w_n)^T$  can truly reflect the reasons that affect federated income. In this case, we can use the random consistency index (RI) [38, 39] in Table 2.

$$\begin{cases} \lambda_{\max} = \sum_{i=1}^n \frac{(Aw)_i}{nw_i}, \\ \text{CI} = \frac{(\lambda_{\max} - n)}{(n-1)}, \\ \text{CR} = \frac{\text{CI}}{\text{RI}}, \end{cases} \quad (30)$$

where  $\lambda_{\max}$  is the largest eigenvalue of the matrix  $A(a_{ij})_{n \times n}$ , the CI represents public consistency index, RI is the average

TABLE 2: Random consistency index.

Order	1	2	3	4	5	6	7	8	9
RI	0.00	0.00	0.52	0.89	1.11	1.25	1.35	1.4	1.45

random consistency index, and CR is the consistency ratio. As we refer to Table 2, matrix  $A(a_{ij})_{n \times n}$  needs to be satisfied with the consistency.

$$\text{CR} = \begin{cases} \text{The matrix } A(a_{ij})_{n \times n} \text{ is just what we need,} & \frac{\text{CI}}{\text{RI}} < 0.1, \\ \text{The matrix } A(a_{ij})_{n \times n} \text{ needs to be adjusted,} & \frac{\text{CI}}{\text{RI}} > 0.1. \end{cases} \quad (31)$$

## 5. Numerical Results

We assume that there are three banks: bank 1, bank 2, and bank 3, which are taking part in the FL model. The incomes of three banks not participating in FL are  $v(1) = 150$ ,  $v(2) = 250$ , and  $v(3) = 350$ . If bank 1 and bank 2 coalition can make income  $v(1, 2) = 400$ , bank 1 and bank 3 coalition can make income  $v(1, 3) = 800$ , bank 2 and bank 3 coalition can make income  $v(2, 3) = 700$ , and banks 1, 2, and 3 coalition can make income  $v(1, 2, 3) = 1500$ . After the three banks unite together, the income distribution of each bank is solved.

*5.1. Illustrating Result of the Shapley Value Method.* In this subsection, the Shapley value method does not consider any factors to the federated income but distribute the income according to the average distribution method. Therefore, we can calculate the income distribution value of three banks after FL according to the relevant knowledge of cooperative game theory and Equations (6) and (7). The details are shown in Table 3.

Therefore, add the last row of Table 3, and we achieve the federated income value allocated by bank 1 which is the following:  $\varphi_1(v, t) = 416.67$ ; similarly, the federated income value allocated by bank 2 is as follows:  $\varphi_2(v, t) = 416.67$ , and the federated income value allocated by bank 3 is as follows:  $\varphi_3(v, t) = 666.67$ .

*5.2. Illustrating Result of the Enhanced Shapley Value Method.* In the FL incentive model, the income distribution of the federated bank depends not only on the marginal contribution but also on other factors that affect the total income. In this example, we mainly consider the risk factor, data quality, effort degree, and cost input. For banks with higher risk, higher input data quality, higher level of cooperation, and perhaps higher-cost investment, the weight of benefit distribution should be appropriately increased. According to the enhanced Shapley value model and the following AHP flow chart in Figure 2, experts give scores to the influence factors according to Table 1, and we achieve the judgment matrix  $A, B_1, B_2, B_3, B_4$ . Here, matrix  $A$  is the weight matrix of reference layer  $B$  to target layer.

Through the weight matrix  $A$ , the weight value of reference layer  $B$  to target layer  $A$  can be calculated. Similarly, we

TABLE 3: Bank 1 federated income calculation.

S	{1}	{1, 2}	{1, 3}	{1, 2, 3}
$\nu(S)$	150	400	800	1500
$\nu(S \setminus \{1\})$	0	250	350	700
$\nu(S) - \nu(S \setminus \{1\})$	150	150	450	800
$ S $	1	2	2	3
$w( S )$	1/3	1/6	1/6	1/3
$w( S )[\nu(S) - \nu(S \setminus \{1\})]$	150/3	150/6	450/6	800/3

can obtain the weight matrix  $B_1, B_2, B_3, B_4$  of scheme layer C to reference layer B and the corresponding weight. These weight values of influence factors and the corresponding  $\lambda_{\max}$ , and CR values are solved in the following sheet.

Suppose three banks construct judgment matrix according to their own actual situation, and applying to formulas (20)–(31), we achieve the following conclusions:

(1) Expert judgment matrix A – B and weight  $w$

A	$B_1$	$B_2$	$B_3$	$B_4$
$B_1$	1	1	1/3	1/5
$B_2$	1	1	1/4	1/3
$B_3$	3	4	1	1/4
$B_4$	5	3	4	1

$$w = \begin{pmatrix} 0.0918 \\ 0.1040 \\ 0.2546 \\ 0.5496 \end{pmatrix}$$

$$\lambda_{\max} = 4.2556, CI = 0.0852, CR = 0.0957 < 0.1;$$

(2) Expert judgment matrix  $B_1 - C$  and weight  $w_1^{(1)}$

$B_1$	$C_1$	$C_2$	$C_3$
$C_1$	1	5	3
$C_2$	1/5	1	1/4
$C_3$	1/3	4	1

$$w_1^{(1)} = \begin{pmatrix} 0.6267 \\ 0.0936 \\ 0.2797 \end{pmatrix}, \lambda_{\max} = 3.0858, CI = 0.0429, CR = 0.0825 < 0.1;$$

(3) Expert judgment matrix  $B_2 - C$  and weight  $w_2^{(1)}$

$B_2$	$C_1$	$C_2$	$C_3$
$C_1$	1	7	7
$C_2$	1/7	1	2
$C_3$	1/7	1/2	1

$$w_2^{(1)} = \begin{pmatrix} 0.7732 \\ 0.1392 \\ 0.0877 \end{pmatrix}, \lambda_{\max} = 3.0536, CI = 0.0268, CR = 0.0516 < 0.1;$$

(4) Expert judgment matrix  $B_3 - C$  and weight  $w_3^{(1)}$

$B_3$	$C_1$	$C_2$	$C_3$
$C_1$	1	9	1/2
$C_2$	1/9	1	1/8
$C_3$	2	8	1

$$w_3^{(1)} = \begin{pmatrix} 0.3743 \\ 0.0545 \\ 0.5712 \end{pmatrix}, \lambda_{\max} = 3.0735, CI = 0.03675, CR = 0.0707 < 0.1;$$

(5) Expert judgment matrix  $B_4 - C$  and weight  $w_4^{(1)}$

$B_4$	$C_1$	$C_2$	$C_3$
$C_1$	1	3	1
$C_2$	1/3	1	1/4
$C_3$	1	4	1

$$w_4^{(1)} = \begin{pmatrix} 0.4161 \\ 0.1260 \\ 0.4579 \end{pmatrix}, \lambda_{\max} = 3.0092, CI = 0.0046, CR = 0.0088 < 0.1.$$

Consequently, we let

$$w^{(1)} = \left( w_1^{(1)}, w_2^{(1)}, w_3^{(1)}, w_4^{(1)} \right) = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \end{bmatrix}, \quad (32)$$

and then, using formula (30), we can calculate

$$\begin{aligned} P &= w^{(1)}w \\ &= \begin{bmatrix} 0.6267 & 0.7732 & 0.3747 & 0.4161 \\ 0.0936 & 0.1392 & 0.0545 & 0.1260 \\ 0.2797 & 0.0877 & 0.5712 & 0.4579 \end{bmatrix} \begin{bmatrix} 0.0918 \\ 0.1040 \\ 0.2546 \\ 0.5496 \end{bmatrix} \\ &= \begin{bmatrix} 0.4620 \\ 0.1062 \\ 0.4319 \end{bmatrix}, \end{aligned} \quad (33)$$

and  $\Delta P_i = P_i - 1/n = (0.4620, 0.1062, 0.4319)^T - 1/3 = (0.1287, -0.2271, 0.0986)^T$ . According to formula (22),  $\Delta P_2 < 0$  means that the second bank's enthusiasm in the FL process is significantly lower than the average level of the federated banks and should be punished. Therefore, using formulas (20) and (21) can calculate the income distribution of the three banks in FL as follows:

$$\begin{aligned} \phi_1'(v, t) &= \frac{2500}{6} + 1500 \times 0.1287 = 609.72, \\ \phi_2'(v, t) &= \frac{2500}{6} + 1500 \times (-0.2271) = 76.02, \\ \phi_3'(v, t) &= \frac{4000}{6} + 1500 \times 0.0986 = 814.57. \end{aligned} \quad (34)$$

Through the numerical calculation of the enhanced Shapley method, we achieve the expected result which is the following:

$$\sum_{i=1}^3 \phi_i'(v, t) = 609.72 + 76.02 + 814.57 \approx 1500. \quad (35)$$

After verification, it is not difficult to find that in the case of the income distribution scheme of the enhanced Shapley value method, the income of the three banks is consistent with the theory.

**5.3. Numerical Comparison.** In this subsection, we compare the results of income distribution using the Shapley value



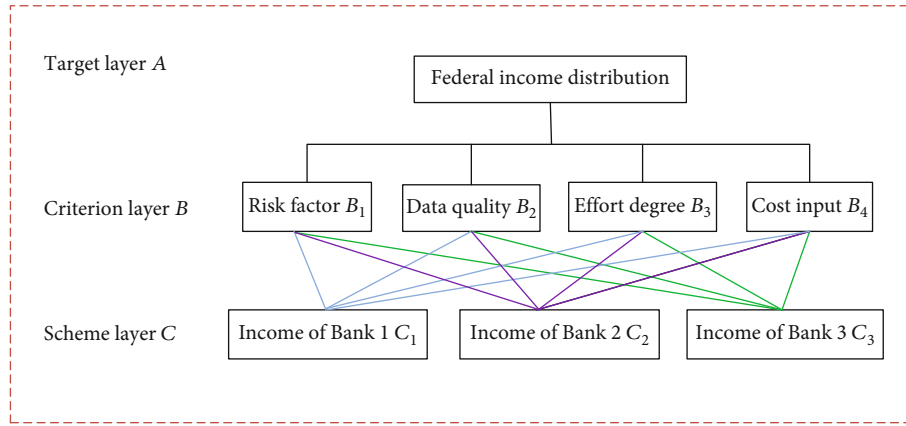


FIGURE 2: AHP flow chart.

TABLE 4: Income comparison.

Participant	Shapley value method		Enhanced Shapley value method	
	Average weight	Income distribution	Enhanced weight	Income distribution
Bank 1	0.3333	416.67	0.4620	609.72
Bank 2	0.3333	416.67	0.1062	76.02
Bank 3	0.3333	666.67	0.4319	814.57

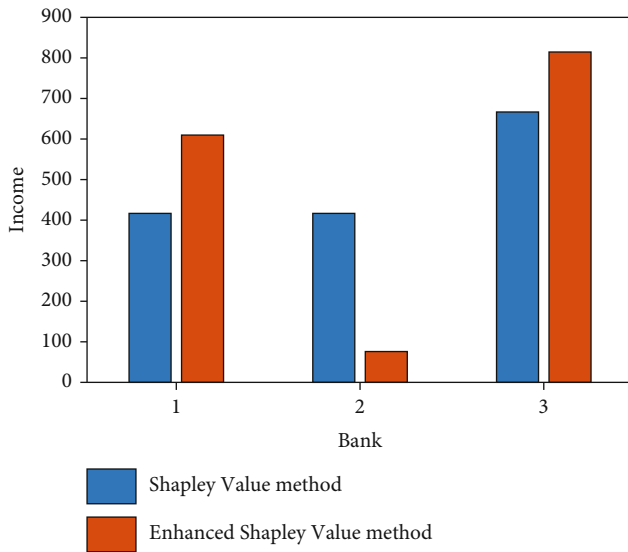


FIGURE 3: Income histogram of federated model.

method with the results of income distribution using the enhanced Shapley value method [40] in Table 4 and simulate their values in Figure 3.

Table 4 indicates that the Shapley value method does not consider any factors and the corresponding weights in the income distribution; however, in the enhanced Shapley value method, we consider the factors that affect income distribution and the corresponding weights. It is not difficult to find out bank 1 has the largest weight, indicating that bank 1 is

the most positive in the whole FL process, followed by the third bank, and the second bank is the worst, which should be punished.

From Figure 3, we can clearly see that income allocated by the Shapley value method shows that the first bank and the second bank have the same income. However, relying on the enhanced Shapley value method, it is easy to find that the second bank is the least profitable.

## 6. Conclusion and Future Work

Federated learning (FL) is a distributed machine learning framework, and how to design an incentive mechanism for federated participants to sustain participate in model training is very important to train an accurate federated model. In this paper, we propose an incentive mechanism using the enhanced Shapley value method, and this incentive mechanism considers four factors that affect the federated income distribution, which can better reflect the fairness of income distribution. Further, the analytic hierarchy process is used to calculate the corresponding weight of four factors. Numerical results verify that the income distribution of all participants can better reflect multiple factor contribution when using the incentive mechanism with the enhanced Shapley value method, which gets a better incentive for federated participants.

In the next step, we will apply the proposed incentive mechanism to solve the problem of participants' income distribution in FL train models. This incentive mechanism can be applied in WeBank, such as financial institutions, insurance companies, and enterprises, for reducing loan risks and improving economic earning significantly in practice.

## Data Availability

The data used to support the finding of this study are included in the article.

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. U1836205), the National Natural Science Foundation of China (No. 71961003), the Science and Technology Program of Guizhou Province (No. Guizhou-Science-Contract-Platform-Talent [2020]5017), the Natural Science Foundation of Department of Education of Guizhou Province (Nos. Qian-Education-Contact-Telent [2013]09 and Qian-Education-Contact-KY [2021]140), the 13th Five-Year National Cryptography Development Foundation (No. MMJJ20170129), the Research Project of Guizhou University for Talent Introduction (No. [2020]61), the Cultivation Project of Guizhou University (No. [2019]56), and the Open Fund of Key Laboratory of Advanced Manufacturing Technology, Ministry of Education, GZUAMT2021KF[01].

## References

- [1] J. B. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.
- [2] Q. Li, B. Xia, H. P. Huang, Y. H. Zhang, and T. Zhang, "TRAC: traceable and revocable access control scheme for mHealth in 5G-enabled IIoT," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 1551–3203, 2021.
- [3] K. Bonawitz, H. Eichner, W. Grieskamp et al., "Towards federated learning at scale: system design," 2019, <http://arxiv.org/abs/1902.01046>.
- [4] Q. Yang, Y. Liu, T. J. Chen, and Y. X. Tong, "Federated machine learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
- [5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.
- [6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Conference on Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.
- [7] J. Konecny, H. B. McMahan, D. Ramage, and P. Richt, "Federated optimization: distributed machine learning for on-device intelligence," 2016, <http://arxiv.org/abs/1610.02527>.
- [8] J. Konečný, M. M. HB, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," 2016, <http://arxiv.org/abs/1610.05492>.
- [9] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," 2016, <http://arxiv.org/abs/1602.05629>.
- [10] J. B. Xiong, X. H. Chen, Q. Yang, L. Chen, and Z. Q. Yao, "A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2347–2360, 2020.
- [11] Y. Liu and J. H. Wei, "Incentives for federated learning: a hypothesis elicitation approach," 2020, <http://arxiv.org/abs/2007.10596>.
- [12] W. Y. B. Lim, Z. H. Xiong, C. Miao et al., "Hierarchical incentive mechanism design for federated machine learning in mobile networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9575–9588, 2020.
- [13] Y. F. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6360–6368, 2020.
- [14] H. Yu, Z. L. Liu, Y. Liu et al., "A sustainable incentive scheme for federated learning," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 58–69, 2020.
- [15] P. Kairouz, M. M. HB, B. Avent et al., "Advances and open problems in federated learning," 2019, <http://arxiv.org/abs/1912.04977>.
- [16] H. Yu, Z. Liu, Y. Liu et al., "A fairness-aware incentive scheme for federated learning," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 393–399, New York, NY, USA, 2020.
- [17] J. Xiong, R. Ma, L. Chen, Y. Tian, L. Lin, and B. Jin, "Achieving incentive, security, and scalable privacy protection in mobile crowdsensing services," *Wireless communications and mobile computing*, vol. 2018, Article ID 8959635, 12 pages, 2018.
- [18] J. Kang, Z. H. Xiong, D. Niyato, S. Xie, and J. S. Zhang, "Incentive mechanism for reliable federated learning: a joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10700–10714, 2019.
- [19] J. Kang, Z. H. Xiong, D. Niyato, H. Yu, Y. C. Liang, and D. Kim, "Incentive design for efficient federated learning in mobile networks: a contract theory approach," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, Singapore, 2019.
- [20] N. H. Tran, W. Bao, A. Y. Zomaya, M. N. H. Nguyen, and C. Hong, "Federated learning over wireless networks: optimization model design and analysis," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, Paris, France, 2019.
- [21] L. U. Khan, S. R. Pandey, N. H. Tran et al., "Federated learning for edge networks: resource optimization and incentive mechanism," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 88–93, 2020.
- [22] Y. Sarikaya and O. Ercetin, "Motivating workers in federated learning: a Stackelberg game perspective," *IEEE Networking Letters*, vol. 2, no. 1, pp. 23–27, 2020.
- [23] T. Li, M. Sanjabi, and V. Smith, "Fair resource allocation in federated learning," 2020, <http://arxiv.org/abs/1905.10497>.
- [24] A. Richardson, A. F. Ratsikas, and B. Faltings, "Rewarding high-quality data via influence functions," 2019, <http://arxiv.org/abs/1908.11598>.
- [25] Z. Chen, Z. Liu, K. L. Ng, H. Yu, Y. Liu, and Q. Yang, *A Gamified Research Tool for Incentive Mechanism Design in Federated Learning*, Springer, Cham, 2020.
- [26] J. B. Xiong, M. F. Zhao, M. Z. A. Bhuiyan, L. Chen, and Y. L. Tian, "An ai-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2021.

- [27] J. E. Augustine, N. Chen, E. Elkind, A. Fanelli, N. Gravin, and D. Shiryayev, "Dynamics of profit-sharing games," *Internet Mathematics*, vol. 11, no. 1, pp. 1–22, 2015.
- [28] J. Xiong, X. Chen, Y. Tian, R. Ma, L. Chen, and Z. Yao, "MAIM: a novel incentive mechanism based on multi-attribute user selection in mobile crowdsensing," *IEEE Access*, vol. 6, pp. 65384–65396, 2018.
- [29] L. S. Shapley, "17. A Value for n-Person Games," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.
- [30] T. S. Song, Y. X. Tong, and S. Y. Wei, "Profit allocation for federated learning," in *2019 IEEE International Conference on Big Data (Big Data)*, Los Angeles, CA, USA, 2019.
- [31] G. Wang, "Interpret federated learning with Shapley values," 2019, <http://arxiv.org/abs/1905.04519>.
- [32] S. Yang, F. Wu, S. J. Tang, X. Gao, B. Yang, and G. Chen, "On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 832–847, 2017.
- [33] R. Jia, D. Dao, B. Wang et al., "Towards efficient data valuation based on the Shapley value," 2019, <http://arxiv.org/abs/1902.10275>.
- [34] Y. Liu, Z. Ai, S. Sun, S. Zhang, Z. Liu, and H. Yu, *Fedcoin: A Peer-to-Peer Payment System for Federated Learning*, Springer, Cham, 2020.
- [35] R. Aumann and B. Peleg, "Von Neumann-Morgenstern solutions to cooperative games without side payments," *Bulletin of the American Mathematical Society*, vol. 66, no. 3, pp. 173–179, 1960.
- [36] R. J. Aumann and J. H. Dreze, "Cooperative games with coalition structures," *International Journal of Game Theory*, vol. 3, no. 4, pp. 217–237, 1974.
- [37] D. Butnariu, "Stability and Shapley value for an n-persons fuzzy game," *Fuzzy Sets and Systems*, vol. 4, no. 1, pp. 63–72, 1980.
- [38] R. W. Saaty, "The analytic hierarchy process—what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3-5, pp. 161–176, 1987.
- [39] T. Saaty, "How to make a decision: the analytic hierarchy process," *European Journal of Operational Research*, vol. 48, no. 1, pp. 9–26, 1990.
- [40] E. Kalai and D. Samet, "On weighted Shapley values," *International Journal of Game Theory*, vol. 16, no. 3, pp. 205–222, 1987.
- [41] D. Mishra and D. Veeramani, "Vickrey-Dutch procurement auction for multiple items," *European Journal of Operational Research*, vol. 180, no. 2, pp. 617–629, 2007.
- [42] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, 2010.

## Research Article

# Local Epochs Inefficiency Caused by Device Heterogeneity in Federated Learning

Yan Zeng <sup>1,2,3</sup>, Xin Wang <sup>4</sup>, Junfeng Yuan <sup>1</sup>, Jilin Zhang <sup>1,2,3</sup> and Jian Wan <sup>1</sup>

<sup>1</sup>School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

<sup>2</sup>Key Laboratory of Complex Systems Modeling and Simulation Ministry of Education, Hangzhou 310018, China

<sup>3</sup>Zhejiang Engineering Research Center of Data Security Governance, Hangzhou 310018, China

<sup>4</sup>HDU-ITMO Joint Institute, Hangzhou Dianzi University, Hangzhou 310018, China

Correspondence should be addressed to Jilin Zhang; [jilin.zhang@hdu.edu.cn](mailto:jilin.zhang@hdu.edu.cn)

Received 5 August 2021; Revised 17 November 2021; Accepted 29 November 2021; Published 6 January 2022

Academic Editor: Jinbo Xiong

Copyright © 2022 Yan Zeng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Federated learning is a new framework of machine learning, it trains models locally on multiple clients and then uploads local models to the server for model aggregation iteratively until the model converges. In most cases, the local epochs of all clients are set to the same value in federated learning. In practice, the clients are usually heterogeneous, which leads to the inconsistent training speed of clients. The faster clients will remain idle for a long time to wait for the slower clients, which prolongs the model training time. As the time cost of clients' local training can reflect the clients' training speed, and it can be used to guide the dynamic setting of local epochs, we propose a method based on deep learning to predict the training time of models on heterogeneous clients. First, a neural network is designed to extract the influence of different model features on training time. Second, we propose a dimensionality reduction rule to extract the key features which have a great impact on training time based on the influence of model features. Finally, we use the key features extracted by the dimensionality reduction rule to train the time prediction model. Our experiments show that, compared with the current prediction method, our method reduces 30% of model features and 25% of training data for the convolutional layer, 20% of model features and 20% of training data for the dense layer, while maintaining the same level of prediction error.

## 1. Introduction

In recent years, with the rapid development of 5G, “Interconnection of All Things” has become a trend of information technology. The speed of information circulation on the Internet has reached an unprecedented level. The rapid information circulation has brought a dramatic increase in data and promoted the development of big data and artificial intelligence technology. However, the interconnection of more devices brings higher security risks. How to enjoy the benefits brought by artificial intelligence on the premise of ensuring data privacy has become a challenge.

In 2016, McMahan et al. [1] first proposed the concept of federated learning to respond to the challenge. Multiple clients can jointly train the model under the coordination of a central server or service provider in federated learning. Each client uses its data to train the local model and uploads

the parameters of local model to the server for model aggregation to achieve the global model. Therefore, the original data of each client is stored locally without exchange or transmission. At present, federated learning has been widely applied to optimize the user experience on the premise of protecting privacy in Google, Apple, and other enterprises. For example, Google has widely used federated learning in Gboard [2], Pixel mobile phone [3], and Android Message [4], so as IOS13 [5] of Apple.

In the FedAvg algorithm proposed by McMahan et al., the central server generates a global model after aggregation in each synchronization and then distributes the global model to some clients which are selected randomly. When the clients received the global model, they use their own data to train their local models with the parameters of global model in the specified epochs. After the local models are trained, the clients will send them to the server, and the

server will execute model aggregation with the weighted average strategy. The global model will gradually converge after several synchronizations. According to the FedAvg, the update of global model in each synchronization can be expressed by formula (1), where  $x_g^{(t)}$  represents the parameters of the global model in the  $t$ -th synchronization of model aggregation,  $n_i$  is the amount of client  $i$ 's data, and  $n$  denotes the amount of all clients' data.  $\eta$  is the client learning rate,  $\tau_i$  represents the number of client  $i$ 's local updates, which can be obtained by  $\lfloor E \cdot n_i / B \rfloor$ ,  $E$  is the number of local epochs,  $B$  is the mini-batch size of the client,  $x_i^{(t,k)}$  represents the parameters of local model in  $k$ -th local update on  $i$ -th client after the  $t$ -th synchronization, and  $g_i$  denotes the local model's parameters gradient of client  $i$ .

$$x_g^{(t+1)} - x_g^{(t)} = - \sum_{i=1}^m \frac{n_i}{n} \cdot \eta \sum_{k=0}^{\tau_i-1} g_i(x_i^{(t,k)}), \text{ where } \tau_i = \left\lfloor \frac{E \cdot n_i}{B} \right\rfloor. \quad (1)$$

In each synchronization of global model updating, FedAvg sets the same number of local epochs  $E$  for all clients participating in model aggregation, which will lead to inefficient training as the difference in training speed. A lot of other federated learning algorithms are also set in the same way, such as FedProx [6]. Although FedNova [7] has implemented the training of the global model under the situation of different local epochs, it randomly sets the number of local epochs, which may set a small number of local epochs for clients with high training speed, and leads to the idling problem. But, FedNova gives us an inspiration: if we can predict the model training time of clients, the local epochs will be dynamically set according to the predicted training time, which can reflect the training speed of clients.

As the idling of faster clients in federated learning will prolong the training of global model, we propose to predict the training time of deep learning models on heterogeneous clients to guide dynamically set the number of local epochs. In the deep learning task, the training time may be affected by the amount of training data and the setting of hyperparameters. We call the factors in training data and hyperparameters that may affect the training time as model features. Justus et al. [8] have trained a multilayer perceptron (MLP) to predict the training time of layers in the neural network using the model features and training time collected on different GPUs. They divide model features into layer features and predict the training time of the complete model by accumulating the prediction results of multiple layers. However, when the structure of the model is very complex or the number of layers is very large, collecting many features in each layer will increase the burden of the system and hinder the convergence progress of global model. What is more, when a new device is added to the federated learning system, it is necessary to collect a large amount of high dimension training data on this device to tune the current prediction model, which is very time-consuming and will lead to long-term failure of training time prediction for the new device.

To solve these problems, we propose a method based on deep learning to reduce the number of model features and the amount of training data required by training time prediction. We first design a neural network to extract the influence of model features on training time, which can accurately interpret the relationship between model features and training time. Then, we propose a dimensionality reduction rule to extract the key features based on the influence of model features on training time. A large number of experiments show that compared with the current prediction method, the features of the convolutional layer and dense layer can be reduced by 30% and 20%, respectively, and the error of training time prediction still maintains the original level. At the same time, 25% convolutional layer training data and 20% dense layer training data are reduced, which speeds up the adaptation of the time prediction model to the new device.

The rest of this paper is arranged as follows: in Section 2, we discuss the related work in time prediction; in Section 3, we introduce the method proposed in this work, including our neural network and dimensionality reduction rule. We also provide an algorithm to dynamically set the number of local epochs in this section. To verify our work, we set up a large number of experiments and interpret the experimental results in Section 4. Finally, we provide the summary of all work in Section 5.

## 2. Related Work

At first, machine learning regression algorithms are often used to predict time series, such as linear regression, random forest, and GBDT. Edelman et al. [9] use a linear regression model to predict the execution time of surgery; Wang et al. [10] train regression decision trees to predict the arrival time of buses by using the nearest neighbor-based random forest algorithm; Cheng et al. [11] use GBDT to predict traffic time in different time ranges and find the variables which have a great impact on prediction error. These regression methods have good universality, and they can be used in many fields. But, the error range of their prediction is very large, so they can only be applied to scenarios with low sensitivity to time fluctuations.

To narrow the error range of time series prediction, some scholars have proposed the method with specific domain knowledge to predict time series. It constructs mathematical models to achieve time series prediction, by studying the calculation characteristics of the specific domain such as PALEO [12] and Optimus [13] method. PALEO is a method to predict the computing time by counting the number of floating-point operations. It counts the number of floating-point operations required in a model training epoch and multiplies the number by a scale factor to predict the training time. However, PALEO assumes that the whole training process of the model is linearly related to the number of floating-point operations, ignoring some operations that are not, such as parameter transmission. Unlike PALEO, Optimus mathematically summarizes the factors that affect model training, establishes a performance model to evaluate the training speed, and can predict the model

convergence according to online resources. Compared with the regression methods, these works reduce the prediction error range of model training time to a certain extent, but the mathematical model established for the training is fuzzy and it ignores some factors which contribute greatly to the training time, resulting in instability of the prediction.

Because of the excellent performance of deep learning models, researchers began to use deep learning methods for predicting time series, and trying to further reduce the error of time series prediction. Xu et al. [14] creatively combine linear regression and deep belief network (DBN) to predict time series; PreVIOUS [15] trains the MLP model to predict the inference time of convolutional neural networks according to the throughput and energy consumption of the Internet of Things vision device; Petersen et al. [16] design a neural network mixed with convolutional layers and LSTM layers to accurately predict the bus arrival time. These works have achieved high prediction accuracy, but their application in the training time prediction of deep learning models is limited by the specific model structure. Their time prediction models can only predict those network structures contained in their training data (such as VGG [17], ResNet [18], or user-defined network). When a network with a new structure is encountered, their models need to be retrained, that is, they cannot apply to other new deep learning models. Although Fathom [19] has proved that the inference time of a model can be estimated by another model with a similar structure and known performance, its prediction is very rough, and it is still to be proved that whether this method can be used in the prediction of training time. In order to accurately predict the training time of networks with different structures, Justus et al. divide the neural network into layers and classify these layers (such as convolutional layer and dense layer) according to the structural characteristics, then collect the layer features and train an MLP model to predict the training time of a single layer in the neural network, which can achieve high prediction accuracy. This method has good generality. When a network with a new structure is encountered, it only needs to predict the training time of layers according to the layer model features, and then the training time of the whole model can be predicted by accumulating the training time of layers. However, there are some problems when Daniel Justus' method is applied to federated learning: (1) the relationship between model features and training time is not accurate. They assumed that almost every model feature is necessary for training time prediction, including features that have no or little impact on the final result. (2) Too many unimportant features need to be collected. When the neural network is very deep, collecting features in every layer will increase the burden of the federated learning system, and it is usually difficult to obtain all details of clients' models. (3) High training cost caused by too much redundant training data. Unimportant features produce a lot of redundant training data, which increases the transfer training cost of the time prediction model on the newly added device and reduces the training efficiency of the global model.

To solve the above problems, we propose a training time prediction method based on deep learning, which can reduce

the required model features and training data on the premise of ensuring low prediction error and improve the feasibility of practical application in federated learning. The contributions of this paper are as follows:

- (1) We design a neural network to extract the influence of model features on training time according to the characteristics of the deep learning model, which provides an effective analysis of the relationship between model features and training time
- (2) We propose a dimensionality reduction rule to extract the key features that have a great impact on training time according to the influence of features, which can reduce the number of features required for predicting model training time without loss of prediction accuracy. By using the dimensionality reduction rule, 7 dimensions are extracted from convolutional layer features (10 dimensions in total), and 4 dimensions are extracted from dense layer features (5 dimensions in total)
- (3) We train the time prediction model using dimension-reduced datasets. Compared with the method of Justus et al., the training data of the convolutional layer is reduced by 25%, and the training data of the dense layer is reduced by 20%, with the error of prediction remaining at the same level

### 3. Methodology

In this section, we will introduce the overall process and technical details of extracting the influence of model features on training time, dimensionality reduction, and the algorithm of dynamically setting the number of local epochs. First, we prove the feasibility of accumulating the layers' training time for the prediction of the whole network by interpreting the calculation process of training. And we describe the layer features based on the work of Justus et al. in detail. Second, we introduce the structure of the neural network (Here we name our neural network weights model), which is designed for extracting the influence of model features on the training time. Third, we propose the dimensionality reduction rule to extract the key features which have a great impact on training time, based on the influence of model features. Finally, we provide a representative algorithm for dynamically setting the number of local epochs.

*3.1. Feature Analysis.* One training of neural network consists of forward propagation and backward propagation. With the widespread use of Batch Normalization [20] which can speed up the convergence of neural networks, it usually has to perform a batch of forward propagation before one backward propagation. A complete round of training (including multiple batches) of a neural network in the training set is called an epoch. Generally, a model with high accuracy needs to be trained many epochs until the model converges. At present, the method of setting the number of epochs is based on the experience of deep learning engineers.

TABLE 1: The description of layer features.

Kind of features	Features	Description
Common features	Activation function	The activation function of neuron output, the common ones are sigmoid, tanh, and relu, etc.
	Optimizer	The optimization method of the model, the common ones are SGD, Adadelta, Adagrad, momentum, Adam, and RMS prop, etc.
Dense features	Number of inputs	Since the MLP layers are fully connected, the input of each layer comes from the output of the previous layer.
	Number of neurons	The number of neurons.
Convolutional features	Matrix size	The size of the input data.
	Kernel size	The size of convolutional kernel.
	Input depth	The number of input channels.
	Output depth	The number of output channels.
	Stride size	The convolution step size of convolution kernel.
	Input padding	The number of edge padding after convolution.
Hardware features	GPU clock speed	GPU clock cycle speed.
	GPU memory bandwidth	GPU bandwidth.
	GPU core count	The number of GPU processing units, which represents the number of CUDA cores in NVIDIA GPU.

It needs to set the different number of epochs for different models to achieve the specified accuracy. Therefore, it becomes a challenge to accurately predict the training time of models with the different number of epochs. In addition, since the structures of models are heterogeneous, the training time will be significantly different. For example, the training of the convolutional layer is usually more time-consuming than the dense layer. Therefore, it is also a challenge to accurately predict the training time for models with different structures. To solve these problems, Justus et al. proposed a method to predict the training time of different layers in a batch. The training time of the whole model in one batch can be obtained by accumulating the prediction results of layers. And the total training time can be predicted by accumulating the training time in batches. Ulteriorly, we prove the feasibility of predicting the whole model's training time through layers combined with the training characteristics and structural characteristics of the deep learning model.

Usually, a neural network needs to be trained repeatedly on the training set many times, which has obvious iteration characteristics. According to the iteration, the time required for model training can be expressed as formula (2), where  $E$  represents the number of epochs,  $M$  represents the number of batches in an epoch, and  $T_b$  denotes the training time in a batch. The total number of batches in  $E$  epochs can be obtained by  $\lfloor E \cdot n / B \rfloor$ ,  $n$  is the amount of training data, and  $B$  is the size of a batch (i.e., batch size).

$$T = E \cdot M \cdot T_b = \left\lfloor \frac{E \cdot n}{B} \right\rfloor \cdot T_b. \quad (2)$$

One training of the neural network consists of a batch of forward propagation and one backward propagation. Therefore, the time cost of the propagation can be expressed as formula (3),  $t_{\text{forward}}$  represents the time cost of forward prop-

agation,  $t_{\text{backward}}$  represents the time cost of backward propagation, and  $x_i$  is the  $i$ -th training data in a batch.

$$T_b = \left( \sum_{i=1}^B t_{\text{forward}}(x_i) \right) + t_{\text{backward}}. \quad (3)$$

Combining formula (2) and formula (3), the training time of a deep learning model can be described as the following formula:

$$T = \left\lfloor E \cdot \frac{n}{B} \right\rfloor \cdot \left[ \left( \sum_{i=1}^B t_{\text{forward}}(x_i) \right) + t_{\text{backward}} \right]. \quad (4)$$

A neural network is composed of many layers, and the output of the current layer is used as the input of the next layer. Besides the iteration characteristic of the training process, the neural network also has obvious hierarchical structure characteristics. According to this hierarchy characteristic, a complete neural network can be divided into layers, and its forward and backward propagation can be expressed as formulas (5) and (6) on layer level.  $t_{\text{forward}}^l$  and  $t_{\text{backward}}^l$ , respectively, represent the time cost of layer  $l$ 's forward and backward propagation, and  $m$  denotes the number of model layers.

$$t_{\text{forward}}(x_i) = \sum_{l=1}^m t_{\text{forward}}^l(x_i), \quad (5)$$

$$t_{\text{backward}} = \sum_{l=1}^m t_{\text{backward}}^l. \quad (6)$$

Combining formula (4) with formulas (5) and (6), the training time formula of the complete model with the layer's

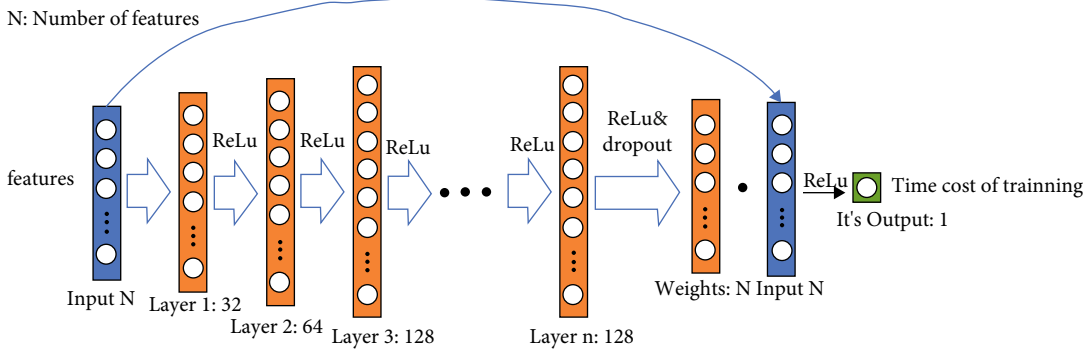


FIGURE 1: The structure of weights model.

TABLE 2: Hyperparameter settings.

Hyperparameter	Value
Initial learning rate	0.1
The decay period of learning rate	40
The decay ratio of learning rate	$2^{-\lfloor \text{epoch}/40 \rfloor}$
L2 regularization	0.00001
Dropout	0.2
Batchsize	128
Epoch	300
Activation	ReLU
Optimizer	Adam
Loss	Mean squared error

training time as the unit can be derived, as shown in the following formula.

$$T = \left[ E \cdot \frac{n}{B} \right] \cdot \left[ \left( \sum_{i=1}^B \sum_{l=1}^m t_{\text{forward}}^l(x_i) \right) + \sum_{l=1}^m t_{\text{backward}}^l \right]. \quad (7)$$

In summary, the training time of a single layer can be used as the basic unit of the whole model's training time. Therefore, for models with number of different epochs, the total training time of the model can be obtained by accumulating training time in a batch; for models with different structures, one batch training of the model can be obtained by accumulating the forward and backward propagation time of layers, well solved the two challenges in training time prediction of models.

In order to predict the training time of a single layer, it is necessary to analyze the layer features of neural networks. First of all, we classify the layer features into common features, dense features, convolutional features, and hardware features according to the device characteristics and computing characteristics. And then we extract the features according to the categories. The layer features are shown in Table 1. Since the deep learning model contains a large number of convolutional layers and dense layers, we mainly focus on the convolutional layer and dense layer in this paper. And we trained the time prediction model for the convolutional layer and the dense layer, respectively, based on the data of

layer features and training time, which are collected on six different types of GPUs (P100, V100, K40, K80, M60, and 1080ti).

**3.2. Model Design.** In the previous section, we analyzed the layer features which may affect the training time of neural networks by parsing the structure of different layers. However, we find that collecting layer features will increase the system overhead for deep-seated neural networks. For example, in terms of the ResNet101 with 100 convolutional layers and 1 dense layer, if we collect 10 features of the convolutional layer and 5 features of the dense layer, we finally need to collect 1005 features to predict the training time of ResNet101. For clients with low computing power, the process of predicting will take a lot of time.

What is more, since there are a large-scale of heterogeneous devices in federated learning, we cannot use the time prediction model to predict the training time of models for newly added devices whose types are not in the set of preset device types. To predict the training time for the new device, we need to tune the parameters of the time prediction model based on the training data collected on this device. But it needs a long time to collect a large number of high dimension training data for tuning the time prediction model. The prediction model cannot quickly adapt to the new device, and the number of local epochs is set to be a fixed value for a long time, which may lead to the clients remaining idle with high training speed.

For cutting down the time cost of predicting and accelerating the adaptation of the time prediction model to new devices, it is necessary to reduce the dimension of layer features and redundant training data. In the case of ensuring high prediction accuracy, we choose to exclude the features that have no or little impact on training time. So, the influence of model features on training time needs to be analyzed.

In order to extract the influence of features, we abstract the relationship between model features and training time as  $f(x) = wx$ , where  $x$  represents the model features,  $f(x)$  denotes the training time of a single layer, and  $w$  denotes the weights of features which can be treated as the influence of features. It should be noted that due to different value ranges of features, and  $w$  cannot represent the real influence



```

Input:  $W$  weights model,  $N$  the number of dataset,  $F$  feature set,  $X_k$  dataset  $k$ ,  $M_d$  the amount of dataset  $d$ 
Output:  $\Theta$  the set of key features
1: Initialize  $\Theta$ 
2: For each dataset
3:    $w_{i,j} \leftarrow \text{GetWeights}(W, X_k)$ 
4:    $r_{i,j} \leftarrow \text{rank}(w_{i,j})$ 
5: End for
6: For  $j$  in  $F$  do
7:    $\text{meanrank}_j \leftarrow (1/N) \sum_{d=1}^N (1/M_d) \sum_{i=1}^{M_d} r_{i,j}$ 
8:    $\text{meanstd}_j \leftarrow (1/N) \sum_{d=1}^N \text{RankStd}(d, j)$ 
9:   If  $\text{meanrank}_j > s$  and  $\text{meanstd}_j < r$  then
10:      $\text{AddNewFeature}(\Theta, j)$ 
11:   End if
12: End for
13: Return  $\Theta$ 

```

ALGORITHM 1: Dimensionality reduction.

```

Input: The  $K$  clients are indexed by  $i$ ;  $B$  local minibatch size,  $T$  Communication time window,  $M$  time prediction model.
1: Server executes:
2: initialize  $x_g^{(0)}$ 
3: for each round  $t, t=1,2,\dots,N$  do
4:    $m \leftarrow \text{Max}(C \cdot K, 1)$ 
5:    $S_t \leftarrow$  (random set of  $m$  clients)
6:   for each client  $i \in S_t$  in parallel: do
7:      $\eta d_i^{(t)} \leftarrow \text{ClientUpdate}(i, x_g^{(t)}, M)$ 
8:   end for
9:    $x_g^{(t+1)} \leftarrow x_g^{(t)} - \tau_{eff}^{(t)} \sum_{i=1}^m (n_i/n) \eta d_i^{(t)}$ 
10: end for
11: ClientUpdate( $i, x_g^{(t)}, M$ ):
12:  $x_i^{(t)} \leftarrow x_g^{(t)}$  // Client receives the global model
13:  $N_i \leftarrow$  (number of batches per epoch divided by  $B$ )
14:  $f \leftarrow \text{GetFeatures}(i, x_i^{(t)})$ 
15:  $T_{train} \leftarrow \text{GetTrainingTime}(M, f)$ 
16:  $E_i \leftarrow \lfloor T/T_{train} \rfloor$ 
17: for each epoch from  $e$  to  $E_i$  do
18:   for each batch from  $k$  to  $N_i$  do
19:      $x_i^{(t+1,k)} \leftarrow x_i^{(t,k)} - \eta g_i^{(t,k)}$ 
20:      $d_i^{(t,k)} \leftarrow d_i^{(t,k)} + (1/E_i * N_i) g_i(x_i^{(t,k)})$ 
21:      $d_i^{(t)} \leftarrow d_i^{(t)} + d_i^{(t,k)}$ 
22:   end for
23: end for
24: return  $\eta d_i^{(t)}$  to the Server

```

ALGORITHM 2: Dynamically set number of local epochs.

of features when simply taking the original feature data as  $x$ . The value of  $x$  should be the standardized feature data.

In related work, we introduced some machine learning regression models, including linear models and nonlinear models. The linear regression model can directly extract the features' weights  $w$ , but it underperforms in time prediction. The nonlinear models are difficult to extract  $w$  since their weights are implicitly dispersed in the model param-

eters. To get the weights of features accurately, we design a weights model which can use a neural network to explicitly extract features. The structure of the weights model is shown in Figure 1. See Table 2 for the hyperparameter settings of weights model.

As can be seen from Figure 1, the input of the weights model is the standardized feature data, and the output is the predicted training time. Each neuron of layers is

TABLE 3: The description of datasets.

Datasets	Description
P100_Conv	Convolutional layer feature dataset of P100.
P100_Dense	Dense layer feature dataset of P100.
V100_Conv	Convolutional layer feature dataset of V100.
V100_Dense	Dense layer feature dataset of V100.
K40_Conv	Convolutional layer feature dataset of K40.
K40_Dense	Dense layer feature dataset of K40.
All_Conv	Convolutional layer feature datasets of six different types of GPUs, including P100, V100, K40, K80, M60, and 1080ti (stacked dataset).
All_Dense	Dense layer feature datasets of six different types of GPUs, including P100, V100, K40, K80, M60, and 1080ti (stacked dataset).

activated by ReLu and then output. For ensuring the convergence of the model, the settings of layer 1 to layer  $N$  are consistent with the hidden layers of Justus et al.'s time prediction model. In order to learn the weights of features, we multiply the output of the weights layer and the input layer, and then output after ReLu activation. The weights of features  $w$  calculated from layer 1 to weights layer is multiplied by the feature data  $x$  to form  $f(x) = wx$ .

Different from the simple linear model whose  $w$  is fixed, the weight extracted by weights model will change with the input data, which can fit the training data better. In weights model, for each input data  $x_i$ , the output is  $f(x_i) = g(x_i)x_i$ , where  $g(x_i)$  is a weight function that the weight will change with the input data. It can be obviously found that the weight  $g(x_i)$  is obtained by the deep learning model, which means that it can not only benefit from the high accuracy of the nonlinear model but also use the output of weight layer to obtain the weight data explicitly. Justus et al. have proved that their MLP model has higher prediction accuracy than the linear regression model, but it cannot characterize the performance of our weights model, which is the reconstructed MLP. Therefore, we conduct the comparative experiments to prove the superiority of weights model (see Section 4.2 for the results).

**3.3. Dimensionality Reduction Rule.** We have introduced the weights model  $g(x_i)$  which used to extract the weights of features in the last section. As the weights  $g(x_i)$  will change with the input data  $x_i$ , the order of features' influence (i.e., weights ranking) may fluctuate. For example, for input data  $x_1$ , the feature batchsize has the greatest influence, but for data  $x_2$ , its influence may be the smallest. The prediction error will be further expanded and the influence of features cannot be measured uniformly because of the fluctuation of  $(x_i)$ . Therefore, we use the average ranking of feature weights and the average standard deviation of weights ranking to comprehensively analyze the influence of features. The average ranking of feature weights can represent the overall contribution of features to training time and the average standard deviation of weights ranking can measure the fluctuation of weights. According to these two metrics, we propose a dimensionality reduction rule to extract the

TABLE 4: Layer features in the convolutional layer.

Features	Description
Batchsize	The size of a batch.
Elements_matrix	The number of elements of input.
Elements_kernel	The number of elements of convolutional kernel.
Channels_in	The number of input channels.
Channels_out	The number of output channels.
Padding	The number of edge padding.
Strides	The step size of convolution.
Use_bias	Whether to use bias, 0: not use 1: use.
Opt_SGD	Whether to use SGD optimizer, 0: not use 1: use.
Opt_Adadelta	Whether to use Adadelta optimizer, 0: not use 1: use.
Opt_Adagrad	Whether to use Adagrad optimizer, 0: not use 1: use.
Opt_Momentum	Whether to use momentum optimizer, 0: not use 1: use.
Opt_Adam	Whether to use Adam optimizer, 0: not use 1: use.
Opt_RMSProp	Whether to use RMSProp optimizer, 0: not use 1: use.
Act_relu	Whether to use ReLu activation, 0: not use 1: use.
Act_tanh	Whether to use tanh activation, 0: not use 1: use.
Act_sigmoid	Whether to use sigmoid activation, 0: not use 1: use.

TABLE 5: Layer features in the dense layer.

Features	Description
Batchsize	The size of a batch.
Dim_input	The dimension of input.
Dim_output	The dimension of output.
Opt_SGD	Whether to use SGD optimizer, 0: not use 1: use.
Opt_Adadelta	Whether to use Adadelta optimizer, 0: not use 1: use.
Opt_Adagrad	Whether to use Adagrad optimizer, 0: not use 1: use.
Opt_Momentum	Whether to use momentum optimizer, 0: not use 1: use.
Opt_Adam	Whether to use Adam optimizer, 0: not use 1: use.
Opt_RMSProp	Whether to use RMSProp optimizer, 0: not use 1: use.
Act_relu	Whether to use ReLu activation, 0: not use 1: use.
Act_tanh	Whether to use tanh activation, 0: not use 1: use.
Act_sigmoid	Whether to use sigmoid activation, 0: not use 1: use.

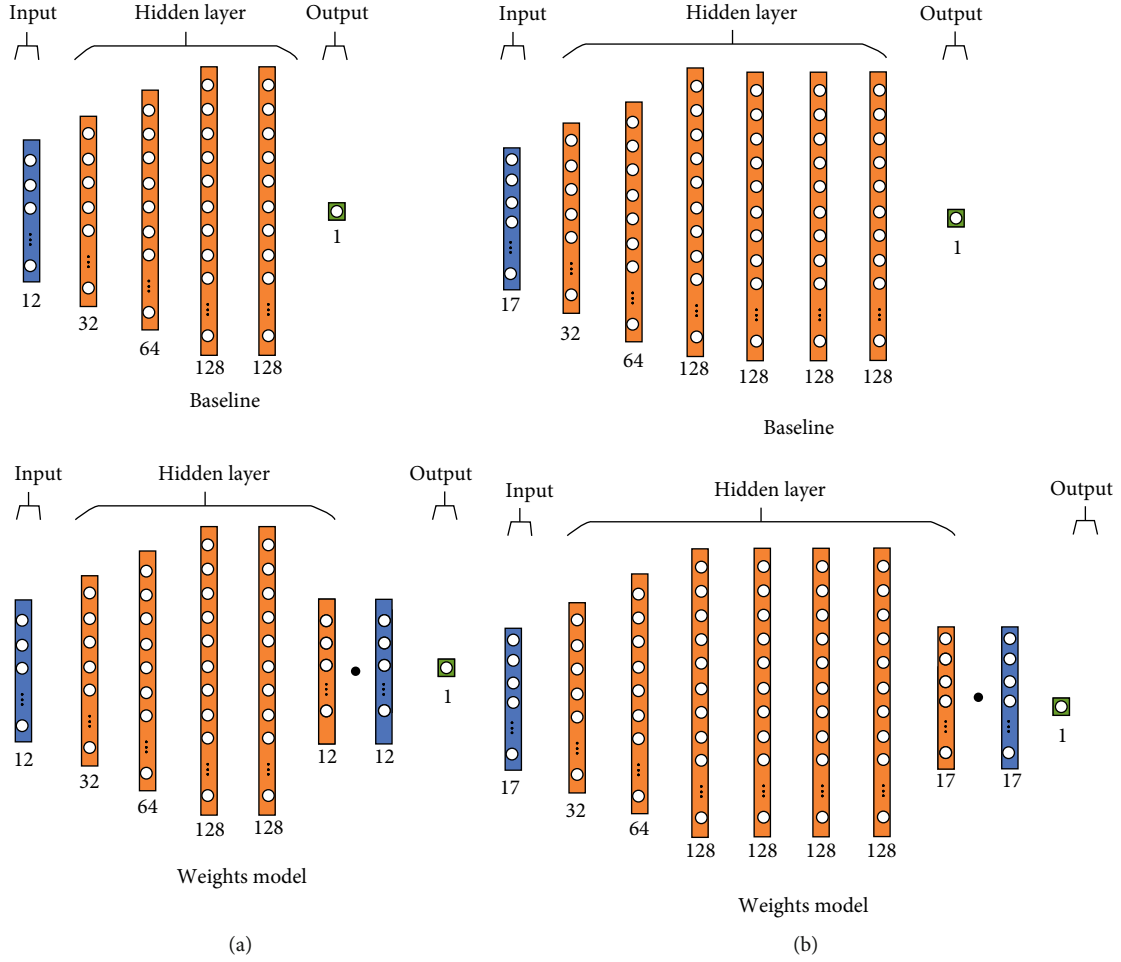


FIGURE 2: (a) The structures of baseline and weights model for single-GPU datasets. (b) The structures of baseline and weights model for stacked datasets.

key features that have a great overall influence on training time. Our analysis method and dimensionality reduction rule will be introduced in detail as follows.

Before analyzing the influence of feature weights, we first need to extract the weights of features by weights model based on multiple datasets described in Section 3.1. Then, we use formula (8) to calculate the ranking of the  $j$ -th feature's weight in weights data  $i$ , where  $w_{i,k}$  represents the weight of the  $k$ -th feature in the weight data  $i$ , and  $n$  represents the total number of features.

$$\text{Rank}(i, j) = 1 + \sum_{k=1}^n \max\left(\frac{||w_{i,k}| - |w_{i,j}||}{|w_{i,k}| - |w_{i,j}|}, 0\right). \quad (8)$$

The standard deviation is an indicator which can reflect the extent of data dispersion. For measuring the fluctuation of the ranking of feature weights, we calculate the standard deviation of weights' ranking, and use  $\text{RankStd}(d, j)$  to represent the standard deviation of the  $j$ -th feature weight's ranking of dataset  $d$ .

Because of the device heterogeneity in federated learning, it is not universal to extract key features based only on the dataset generated by a single device. Therefore, we ana-

lyze the weights of features in many datasets collected from different GPUs. We use formula (9) to calculate the average ranking of feature weights to represent the overall contribution of features on heterogeneous devices and use formula (10) to obtain the average standard deviation of weights ranking for measuring the overall fluctuation extent of the weights ranking. Where  $N$  represents the number of datasets, and  $M_d$  represents the data volume of dataset  $d$ .

$$\text{MeanRank}(j) = \frac{1}{N} \sum_{d=1}^N \frac{1}{M_d} \sum_{i=1}^{M_d} \text{Rank}(i, j), \quad (9)$$

$$\text{MeanRankStd}(j) = \frac{1}{N} \sum_{d=1}^N \text{RankStd}(d, j). \quad (10)$$

MeanRank and MeanRankStd reflect the overall influence of features on training time from the perspective of contribution and fluctuation. In order to reduce the feature dimension, we formulate a unified dimensionality reduction rule according to MeanRank and MeanRankStd for extracting the key features which have a great overall impact on training time. The dimensionality reduction

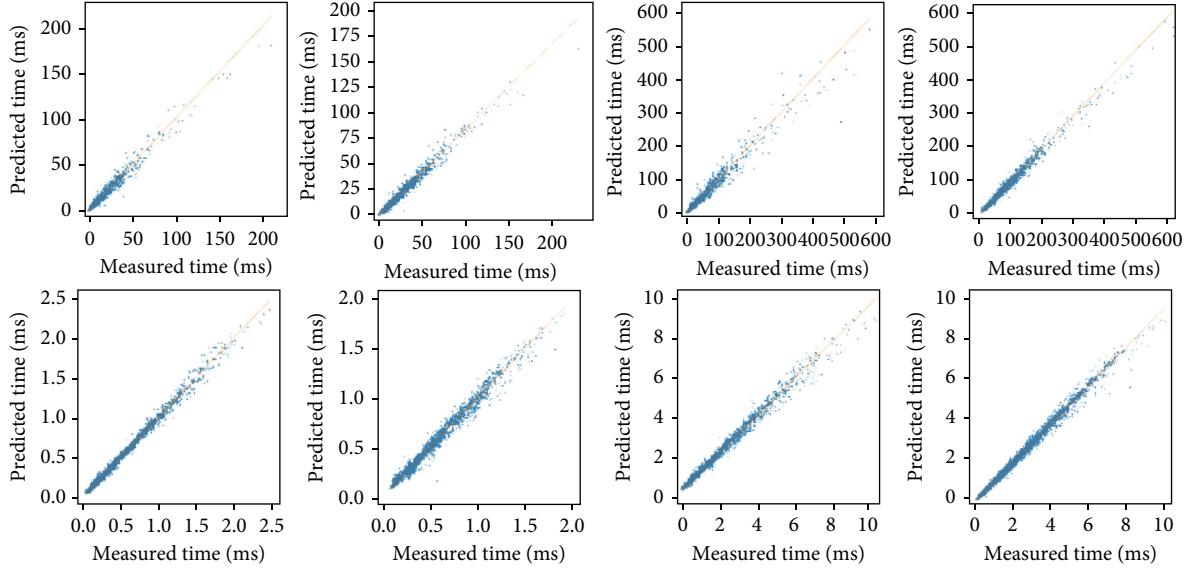


FIGURE 3: Predicted time vs. observed time. The upper of the figure: predicted time vs. observed time on the convolutional feature datasets, from left to right, is the results on *P100\_Conv*, *V100\_Conv*, *K40\_Conv*, and *All\_Conv*; the lower of the figure: predicted time vs. observed time on the dense feature datasets, from left to right, is the results on *P100\_Dense*, *V100\_Dense*, *K40\_Dense*, and *All\_Dense*.

TABLE 6: Weights model vs. linear regression model on *P100*, *V100*, and *K40*.

Datasets	RMSE	
	Weights model	Linear regression model
P100_Conv	3.09 ms	12.59 ms
V100_Conv	1.85 ms	9.46 ms
K40_Conv	11.67 ms	47.41 ms

rule can be expressed as formula (11),  $\Theta$  represents the set of key features,  $F$  is the collection of all features, and  $s$  and  $r$  are constants, whose values need to be set according to the MeanRank and MeanRankStd. Note that the values of  $s$  and  $r$  are set empirically. After many experiments, we found that the effect of dimensionality reduction is the best when  $s$  is set to 1.55 and  $r$  is set to 8 for convolutional layers in this paper. And for dense layers,  $s$  should be set to 2 and  $r$  should be set to 8.

$$\Theta = \{j \mid \text{Mean Rank Std}(j) > s \cup \text{Mean Rank}(j) < r, j \in F\}. \quad (11)$$

The selection of key features by dimensionality reduction rule can be divided into the following two steps:

- (1) Select the features with greater MeanRankStd (greater than  $s$ )

From the metrics of MeanRankStd, the overall stability of weights ranking can be intuitively judged. Features with smaller MeanRankStd have a relatively stable overall influence on training time, while others with greater MeanRankStd usually fluctuate wildly. As for features with

strong ranking fluctuations, we find that they may have a small influence on some input data, but have a large influence on other data. It will cause serious deviations in the prediction of training time without such features. So, we choose to extract features with greater MeanRankStd.

- (2) Select the features with smaller MeanRank (smaller than  $r$ )

After screening in (1), there are some features with stable rankings (smaller MeanRankStd) in the feature collection. These features contain the ones with smaller MeanRank which have a greater overall impact on training time, such as the feature input channels whose influence is the largest for almost every input data. Therefore, we select the features with smaller MeanRank from the rest of the feature collection.

The process of extracting key features using the dimensionality reduction rule can be described by Algorithm 1.

By using the dimensionality reduction rule, the layer features with no or little impact on training time will be eliminated and the high prediction accuracy will be kept. Therefore, the dimension of layer features and redundant training data for the time prediction model will be reduced. Such as the feature dimension and training data are reduced by 30% and 25% for the convolutional layer, and they are both reduced by 20% for the dense layer. We take Justus et al.'s time prediction model as the baseline to verify the accuracy of our dimensionality reduction rule. Then, we retrain the baseline with the dimension-reduced dataset and compare it with the original baseline. The results show that the prediction error of the model trained with the dimension-reduced dataset remains at the same level as the original baseline. See the experimental analysis in Section 4.3 for details.

TABLE 7: Weights model vs. baseline.

Datasets	Model	Test RMSE	Test MAPE	Validation RMSE	Validation MAPE
P100_Conv	Baseline	2.962 ms	14.36%	3.455 ms	15.17%
	Weights model	2.894 ms	15.24%	3.091 ms	14.58%
V100_Conv	Baseline	1.722 ms	11.44%	1.547 ms	11.13%
	Weights model	1.660 ms	11.03%	1.850 ms	11.49%
K40_Conv	Baseline	10.136 ms	15.59%	10.361 ms	16.39%
	Weights model	9.051 ms	15.14%	11.675 ms	15.16%
P100_Dense	Baseline	0.033 ms	3.15%	0.032 ms	3.04%
	Weights model	0.032 ms	3.05%	0.032 ms	3.02%
V100_Dense	Baseline	0.051 ms	6.26%	0.046 ms	5.84%
	Weights model	0.046 ms	6.07%	0.047 ms	6.10%
K40_Dense	Baseline	0.157 ms	7.57%	0.179 ms	7.92%
	Weights model	0.159 ms	7.11%	0.150 ms	7.07%
All_Conv	Baseline	4.079 ms	11.44%	4.021 ms	11.16%
	Weights model	4.273 ms	10.40%	3.893 ms	10.39%
All_Dense	Baseline	0.077 ms	4.70%	0.074 ms	4.68%
	Weights model	0.077 ms	4.72%	0.076 ms	4.73%

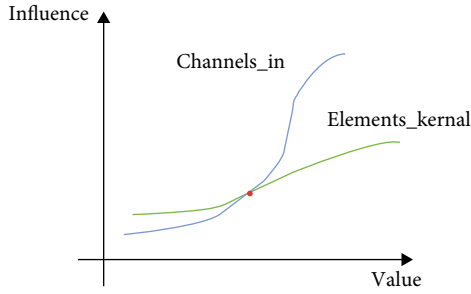


FIGURE 4: Nonlinear influence.

**3.4. Dynamically Setting Number of Local Epochs.** In federated learning, each client only uses its own data for model training. And most of the training data are generated by the client itself. Due to the different characteristics of clients, the distribution of data generated by different client is usually different. Therefore, the training data of federated learning is nonindependent and identically distributed (Non-IID). Unfortunately, Non-IID will cause the divergence between the local model and the global model, resulting in the error convergence of the local model. If the number of local epochs set for each client is different, it will undoubtedly aggravate the divergence of the local model. Therefore, FedNova can eliminate the fast error convergence by normalized averaging the gradients of local models and ensure the convergence of the global model when set different number of local epochs for clients. The update of the global model in FedNova can be described by the following formula:

$$x_g^{(t+1)} - x_g^{(t)} = -\tau_{eff}^{(t)} \sum_{i=1}^m \frac{n_i}{n} \bullet \eta d_i^{(t)}, \text{ where } \tau_i = \left\lfloor \frac{E \bullet n_i}{B} \right\rfloor. \quad (12)$$

Compared with the update of global model in FedAvg (formula (1)), FedNova uses the normalized averaging gradient  $d_i^{(t)}$  to replace the accumulation of the local model's gradient  $\sum_{k=0}^{\tau_i-1} g_i(x_i^{(t,k)})$ , which can restrict the error convergence of local models.  $\tau_{eff}^{(t)}$  in formula (11) can be treated as the learning rate of the global model. It turns out that compared with FedAvg, FedNova can improve the performance of the global model while reducing the number of communications between the clients and the server.

Although we can set different number of local epochs for clients while ensuring good performance of the global model by using FedNova, the problem of the clients with high training speed waiting for the clients with low training speed still exists. This is because FedNova adopts a random sampling method within a given value range to set the number of local epochs, which may set a small number of epochs for the faster clients and set a large number of epochs for the slower clients.

In order to solve the idling problem and improve the training efficiency of global model, we predict the training time of local models in FedNova for guiding the dynamic setting of local epochs. It should be noted that training time prediction can be combined with any algorithm that supports setting different numbers of local epochs for clients. And we choose FedNova for its excellent performance. We use the time prediction model trained with dimension-reduced datasets to predict the training time of one epoch of local models, and then we calculate the number of local epochs for each client according to the time window of communication between the client and the server. There are many ways to combine FedNova and training time prediction. For example, you can choose to predict the training time of local models on the server side and calculate the number of local epochs, or you can choose to send the time prediction model to clients and let clients calculate the number of local epochs by themselves. In actual scenarios, the

way of dynamically setting the number of local epochs needs to be determined according to specific needs. Next, we will introduce the method based on the FedNova and training time prediction algorithm, and the details are described in Algorithm 2.

First, the server randomly selects some clients according to proportion  $C$  and collects their model updates. Second, the server aggregates the updates of the local models to the global model and prepares the time prediction model for clients. Third, the client obtains the global model and time prediction model from the server, divides the local dataset into batches, and extracts the features required for time prediction. Then, the client uses the time prediction model to predict the training time of one epoch of the local model and calculates the number of local epochs according to the preset communication time window  $T$ . Finally, the client uses FedNova to update the local model and send it to the server.

## 4. Experiments

In this section, we arrange experiments to verify the effectiveness of our work, including the weights model and the dimension reduction rule. First, the experimental settings are introduced, including the selection and description of datasets, the setting of model structure, the evaluation metrics, and the introduction of the experimental environment. Second, we use the datasets collected on heterogeneous GPUs to train the weights model for verifying its convergence, and then we conduct a large number of comparative experiments to compare the prediction error between weights model and baseline. We also compare the performance of weights model and linear regression model to prove the superiority of our weights model. Finally, for verifying the effectiveness of the dimensionality reduction rule, we train the baseline based on the complete dataset and the dimension-reduced dataset, respectively, and then compare the error level of prediction between them.

### 4.1. Experiment Settings

**4.1.1. Datasets.** We use the layer features described in Section 3.1 and use 8 different datasets for experiments. There are 6 datasets composed of common features, convolutional features, dense features, and training time collected on different GPUs, which we called single-GPU datasets. The other 2 datasets are the stacking of these 6 datasets, which we called stacked datasets. In order to distinguish different types of GPU, we add three hardware features to the stacked datasets: GPU bandwidth, GPU processing units' number, and GPU clock cycle speed. See Table 3 for the description of the datasets.

The layer features selected in the convolutional layer are shown in Table 4. One-hot encoding is used to represent different optimizers and activation functions for more appropriately measuring the feature distance.

The number of features in the dense layer is less than that in the convolutional layer, and some features are different, such as the dimension of input and the dimension of output. The features selected in the dense layer are shown in Table 5.

TABLE 8: The standard deviation of convolutional feature weights ranking.

Features	P100_ Conv	V100_ Conv	K40_ Conv	MeanRankStd
Batchsize	1.7653	1.3045	1.4026	1.4908
Elements_ matrix	2.3233	2.0775	1.1578	1.8529
Elements_ kernel	1.7904	2.2949	1.4356	1.8403
Channels_in	0.4440	0.4218	0.7599	0.5419
Channels_out	3.0549	2.1915	2.1091	2.4518
Padding	1.6192	1.8036	1.2830	1.5686
Strides	2.4347	2.0012	3.2603	2.5654
Use_bias	1.2580	1.5514	1.3097	1.3730
Opt_SGD	2.3229	3.0940	1.6340	2.3503
Opt_Adadelta	2.2164	2.8144	1.9606	2.3305
Opt_Adagrad	2.7459	2.4789	1.1959	2.1402
Opt_ Momentum	2.6090	2.1952	1.1699	1.9914
Opt_Adam	1.4019	2.1550	2.4311	1.9960
Opt_RMSProp	2.0723	2.3218	0.9707	1.7883
Act_relu	1.5234	1.0455	0.9251	1.1647
Act_tanh	1.7225	1.1762	1.4122	1.4370
Act_sigmoid	1.3042	1.1182	1.1107	1.1777

TABLE 9: The standard deviation of dense feature weight ranking.

Features	P100_ Dense	V100_ Dense	K40_ Dense	MeanRankStd
Batchsize	1.8927	1.4465	1.5556	1.6316
Dim_input	3.3842	2.2099	2.3296	2.6412
Dim_output	2.4491	2.7088	1.7920	2.3166
Opt_SGD	2.8395	2.9443	2.1840	2.6559
Opt_Adadelta	3.3851	2.9903	1.3891	2.5882
Opt_Adagrad	2.0481	3.0305	2.0840	2.3875
Opt_ Momentum	2.8527	2.9168	1.9885	2.5860
Opt_Adam	2.7488	2.9804	2.4706	2.7333
Opt_ RMSProp	3.4788	1.7232	2.0469	2.4163
Act_relu	1.8273	1.4856	1.0920	1.4683
Act_tanh	2.1289	2.2291	0.7319	1.6966
Act_sigmoid	2.2241	2.1797	1.1457	1.8498

**4.1.2. Model Settings.** Compared with the baseline, we only add a weights layer at the end of hidden layers for extracting the weights of features, and the rest of the layers are the same as the baseline. Note that the data volume of the stacked datasets (All\_Conv and All\_Dense) is larger than single-GPU datasets, and three more hardware features are added to distinguish the GPUs. Therefore, for the single-GPU datasets and stacked datasets, the number of weights model's layers are different. Figure 2 shows the settings of the

TABLE 10: The mean ranking of convolutional feature weights.

Features	P100_ Conv	V100_ Conv	K40_ Conv	MeanRank
Batchsize	8.692	9.574	5.702	7.989333333
Elements_ matrix	5.8396	8.6824	7.5608	7.360933333
Elements_ kernel	11.2952	11.7014	10.1	11.0322
Channels_in	1.09	1.0334	1.0112	1.044866667
Channels_out	7.0608	5.6072	12.0992	8.255733333
Padding	14.1344	13.7096	13.9144	13.91946667
Strides	8.9108	9.8826	8.4852	9.092866667
Use_bias	14.5956	14.595	15.7176	14.9694
Opt_SGD	6.3264	6.853	4.2328	5.804066667
Opt_Adadelta	3.764	7.3048	7.0828	6.050533333
Opt_Adagrad	7.732	5.3538	6.3752	6.487
Opt_ Momentum	5.2476	5.3584	11.0724	7.226133333
Opt_Adam	10.1944	3.6814	5.1448	6.3402
Opt_RMSProp	2.9736	4.107	5.9512	4.343933333
Act_relu	15.8232	15.6576	7.3332	12.938
Act_tanh	14.0576	15.8164	15.514	15.12933333
Act_sigmoid	15.2628	14.082	15.7032	15.016

baseline and weights model for the single-GPU datasets and the stacked datasets.

**4.1.3. Metrics.** We use the root mean square error (RMSE) and mean absolute percentage error (MAPE) to measure the error between predicted training time and observed training time. The calculation formula of RMSE is shown in equation (13), and the unit is milliseconds (ms); the calculation formula of MAPE is shown in equation (14).  $Y_{obs,i}$  represents the observed training time of the  $i$ -th input data, and  $Y_{pred,i}$  represents the predicted training time of the model of the  $i$ -th input data.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (Y_{obs,i} - Y_{pred,i})^2}{n}}, \quad (13)$$

$$MAPE = \left(\frac{100}{n}\right) \sum_{i=1}^n \left| \frac{Y_{obs,i} - Y_{pred,i}}{Y_{obs,i}} \right|. \quad (14)$$

**4.1.4. Experimental Environment.** The hardware environment of our experiments is a stand-alone machine, using a CPU for training which contains 6 cores and 12 threads, the main frequency of this CPU is 3.1 GHz, and the memory is 16 G. And we use TensorFlow 1.13.2 framework on 64-bit Windows 10 operating system to finish this program.

## 4.2. Analysis of Weights Model

**4.2.1. Convergence Verification of Weights Model.** In the convergence verification of weights model, we train the weights model using all datasets. Figure 3 shows the comparison of

TABLE 11: The mean ranking of dense feature weights.

Features	P100_ Dense	V100_ Dense	K40_ Dense	MeanRank
Batchsize	10.3196	10.6268	9.5132	10.1532
Dim_input	6.0088	4.9544	3.854	4.939066667
Dim_output	7.1052	4.882	5.9032	5.963466667
Opt_SGD	4.7872	7.2696	3.9624	5.339733333
Opt_Adadelta	4.396	4.2684	2.2148	3.6264
Opt_Adagrad	5.1912	6.3724	7.1868	6.250133333
Opt_ Momentum	8.072	7.9004	3.562	6.511466667
Opt_Adam	5.4128	3.7576	5.1504	4.7736
Opt_ RMSProp	7.35	2.5372	5.1288	5.005333333
Act_relu	6.0432	10.2896	9.9068	8.746533333
Act_tanh	5.6348	7.3096	11.452	8.132133333
Act_sigmoid	7.6792	7.832	10.1656	8.558933333

predicted time and observed time on different datasets. On the *P100\_Conv*, *V100\_Conv*, *K40\_Conv*, and *All\_Conv*, the RMSE of weights model is 2.894 ms, 1.660 ms, 9.051 ms, and 4.273 ms, respectively; on the *P100\_Dense*, *V100\_Dense*, *K40\_Dense*, and *All\_Dense*, the RMSE of weights model is 0.032 ms, 0.046 ms, 0.159 ms, and 0.077 ms, respectively. Therefore, the weights model has good convergence whether based on single-GPU datasets (*P100*, *V100*, and *K40*) or stacked datasets (*All\_Conv* and *All\_Dense*).

**4.2.2. Weights Model vs. Linear Regression Model.** In order to verify the comparative analysis of the weights model and linear regression model in Section 3.2, we use *P100\_Conv*, *V100\_Conv*, and *K40\_Conv* datasets to train the weights model and linear regression model, respectively. Table 6 shows the RMSE comparison between the linear model and weights model. It can be seen that the error of weights model is far lower than that of the linear regression model. On the *P100\_Conv*, *V100\_Conv*, and *K40\_Conv* datasets, the RMSE of the linear regression model is 9.5 ms, 7.61 ms, and 35.74 ms larger than the weighted model, respectively.

**4.2.3. Weights Model vs. Baseline.** Since the training time prediction accuracy of the weights model determines the authenticity of feature weights extracted by weights model, we divide the dataset into the training set, test set, and validation set according to the ratio of 8:1:1 and calculate the test error and the validation error of the weights model to prove the accuracy of weights model. The comparison of test error and verification error between weights model and baseline shows that weights model reaches the same error level as the baseline on *P100*, *V100*, *K40*, and stacked datasets (see Table 7). It is worth mentioning that on the test sets of *K40\_Conv*, *K40\_Dense*, *P100\_Dense*, and *All\_Conv*, the MAPE of weights model is lower than baseline by 0.1%, 0.19%, 0.46%, and 1.04%, respectively, and on the test sets

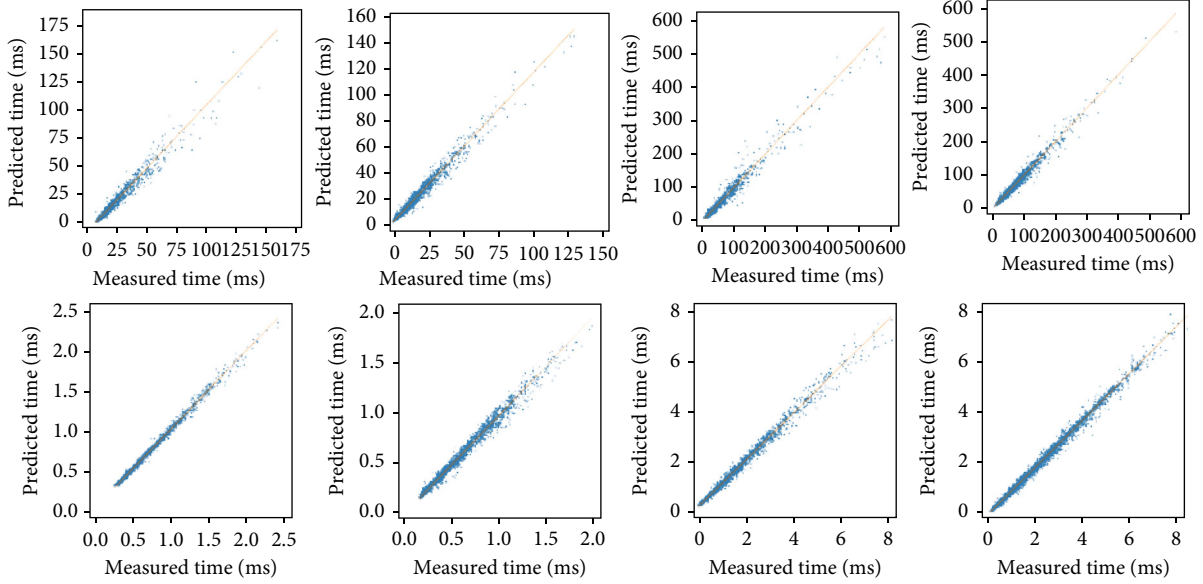


FIGURE 5: Predicted time vs. observed time. The upper of the figure: predicted time vs. observed time on the convolutional feature dimension-reduced datasets, from left to right, is the results on *P100\_Conv\_small*, *V100\_Conv\_small*, *K40\_Conv\_small*, and *All\_Conv\_small*; the lower of the figure: predicted time vs. observed time on the dense feature dimension-reduced datasets, from left to right, is the results on *P100\_Dense\_small*, *V100\_Dense\_small*, *K40\_Dense\_small*, and *All\_Dense\_small*.

and validation sets of *P100\_Conv*, the RMSE of weights model is lower than baseline by 0.068 ms and 0.364 ms.

**4.3. Analysis of Dimensionality Reduction Rule.** In 4.2, we proved that the weights model has good convergence and has a high prediction accuracy that is not lower than baseline. In this section, we will verify the effectiveness of the dimensionality reduction rule. We use the weights model to extract feature weights of different test sets to form the weights datasets, then we use the dimensionality reduction rule described in Section 3.3 to reduce the dimension of features. After that, we compare the prediction error of the model trained based on dimensionality-reduced data with the original baseline.

In our experiments, we found that the ranking of feature weights may fluctuate significantly for different input feature data. We think it is because the influence of layer features on training time does not necessarily change linearly. Assuming the influence of *channels\_in* and *elements\_kernel* on the final training time is shown in Figure 4, when the value of *channels\_in* gradually decreases, its influence on the final result is not necessarily greater than the *elements\_kernel*.

Before using the dimensionality reduction rule, we use formula (8) to calculate the ranking of feature weights in each weights dataset. According to the first step of the dimensionality reduction rule, we need to measure the fluctuation of each feature weight, that is, the standard deviation. Therefore, the standard deviation of weights ranking and the average standard deviation of weights ranking are calculated, see Tables 8 and 9. For the second step of the dimensionality reduction rule, we calculate the average ranking of each feature's weight on all weights datasets and obtain the overall average ranking on all datasets according to formula (9). Tables 10 and 11, respectively, show the aver-

age ranking of feature weights on each dataset, as well as the overall average ranking MeanRank.

Taking the convolutional features as an example, since the optimizers and activation functions are represented by one-hot encoding, we regard all optimizer fields (feature name starting with *opt\_*) as one feature *opt* and all activation function fields (feature name starting with *act\_*) as one feature *act*.

According to step 1 of the dimensionality reduction rule, we select features with MeanRankStd greater than 1.55, the results are *elements\_matrix*, *elements\_kernel*, *channels\_in*, *channels\_out*, *strides*, and *opt* (all optimizer fields). The remaining features with small MeanRankStd are *batchsize*, *channels\_in*, *padding*, *use\_bias*, and *act* (all activation fields).

According to step 2 of the dimensionality reduction rule, we select features with MeanRank less than 8 in *batchsize*, *channels*, *padding*, *use\_bias*, and *act*, and the results are *batchsize* and *channels\_in*.

Finally, by using the dimensionality reduction rule, we get *batchsize*, *channels\_in*, *elements\_matrix*, *elements\_kernel*, *channels\_in*, *channels\_out*, *strides*, and *opt*. The convolutional layer features are reduced by 3 dimensions (*padding*, *use\_bias*, and *act*), and the training data is reduced by 5 dimensions (*padding*, *use\_bias*, *act\_relu*, *act\_tanh*, and *act\_sigmoid*).

For the dense layer features, we use the same method. According to the dimensionality reduction rule, we exclude features that have less influence on training time, including *act\_relu*, *act\_tanh*, *act\_sigmoid*, and *batchsize*. It should be noted that the dense layer has the parameter-intensive characteristic, which means that the transmission of parameters takes a long time. However, the time overhead of parameter transmission was ignored in our datasets. According to the forward propagation process of the neural network, the



TABLE 12: Baseline vs. baseline\_small.

Datasets	Model	Test RMSE	Test MAPE	Validation RMSE	Validation MAPE
P100_Conv	Baseline	2.962 ms	14.36%	3.455 ms	15.17%
	Baseline_small	2.838 ms	15.27%	3.0948 ms	15.47%
V100_Conv	Baseline	1.722 ms	11.44%	1.547 ms	11.13%
	Baseline_small	1.762 ms	11.80%	1.791 ms	11.80%
K40_Conv	Baseline	10.136 ms	15.59%	10.361 ms	16.39%
	Baseline_small	9.882 ms	16.01%	11.508 ms	16.68%
P100_Dense	Baseline	0.033 ms	3.15%	0.032 ms	3.04%
	Baseline_small	0.027 ms	2.86%	0.031 ms	2.92%
V100_Dense	Baseline	0.051 ms	6.26%	0.046 ms	5.84%
	Baseline_small	0.044 ms	5.49%	0.045 ms	5.45%
K40_Dense	Baseline	0.157 ms	7.57%	0.179 ms	7.92%
	Baseline_small	0.154 ms	6.60%	0.140 ms	6.60%
All_Conv	Baseline	4.079 ms	11.44%	4.021 ms	11.16%
	Baseline_small	4.001 ms	11.89%	4.090 ms	12.10%
All_Dense	Baseline	0.077 ms	4.70%	0.074 ms	4.68%
	Baseline_small	0.069 ms	4.70%	0.070 ms	4.75%

dense layer must perform one forward propagation calculation for one input data, and it must perform a batch of forward propagation for a batch of input data. Therefore, the batchsize determines the number of times the parameters are transmitted, which should not be excluded.

After extracting key features by dimensionality reduction rule, we filter the training sets and get the dimension-reduced datasets on *P100\_Conv*, *V100\_Conv*, *K40\_Conv*, *All\_Conv*, *P100\_Dense*, *V100\_Dense*, *K40\_Dense*, and *All\_Dense* (We add *\_small* represents the dimension-reduced datasets. For example, the dimension-reduced dataset of *P100\_Conv* is *P100\_Conv\_small*). For verifying the validity of dimension-reduced datasets, we use the dimension-reduced datasets to train the baseline, and the trained model is called *baseline\_small*. And our experiments have proved that *baseline\_small* also has good convergence. Figure 5 shows the comparison of predicted time and observed time of *baseline\_small* on the test sets.

For determining whether the dimension-reduced datasets caused the loss of prediction accuracy, we calculated the RMSE and MAPE of baseline and *baseline\_small* on all datasets. The results are shown in Table 12. For the convolutional layer datasets, the *baseline\_small* test RMSE is 0.1385 ms lower than the baseline on average, the verification RMSE is 0.3664 ms higher than the baseline on average; for the dense layer datasets, the *baseline\_small* test RMSE is 0.0078 ms lower than the baseline on average, and the verification RMSE is 0.015 ms lower than the baseline. It is worth mentioning that for the dense layer datasets, both RMSE and MAPE of *baseline\_small* are lower than baseline. We consider it is because there are some features in the dense layer that have no contribution to the training time. These features will disturb the data distribution and increase the prediction error of the model.

The experiments show that after reducing the convolutional layer features by 30% and the training data by 25%,

the error level of prediction is still consistent with the original baseline; after reducing the dense layer features by 20% and the training data by 20%, the error level is generally lower than the baseline, which proves the effectiveness of our weights model and dimensionality reduction rule.

## 5. Conclusion

For the problem of setting the number of local epochs for heterogeneous clients in federated learning, we propose a solution of predicting the training time of deep learning tasks on clients to guide the dynamic setting number of local epochs. We design the weights model to extract the weights of features and accurately interpret the relationship between model features and training time. This paper focuses on the combination of weights model and dimensionality reduction rule to extract the key features for reducing the dimension of features and redundant training data required by the time prediction model. The purpose of our work is to improve the feasibility of predicting the training time of deep learning models on heterogeneous clients in federated learning, so as to dynamically set the number of local epochs for clients. Compared with the existing methods, the results of our experiments show that (1) the weights model has good convergence on heterogeneous devices; (2) the predicted training time of weights model reaches the same error level as baseline; (3) the dimensionality reduction rule in this paper can reduce 30% features and 25% redundant data for the convolutional layer and reduce 20% features and 20% redundant data for the dense layer, while maintaining high prediction accuracy.

## Data Availability

Previously reported [Model Features] data were used to support this study and are available at [10.1109/

BigData.2018.8622396]. These prior studies (and datasets) are cited at relevant places within the text as references [8].

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant no. 62072146 and no. 61972358 and the Key Research and Development Program of Zhejiang Province (2019C01059, 2019C03135, and 2019C03134).

## References

- [1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.
- [2] A. Hard, K. Rao, R. Mathews et al., "Federated learning for mobile keyboard prediction," 2018, <https://arxiv.org/abs/1811.03604>.
- [3] "ai.google. Under the hood of the Pixel 2: How AI is supercharging hardware, 2018," 2018, <https://ai.google/stories/ai-in-hardware/>.
- [4] "support.google. Your chats stay private while Messages improves suggestions, 2019," 2019, <http://support.google.com/messages/answer/9327902>.
- [5] Apple, "Private federated learning (NeurIPS 2019 Expo Talk Abstract)," 2019, [https://nips.cc/ExpoConferences/2019/schedule?talk\\_id=40](https://nips.cc/ExpoConferences/2019/schedule?talk_id=40).
- [6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, <https://arxiv.org/abs/1812.06127>.
- [7] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, *Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization*, 2020.
- [8] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the computational cost of deep learning models," in *2018 IEEE International Conference on Big Data (Big data)*, Seattle, WA, USA, December 2018.
- [9] E. R. Edelman, S. M. J. van Kuijk, A. Hamaekers, M. J. M. de Korte, G. G. van Merode, and W. F. F. A. Buhre, "Improving the prediction of total surgical procedure time using linear regression modeling," *Frontiers in Medicine*, vol. 4, p. 85, 2017.
- [10] B. Yu, H. Wang, W. Shan, and B. Yao, "Prediction of bus travel time using random forests based on near neighbors," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 4, pp. 333–350, 2018.
- [11] J. Cheng, G. Li, and X. Chen, "Research on travel time prediction model of freeway based on gradient boosting decision tree," *IEEE Access*, vol. 7, pp. 7466–7480, 2019.
- [12] Q. Hang, E. R. Sparks, and A. Talwalkar, *Paleo: A Performance Model for Deep Neural Networks*, 2016.
- [13] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, *Optimus: An Efficient Dynamic Resource Scheduler for Deep Learning Clusters*, 2018.
- [14] W. Xu, H. Peng, X. Zeng, F. Zhou, X. Tian, and X. Peng, "A hybrid modelling method for time series forecasting based on a linear regression model and deep learning," *Applied Intelligence*, vol. 49, no. 8, pp. 3002–3015, 2019.
- [15] D. Velasco-Montero, J. Fernandez-Berni, R. Carmona-Galan, and A. Rodriguez-Vazquez, "PreVIous: a methodology for prediction of visual inference performance on IoT devices," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9227–9240, 2019.
- [16] N. C. Petersen, F. Rodrigues, and F. C. Pereira, "Multi-output bus travel time prediction with convolutional LSTM neural network," *Expert Systems with Applications*, vol. 120, pp. 426–435, 2019.
- [17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2015, <https://arxiv.org/abs/1409.1556>.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, July 2016.
- [19] R. Adolf, S. Rama, B. Reagen, G.-y. Wei, and D. Brooks, "Fathom: reference workloads for modern deep learning methods," in *2016 IEEE International Symposium on Workload Characterization (IISWC)*, Providence, RI, USA, September 2016.
- [20] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating deep network training by reducing internal covariate shift," *JMLR*, 2015, <https://arxiv.org/abs/1502.03167>.

## Research Article

# An IoT Crossdomain Access Decision-Making Method Based on Federated Learning

Chao Li <sup>1</sup>, Fan Li <sup>1,2</sup>, Zhiqiang Hao,<sup>3</sup> Lihua Yin <sup>1</sup>, Zhe Sun <sup>1</sup> and Chonghua Wang <sup>3</sup>

<sup>1</sup>Cyberspace Institute of Advanced Technology, Guangzhou University, 510700, China

<sup>2</sup>Guangxi Key Laboratory of Cryptography and Information Security, 541004, China

<sup>3</sup>China Industrial Control Systems Cyber Emergency Response Team, China

Correspondence should be addressed to Lihua Yin; [yinlh@gzhu.edu.cn](mailto:yinlh@gzhu.edu.cn)

Received 6 August 2021; Accepted 4 December 2021; Published 27 December 2021

Academic Editor: Lei Chen

Copyright © 2021 Chao Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Crossdomain collaboration allows smart devices work together in different Internet of Things (IoT) domains. Trusted third party-based solutions require to fully understand the access information of the collaboration participants to implement crossdomain access control, which brings privacy risk. In this paper, we propose a federated learning-based crossdomain access decision-making method (FCAD), which builds a crossdomain access decision-making model without sharing privacy information of collaboration participants. Crossdomain access logs are extracted to construct a training dataset. Data enhancement method is used to address the uneven distribution of the dataset. Federated learning and gradient aggregation methods are used to prevent privacy leaks. The experiments on the public dataset show that FCAD obtains a prediction accuracy of 83.6% in the existing crossdomain access system.

## 1. Introduction

Internet of Things (IoT) allows connections of heterogeneous smart devices. More than 25 billion smart devices will be connected through IoT by 2025 [1]. Some IoT service providers support heterogeneous devices collaboration cross their domains. For example, IFTTT provides a platform where users can defined multidevice connection rules [2]. It allows devices in multiple domains to work together by translating user rules to requests in these domains. Suppose that a user defines a rule “if the door is open, then open the house lights.” The smart door lock is provided by Philips, and the smart lights are provided by Samsung. They are managed by different domain, which are the IoT platforms named *Philips Hue* and *SmartThings*. IFTTT sends requests to the two IoT platforms to make the rule effective. The crossdomain collaboration makes some IoT operations be more convenient.

Crossdomain access control is used to prevent unauthorized access in crossdomain collaboration. The “domain” in crossdomain collaboration means domain, and the “crossdomain collaboration” is to describe the collaboration of devices belong to different managers. Existing crossdomain access control methods often rely on a trusted third party (TTP). The TTP verifies the requests legality and makes crossdomain access decisions. For example, the National Health Information Network (NHIN) [3] unites IoT domains of multiple hospitals by providing a trusted third party platform, to form a virtual alliance of medical systems. This alliance guarantees the freedom of information flowing between doctors and patients and implements the crossdomain access control. In the meantime, many companies are providing crossplatform access services, such as *SmartThings* [4] and *Google Home* [5]. The IoT platform makes access decision and translates user rules to requests in different domains, to achieve crossdomain collaboration.

However, the trusted third party-based solutions lack secure access control policies for crossdomain collaboration. Access control policies are the rules which are used to make an *Allow* or *Deny* decision for an access request [6]. Access control policies are mainly configured by experts or generated by policy mining [7, 8]. In policy mining, logs are used as input of policy mining algorithm, to automatically mine access control policies. However, to protect user's privacy, participants are often unwilling to share access logs, which make it difficult for IoT platforms to get the access logs of collaboration participants among different domains. Then, the IoT platform can only use the incomplete information to mine access control policies. This brings many problems like credential leakage, incomplete revocation, and incorrect policy enforcement [9, 10]. The lack of secure crossdomain access control methods puts collaboration participants at risk of being attacked. A security crossdomain collaboration solution is needed that shares no access logs of participants but completes the crossdomain access decision-making.

In this paper, we propose a crossdomain access decision-making method based on federated learning to solve above problems. Federated learning allows users to train their machine learning model locally and then builds a global model by aggregating the shared local model gradients. In the crossdomain collaboration system using federated learning, local access logs will not be shared. Participants use their local access logs to train their own models. Then, gradients of models are exchanged and aggregated in multiple rounds. Finally, we will obtain a global crossdomain access decision-making model to make decisions for crossdomain access requests. Our contributions are as follows:

- (i) We propose a log preprocessing method to address uneven distribution of crossdomain access logs. By using a data enhancement algorithm, the logs are transformed as the input of learning algorithms
- (ii) We propose a federated learning-based crossdomain access decision-making method (FCAD), to build a crossdomain access decision-making model. The model can decide whether to allow or deny the crossdomain access requests without sharing privacy information of collaboration participants
- (iii) We evaluate the effectiveness of FCAD on a public dataset. The experimental results show that FCAD can obtain a prediction accuracy of 83.6% in a crossdomain collaboration system

The rest of this paper is organized as follows. In section 2, we describe related works. The system design is given in Section 3. Section 4 shows the experiments of FCAD. Section 5 summarizes our work.

## 2. Related Works

We divide existing access decision-making methods into policy-based methods and learning-based methods.

*2.1. Policy-Based Methods.* The policy-based methods make access decisions by the access control policies, which are used to describe the system security constraint. Traditional works rely on field expert knowledge. Neumann et al. [11] use role engineering based on professional knowledge to define roles, permissions, constraints, and role hierarchies for the role-based access control (RBAC), but the limited expert knowledge causes the unstable quality of mined policies. Automatic mining access control policies from the existing access control information are the focus of most researches. Iyer et al. [8] propose an attribute-based access control (ABAC) policy generation method, which can extract positive and negative authorization rules from given access control information, and then mine policy entries. Xu et al. [12] propose an ABAC policy mining method based on access logs and attribute information. They convert access logs into user-authority mappings and iteratively obtain a policy set equivalent to the original access control system. Access control policy mining based on algorithm is also a feasible method. Carlos et al. [13] propose UNICORN, which uses the deterministic annealing and mean-field approximation to achieve a universal access control policy mining.

*2.2. Learning-Based Methods.* The powerful effect of deep learning on distinguishing normal and abnormal behaviors makes it widely used in IoT access decision-making [14]. Narouei et al. [15] use natural language processing (NLP) to analyze system documents, which contains security information. Access control content is identified in the documents written with natural language for access decision-making. Mocanu et al. [16] propose a neural network-based ABAC policy mining method. The method adds attribute data to the access log and converts them into a vector, thereby using them as the input of the neural network model. This method can discover the hidden distribution of the data. Karimi et al. [17] propose a policy mining method based on unsupervised learning algorithm, which mines policies from the extracted policy rule pattern. Jabal et al. [18] propose a framework for learning ABAC policies from examples and context information. The framework achieves good results in both real logs and synthetic logs. Xiang et al. [7] propose time changing decision tree to process the existing access logs. The time changing decision tree records and continuously monitors the current access control constraints of the system. When a new access request does not meet the constraints, the administrator will be notified of risks.

These works use the access control information to mine policies or train a learning model, which are designed for the access decision-making in a single system. In the crossdomain collaboration system, the access control information is protected strictly due to the privacy risk, which leads to the loss effectiveness of these methods. A new crossdomain collaboration solution is needed, which can make access decisions without sharing access information of participants. In this paper, we propose a federated learning-based crossdomain access decision-making method (FCAD). The federated learning enables the crossdomain central server to get

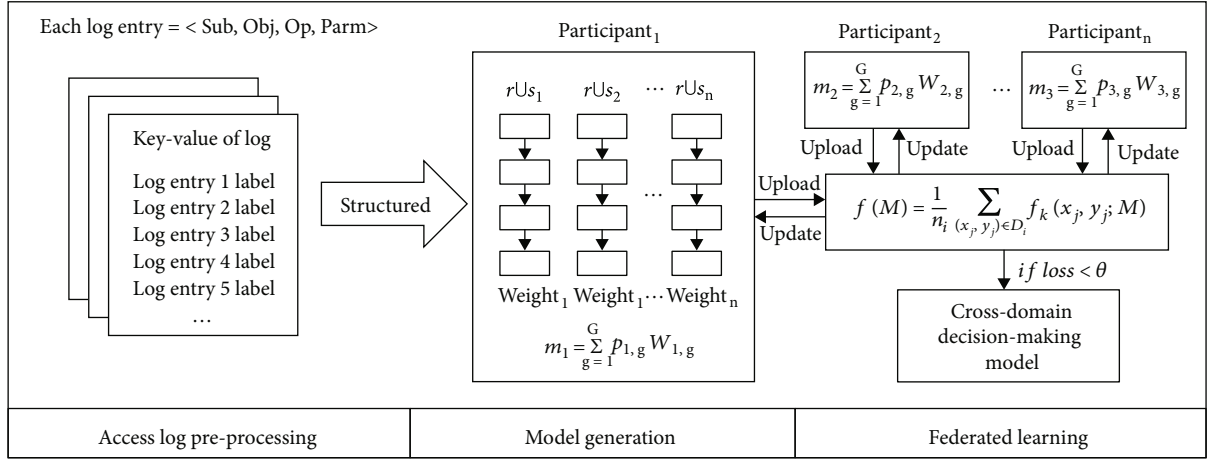


FIGURE 1: The workflow of FCAD.

the decision-making model without obtaining participants logs. This eliminates the privacy risk caused by the shared access information.

### 3. System Design

The workflow of FCAD is shown in Figure 1. In access log preprocessing, the crossdomain access logs are structured to get the access information in a key-value format. The information will be used as the training set of FCAD. Each participant generates its own model and shares the gradients of the model weights. After model generation, the central server collects the gradients from participants and updates the global model. The operation will be repeated until the loss function of the global model is satisfactory.

The system design of FCAD will be introduced in this section, including access log preprocessing, model generation, and federated learning.

**3.1. Access Log Preprocessing.** The crossdomain access logs are generated by different devices and systems, which causes their different formats. We propose an access log preprocessing method to regular the access logs. These logs are mainly generated with the information in crossdomain access requests, which can be defined as a four-tuple  $\langle Subject, Object, Operation, Parameters \rangle$ . *Subject* represents the initiator of the access request. *Object* represents the resource or service being accessed. *Operation* represents the operation such as reading and writing. *Parameters* represents the additional information of the access request, such as the time when the smart door lock is requested to be opened, or the temperature that you want to set when the smart air conditioner is turned on. Logs are generally recorded by natural language in systems. The method of converting natural language logs into key value has been introduced in many works [19]. In FCAD, the crossdomain access logs used for training have been transfer to key-value format by default. A log format example is shown in Figure 2.

*Subject* and *Object* represent the identifications of the requests source and destination. A comprehensive and accurate identity definition enhances the accuracy of the learning

model, since the probability of being attacked and the priority of each participant are different. *Operation* and *Parameters* define an operation together. The security levels of operations are diverse, and the environment context information and command parameters existing in the *Parameters* also affect the decision-making results. Extracting these effective information is necessary to improving the accuracy of the decision-making model.

Most of these valid information are saved by natural language. The extracted information can be expressed as [*Character Types Data, Numerical Data, Label*]. The information in *Character Types Data* is discrete, which cannot be directly used in training. Text vectorization methods are needed in FCAD to transform information into numerical data. On the other hand, the access logs provided by each participant often have different labels. For example, if the participant provides an anomaly detection dataset generated by itself, the data labels may be *DosAttack, ForgeryAttack*, etc. Encoding these labels into numerical values directly will complicate model training and affects the effective of the learning algorithm. To address the problem, we transform the label into  $\langle 0, 1 \rangle$ . 0 and 1 mean that the access request is allowed or denied, so as to simplify the model training.

### 3.2. Model Generation

**3.2.1. Model Training.** We design the FCAD model training process to get a high-accuracy access decision-making model. Based on the features of training dataset, we design a binary classifier based on supervised learning. There are many machine learning models that can achieve good binary classification results, such as random forest and gradient boosting machine [20]. The sequential model is used in FCAD to achieve a model aggregation. Multiple fully connected network structures are stack to complete the classification. The model structure of FCAD is shown in Figure 3. Four fully connected layers are used in FCAD. The activation function of the first three layers is *Relu*, which is the widely used rectified linear units [21]. The *Sigmoid* function is added in the last layer, which decides whether the access request is allowed according to the prediction result. Binary

```

SourceID: Heatingcontrol4
SourceAddress: /agent25/heatingcontrol4
SourceType: /thermostat
SourceLocation: Room_6
DestinationServiceAddress: /agent20/tempin20
DestinationServiceType: /sensorService
DestinationLocation: Room_1
AccessedNodeAddress: /agent20/tempin20
AccessedNodeType: /sensorService
Operation: Read
Parameters: 20
Normality: Normal

```

FIGURE 2: Example of the log format on the used DS2OS dataset.

crossentropy is used as the loss function [22]. The global average pooling [23] can be used to replace the fully connected layer, to reduce the number of parameters and the communication overhead in federated learning.

**3.2.2. Data Enhancement.** In access logs of crossdomain collaboration, the number of allowed requests is much more than denied requests, which will lead to imbalance of the dataset. For example, on the DS2OS dataset, the average malicious access rate of each participant is only 2%. This makes the model trained by participants overfitting easily. To address the problem, we resample the dataset to improve the effectiveness. The rare class samples and part of the abundant class samples on the original dataset are combined as the resampled dataset, and the prediction accuracy is used as the weight for model averaging. The model averaging process can be expressed as follows:

$$\begin{aligned}
 W_{kg} &= M_k(r_k \cup s_{kg}), \\
 m_k &= \sum_{g=1}^G p_{kg} W_{kg},
 \end{aligned} \tag{1}$$

where  $W_{kg}$  is the weight matrix of the  $g$ -th model trained by the  $k$ -th participant,  $M_k$  is the model trained by the  $k$ -th participant,  $r_k$  is the  $k$ -th rare class samples,  $s_{kg}$  is the  $k$ -th abundant class samples selected for the  $g$ -th round,  $m_k$  is the combined weight, and  $p_{kg}$  is the prediction accuracy of  $M_{kg}$ . In FCAD, for a dataset with 2% of malicious access, we randomly generate 50 resampled datasets. The 50 models obtained will be aggregated with their prediction accuracy weights to eliminate the imbalance of the original dataset.

**3.3. Federated Learning.** Although an access decision-making model for a single participant can be obtained, it brings privacy risk in crossdomain access decision-making. We use federated learning solve this problem.

**3.3.1. Workflow.** The workflow of federated learning in FCAD is as following. First, the central server counts the

templates of the local datasets of each participant and performs parameter division and data alignment according to the format of  $\langle \text{Subject}, \text{Object}, \text{Operation}, \text{Parameters} \rangle$ . The learning model will be selected according to the result. Each participant performs data preprocessing on its local dataset and trains model. The data enhancement is independently implemented by participants. Then, the participant encrypts the model weight gradients and transmits it to the central server. Like an adaptive process [24], the central server obtains the updated model parameters by using the gradient aggregation scheme, broadcasts the updated model gradient to the newly selected participant, and iterates above operations until the loss function of the model is satisfactory.

**3.3.2. Gradient Aggregation.** The weights of local models are determined by the participant's data size and the prediction accuracy. We use the impact factor  $\delta$  to measure the contribution of each participant's model, which can be expressed as

$$\begin{aligned}
 \delta_k &= \frac{(n_k/n)P_k}{\sum_{k=1}^K (n_k/n)P_k} = \frac{n_k P_k}{\sum_{k=1}^K n_k P_k}, \\
 \omega_{t+1} &= \sum_{k=1}^K \delta_k \omega_{t,k},
 \end{aligned} \tag{2}$$

where  $\delta_k$  is the model contribution of the  $k$ -th participant, and  $n_k$  is the size of the  $k$ -th local dataset. We obtain the model contribution by calculating the contribution of the dataset and the prediction accuracy  $P_k$ . The model contribution is used to measure the importance of the participant. It directly affects the update of the global model  $\sum_{k=1}^K \delta_k \omega_{t,k} \rightarrow \omega_{t+1}$ . The global model continues to iterate until it satisfies the iteration termination condition, which can be expressed as

$$\text{loss}(o_t, L_t) = -\frac{1}{n} \sum_i (L_t[i] * \log(o_t[i]) + (1 - L_t[i]) * \log(1 - o_t[i])) < \theta. \tag{3}$$

Among them,  $\text{loss}(o_t, L_t)$  is the loss function of the global model.  $L_t[i]$  is the prediction result of the  $i$ -th label of the  $t$ -th iteration.  $o_t[i]$  is the  $i$ -th input of the  $t$ -th iteration, which is the predicted value in  $[0,1]$ . We set  $\text{loss}(o_t, L_t) < \theta$  as the termination condition, which represents that the model is considered to have converged.

**3.3.3. Privacy and Security.** It is worth to discuss privacy and security problems in federated learning-based methods [25, 26]. To prevent problems such as model leakage, data poisoning, and sample exposure, methods such as homomorphic encryption can be used to ensure the safety of learning and data exchanging [27]. Although there are many methods to ensure the data security, side-channel attacks are difficult to defend [28]. For example, we can predict whether the house access logs have been changed by comparing the new model with the old one, so as to predict the activity time of the house owner. In FCAD, the aggregated model

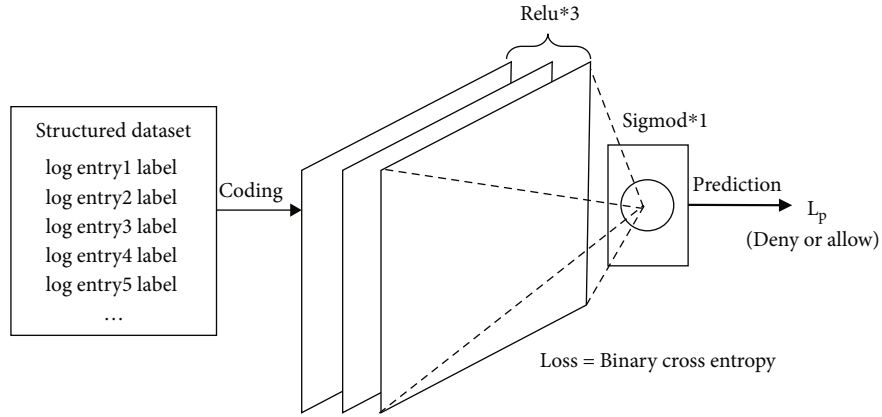


FIGURE 3: The model generation of FCAD.

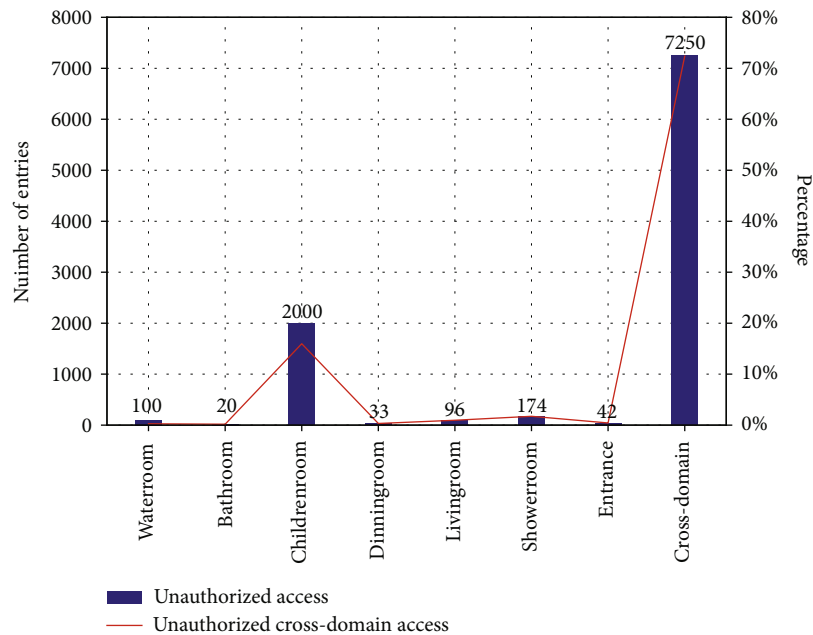


FIGURE 4: Part of abnormal crossdomain access percentage in the DS2OS dataset.

returned to participant changes almost every time, which is used as the old model in the next training round. Since the old model changes, the new participant model will also change, even if the same local access logs are used for training. This prevents the performing of side-channel attacks on FCAD.

#### 4. Experimental Evaluation

We evaluate the prediction accuracy of FCAD in a public dataset to verify the effectiveness.

**4.1. Dataset.** We perform our experiment with the DS2OS traffic traces dataset [29]. The dataset contains access logs obtained in the IoT environment DS2OS. The crossdomain information are generated in application layers from four different simulated IoT sites. They provide different services, such as light controller, thermometer, movement sensors,

washing machines, batteries, thermostats, smart doors, and smart phones. All the devices are implemented in 21 different locations. Each location is regarded as an independent domain. We count the percentage of crossdomain abnormal accesses on the DS2OS dataset, and the results are shown in Figure 4. Among the multiple domains existing in a smart home IoT environment, the percentage of crossdomain abnormal access is 72.4%. In the meantime, more than 7,000 abnormal crossdomain accesses have been made among the 350,000 access entries. The result shows that the DS2OS dataset is close to the real IoT crossdomain collaboration environment.

**4.2. Implementation.** After we extract the log key of DS2OS, the access log entries can be expressed as

$\langle \text{Subject} \rangle: \{ \text{SourceAddress}, \text{SourceType}, \text{SourceLocation} \}.$

$\langle \text{Object} \rangle: \{ \text{SestinationServiceAddress}, \text{DestinationServiceType}, \text{DestinationLocation} \}.$

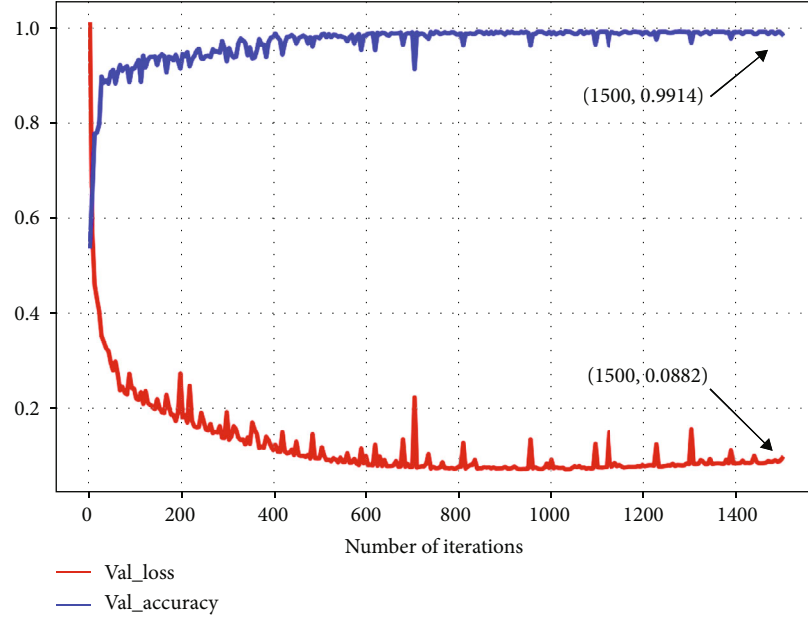


FIGURE 5: Model parameters of FCAD with the complete DS2OS dataset as the training set.

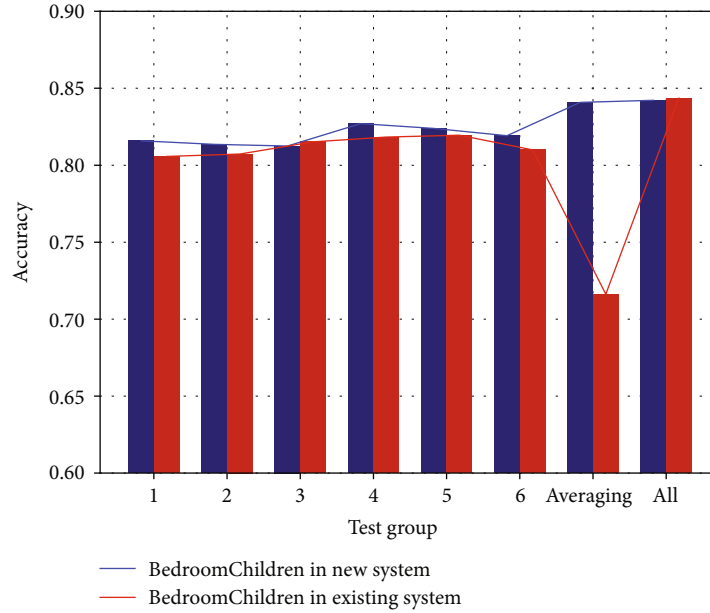


FIGURE 6: Different group accuracies of BedroomChildren local dataset.

TABLE 1: All the chosen proportions in new system and existing system.

Locations	Existing system	New system
Waterroom	3.155%	3.614%
Room_9	3.015%	7.517%
Bathroom	20.495%	0.723%
BedroomChildren	21.703%	72.280%
Entrance	9.334%	1.518%
Showroom	19.856%	6.288%
Total	77.56%	85.65%

$\langle Operation \rangle: \{Operation\}$ .

$\langle Parameters \rangle: \{Value, Timestamp\}$ .

Since DS2OS has already identified the normality of access, we set the label of the normal access logs to 1, and the abnormal access logs to 0. We assume that all the normal access should be allowed, and all the abnormal access should be denied. The different locations are considered to be different domains and finally get 21 domains. We use two methods to obtain the local dataset of each domain. (i) Count entries with the same  $\{DestinationLocation\}$  as the local dataset and (ii) count the access logs within each domain as the local dataset (when



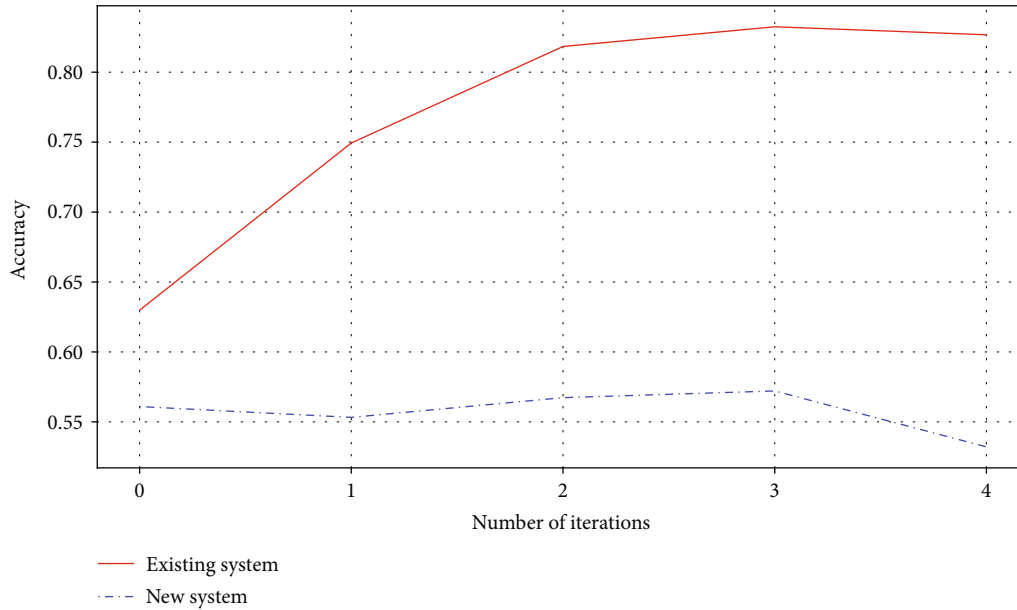


FIGURE 7: Effectiveness of gradient aggregation in FCAD.

$\{SourceLocation\} = \{DestinationLocation\}$ ). The two methods are designed to simulate two actual scenarios: the existing crossdomain collaboration system (which has crossdomain access logs) and the new crossdomain collaboration system (which has no crossdomain access logs). The first method is suitable for long-running crossdomain systems, since there are already a sufficient number of crossdomain access logs as the training set. The second method is suitable for newly constructed crossdomain systems, because there is no crossdomain access request as the training set. We use 30% of entries on the local dataset as the testing set. The dataset preprocessing algorithm is written in Python 3.5. The model training is completed by a laptop computer with a 1.60 Hz CPU (Intel i5-8250U) and 8GB RAM.

#### 4.3. Experiments and Evaluation

**4.3.1. Baseline.** We use all the entries in our dataset to obtain the model training effect as our baseline. The result is shown in Figure 5. We find that the model converged when it iterates 1500 times, and it obtains a prediction accuracy of 99.14%. It shows the effectiveness of the neural network used by FCAD in predicting crossdomain access decisions.

**4.3.2. Model Averaging Effectiveness.** We choose *Bedroom-Children* as an example to evaluate the model averaging effectiveness. The obtained local dataset in existing system has 12768 entries, including 10594 positive samples and 2174 negative samples. The total proportion of negative examples is 17.02%. There are a total of 12,524 entries in the new system dataset, which has 10,524 positive samples and 2000 negative samples. The total proportion of negative examples is 15.96%. We generate 6 resampled datasets and use the same neural network for training. The result is shown in Figure 6. The numbers 1~6 represent the 6 resampled datasets, *Averaging* is the accuracy of the model,

and *All* is the accuracy of the model trained on the original dataset without resampling.

We find that the model trained by single participant is far less effective than the model trained on the entire dataset. Although we have selected a participant with an even distribution, the model is still overfitting. The prediction accuracy is about 80%. Although the prediction accuracy of *All* is the highest, it is close to the original sample distribution. It shows that the model overfitting of *All* may be serious. For the existing system, the prediction accuracy is higher than that of single resampling data, because crossdomain access logs exist on the local dataset. It shows that the data enhancement method is effective to deal with uneven distribution. For the new system, it is difficult to get a good result based on intra-domain access logs; so, the prediction accuracy of the averaged model is greatly reduced.

**4.3.3. Gradient Aggregation Effectiveness.** We choose 6 participants with more negative samples for gradient aggregation. The result is shown in Table 1. For the existing system, the negative samples owned by these 6 participants accounted for 77.56% of all the negative samples. For the new system, it accounted for 85.65%. It is due to a more balanced occurrence of crossdomain abnormal access. We use the proportion of negative samples and the accuracy as the weight of the gradient aggregation.

The prediction accuracy of the access decision-making model after gradient aggregation is evaluated. The result is shown in Figure 7. For the existing system, the accuracy of the model will increase with the number of iterations, which can reach 83.6%. For the new system, the prediction accuracy is about 55%. The reason is that due to the lack of crossdomain access samples in the training dataset, the aggregated model is difficult to predict the decision-making results and almost loses its effect. It shows that FDAC is

more suitable for the existing crossdomain collaboration system.

## 5. Conclusion

In this paper, we propose a federated learning-based crossdomain access decision-making method FCAD. The designed log preprocessing method structures the crossdomain access logs to obtain the training dataset. Data enhancement method is used to address the distribution heterogeneity of the dataset which can be directly used in learning algorithms. Federated learning is used to prevent access logs sharing in crossdomain access decision model establishing. FCAD can obtain information highly relevant to the access decision from access logs and has a prediction accuracy of 83.6% in the existing crossdomain access system. The results show that making an access decision based on the access information obtained by the application layer is a feasible method in crossdomain collaboration. Since the decision made by the learning model can hardly be explained, and it is difficult to deal with updates of the access control policies, FCAD is more likely to be a supplement when access control policies are lacking.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by National Key R&D Program of China (No. 2018YFB2100400), National Science Foundation of China (Nos. 61872100, 62002077), Industrial Internet Innovation and Development Project of China (2019), State Grid Corporation of China Co., Ltd., Technology Project (No. 5700-202019187A-0-0-00), Guangxi Key Laboratory of Cryptography and Information Security (No. GXIS202119), Guangdong Basic and Applied Basic Research Foundation (No. 2020A1515110385), Zhejiang Lab (No. 2020NF0AB01), and Guangzhou Science and Technology Plan Project (No. 202102010440).

## References

- [1] C. Patsioura, *Blockchain and Distributed Ledger Technologies: what's the Value for IoT*, Technical Report. GSMA Intelligence, 2018.
- [2] "IFTTT helps every thing work better together," May 2021, <https://ifttt.com/>.
- [3] T. H. Payne, D. E. Detmer, J. C. Wyatt, and I. E. Buchan, "National-scale clinical information exchange in the United Kingdom: lessons for the United States," *Journal of the American Medical Informatics Association*, vol. 18, no. 1, pp. 91–98, 2011.
- [4] "One simple home system. A world of possibilities. | Smart-Things," May 2021, <https://www.smarthings.com/>.
- [5] "Google Developers," May 2021, <https://developers.google.com/>.
- [6] S. Ravidas, A. Lekidis, F. Paci, and N. Zannone, "Access control in internet-of-things: a survey," *Journal of Network and Computer Applications*, vol. 144, pp. 79–101, 2019.
- [7] C. Xiang, Y. Wu, B. Shen et al., "Towards continuous access control validation and forensics," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 113–129, London, United Kingdom, 2019.
- [8] P. Iyer and A. Masoumzadeh, "Mining positive and negative attribute-based access control policy rules," in *Proceedings of the 23rd ACM on Symposium on Access Control Models and Technologies*, pp. 161–172, Indianapolis, Indiana, USA, 2018.
- [9] B. Yuan, Y. Jia, L. Xing, D. Zhao, X. Wang, and Y. Zhang, "Shattered chain of trust: understanding security risks in cross-cloud IoT access delegation," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 1183–1200, Washington, D. C., 2020.
- [10] Q. Li, B. Xia, H. Huang, Y. Zhang, and T. Zhang, "TRAC: traceable and revocable access control scheme for mHealth in 5G-enabled IIoT," *IEEE Transactions on Industrial Informatics*, p. 1, 2021.
- [11] G. Neumann and M. Strembeck, "A scenario-driven role engineering process for functional RBAC roles," in *Proceedings of the seventh ACM symposium on Access control models and technologies*, pp. 33–42, Monterey, California, USA, 2002.
- [12] Z. Xu and S. D. Stoller, "Mining attribute-based access control policies from logs," in *Data and Applications Security and Privacy XXVIII: IFIP Annual Conference on Data and Applications Security and Privacy*, Lecture Notes in Computer Science, pp. 276–291, Springer Link, 2014.
- [13] C. Cotrini, L. Corinzia, T. Weghorn, and D. Basin, "The next 700 policy miners: a universal method for building policy miners," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 95–112, London, United Kingdom, 2019.
- [14] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020.
- [15] M. Narouei, H. Khanpour, and H. Takabi, "Identification of access control policy sentences from natural language policy documents," in *Data and Applications Security and Privacy XXXI: IFIP Annual Conference on Data and Applications Security and Privacy*, Lecture Notes in Computer Science, pp. 82–100, Springer Link, 2017.
- [16] D. Mocanu, F. Turkmen, and A. Liotta, *Towards ABAC Policy Mining from Logs with Deep Learning*, University of Groningen, 2015.
- [17] L. Karimi and J. Joshi, "An unsupervised learning based approach for mining attribute based access control policies," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1427–1436, Seattle, WA, USA, 2018.
- [18] A. A. Jabal, E. Bertino, J. Lobo et al., "Polisma—a framework for learning attribute-based access control policies," in *Computer Security – ESORICS 2020: European Symposium on Research in Computer Security*, Lecture Notes in Computer Science, pp. 523–544, Springer Link, 2020.

- [19] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, Dallas, Texas, USA, 2017.
- [20] J. H. Min and C. Jeong, "A binary classification method for bankruptcy prediction," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5256–5263, 2009.
- [21] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2018, <https://arxiv.org/abs/1803.08375>.
- [22] U. Ruby and V. Yendapalli, "Binary cross entropy with deep learning technique for image classification," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 5393–5397, 2020.
- [23] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, <https://arxiv.org/abs/1312.4400>.
- [24] P. Zhu, Q. Zhi, Y. Guo, and Z. Wang, "Analysis of epidemic spreading process in adaptive networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 7, pp. 1252–1256, 2018.
- [25] J. Xiong, R. Bi, Y. Tian, X. Liu, and D. Wu, "Towards light-weight, privacy-preserving cooperative object classification for connected autonomous vehicles," *IEEE Internet of Things Journal*, p. 1, 2021.
- [26] J. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2019.
- [27] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batch-crypt: efficient homomorphic encryption for cross-silo federated learning," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, pp. 493–506, Boston, 2020.
- [28] F. Mo and H. Haddadi, *Efficient and Private Federated Learning Using Tee*, Proc. EuroSys Conf., Dresden, Germany, 2019.
- [29] "DS2OS traffic traces," 2021, <https://kaggle.com/francoisxa/ds2ostraffictaces>.

## Research Article

# Multidomain Fusion Data Privacy Security Framework

Jing Yang, Lianwei Qu , and Yong Wang 

*College of Computer Science and Technology, Harbin Engineering University, Harbin, Heilongjiang, China*

Correspondence should be addressed to Yong Wang; wangyongcs@hrbeu.edu.cn

Received 30 July 2021; Accepted 17 November 2021; Published 20 December 2021

Academic Editor: James Ying

Copyright © 2021 Jing Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the collaborative collection of the Internet of Things (IoT) in multidomain, the collected data contains richer background knowledge. However, this puts forward new requirements for the security of data publishing. Furthermore, traditional statistical methods ignore the attributes sensitivity and the relationship between attributes, which makes multimodal statistics among attributes in multidomain fusion data set based on sensitivity difficult. To solve the above problems, this paper proposes a multidomain fusion data privacy security framework. First, based on attributes recognition, classification, and grading model, determine the attributes sensitivity and relationship between attributes to realize the multimode data statistics. Second, combine them with the different modal histograms to build multimodal histograms. Finally, we propose a privacy protection model to ensure the security of data publishing. The experimental analysis shows that the framework can not only build multimodal histograms of different microdomain attribute sets but also effectively reduce frequency query error.

## 1. Introduction

The Internet of Things (IoT) is widely used in data collection in various fields [1–3] and integrates them to apply data analysis in different fields. Therefore, in the development of 5G, 6G wireless networks, and IoT [4, 5], a lightweight, reliable, and intelligent security privacy protection framework is extremely important [6, 7]. The multidomain fusion data set collected by IoT contains strong background knowledge, and it leads to the higher risk of privacy leakage when data publishing [8, 9]. The issue of personal privacy leakage causes security risk to many aspects such as personal life, property, and family. Early research focuses on the improvement of the privacy protection model based on K-anonymity [10–14] and I-diversity [15, 16], but these improved models cannot entirely resist the strong knowledge attacks and new attack methods. Differential privacy proposed is effectively solving privacy leakage caused by background knowledge and is widely used in the domain of data publishing. Although the paper [16–20] fully improved the histogram data publishing algorithm based on differential privacy, the data availability after privacy protection is not better when publishing multimodal histograms. The paper [21] proposes a personalized privacy protection model that supports the

privacy protection needs of publishing histograms, but it is still challenging to balance data privacy and data availability.

Histogram data publishing based on data records is widely used in the domain of data publishing [22]. For example, in the medical data set, we count the number of people suffering from heart disease in different age groups. When facing the multidomain fusion data set, although the histogram data publishing based on data records can provide good data, it is difficult to realize multimode data statistics due to ignoring the attributes sensitivity and relationship between attributes. Therefore, data publishing based on data records cannot publish the relationship between attributes and attribute sensitivity in the multidomain fusion data set. At present, the research on data attributes recognition, classification, grading, and sensitivity focuses on data mining analysis. The domain of data publishing concerns the research of data privacy protection but ignoring the study of data attributes sensitivity and relationship between attributes.

To solve the two problems proposed above, this paper presents a multidomain fusion statistical data publishing privacy protection framework. For the multimode data statistics, we offer two definitions that are microdomain and microdomain attribute set. It applies to the microdomain

recognition of the multidomain personal privacy fusion data set, and the microdomain attribute set is obtained. Afterwards, we take the information gain and attribute sensitivity to implement the classification and grading for microdomain attribute set. Finally, we build the unattributed histogram and universal histogram for microdomain attribute set. For the data publishing personalized privacy protection problem, this paper combined the constraint inference algorithm with grouping reconstruction algorithm to solve personalized privacy protection [23] for the multimodal histogram. Therefore, the contribution of this paper is as follows:

- (1) We propose the multidomain fusion data privacy security framework to solve the multimode data statistics difficult and data publishing privacy security problem.
- (2) We determine the attributes sensitivity and relationship between attributes, through the attributes recognition, classification, and grading model. On this basis, we realize multimode data statistics for multidomain fusion data sets and combine them with the multimodal histogram building model to build multimodal histograms, by the multimodal histogram data publishing privacy protection model to realize the privacy protection of published unattributed histogram and universal histogram.
- (3) The multidomain fusion data privacy security framework can not only build multimodal histograms but also improve the availability of data in long-range queries and small privacy budgets while ensuring privacy security.

In this paper, Section 2 discusses the related work for attributes recognition, classification, grading, and differential privacy. In the following section, we introduce the study of information theory and differential privacy. Section 4 introduces the multimodal histogram data publishing framework. In Section 5, we analyze the experimental results. The last section is the conclusions and future work of this paper.

## 2. Relate Work

*2.1. Data Privacy Security Framework.* With the increasingly serious problem of privacy leakage, a friendly data privacy security framework is a prerequisite to ensure data sharing, publishing, and mining. The core of the data privacy security framework is the targeted privacy protection methods.

Xiong et al. [6] proposed a data privacy security framework. First, the P-CNN model is implemented based on two ciphertexts, and then, P-CNN is deployed with two edge servers to collaboratively solve the problem of original data sharing among connected and autonomous vehicle information leakage. Xiong et al. [7] proposed a personalized privacy protection framework based on game theory and data encryption. This framework solves the problem of QoCS protection imbalance in MCS caused by the unified privacy protection of sensor data. Wu et al. [24] proposed a PETA

framework that not only considers the privacy protection of MCS but also uses powerful edge servers deployed between users and platforms to cluster and manage users based on user attributes. Xiong et al. [12] proposed an ATG framework based on the MCS privacy protection framework, which uses AI technology and game theory to solve the problem of public data privacy leakage in MECS.

At present, the data security frameworks are all based on encryption algorithms. This paper exploits differential privacy algorithms to achieve data publishing privacy protection. The purpose is to improve data availability while ensuring that privacy is not leaked.

*2.2. Attributes Recognition, Classification, and Grading.* Attributes recognition, classification, and grading are important support for data mining, analysis, sharing, and other applications based on data publishing. In contrast to the domain of data publishing, there are relatively few studies on attributes recognition, classification, and grading [25]. The attribute privacy measurement is the key to classification, grading, and privacy protection. The information entropy is one of the effective methods to calculate the amount of information. Therefore, Diaz et al. [26] and Serjantov and Danezis [27] proposed earlier fusion information entropy and other relevant information theory into attribute measurement.

Peng et al. [28] proposed some information entropy privacy protection models based on the Shannon information theory communication framework and solved the problem of privacy measurement from attribute characteristics, background knowledge, and multiple data sources. However, these models ignore the sensitivity between attributes. Yu et al. [29] used the Shannon information theory to measure privacy data and combined it with the BP neural networks to implement privacy data grading. Still this computational cost of this model is larger, and the grading result of BP neural network depends on samples. Krishnamurthy and Wills [30] calculate the privacy leakage amount of the social network attributes to determine the scope of attribute privacy leakage and propose a privacy protection method, but this paper does not consider attributes recognition, classification, and grading. He and Pen [25] put forward a sensitive attribute classification and grading algorithm for structured data set. First, calculate the privacy attribute sensitivity by clustering information entropy and association rule. Second, calculate the attribute sensitivity average to implement attributes classification and grading. However, the clustering result based on  $K$ -means depends on the choice of  $K$  value. If the selected  $K$  value is not appropriate, it will form a local optimal solution leading to inaccurate classification.

The idea of this paper is inspired by the paper [25]. We implement the microdomain recognition of the data set by the proposed definitions for microdomain and microdomain attribute set. Then, we adopt the information gain and attribute sensitivity to represent attributes classification and grading.

*2.3. Data Publishing Privacy Protection.* Data publishing privacy protection is aimed at protecting privacy information

security in the process of data publishing [31]. Though the traditional access control and encryption technology have a better privacy protection effect, the availability of the protected data is insufficient, which defeats the purpose of data publishing. Differential privacy is widely used because it can solve the problem of privacy protection models such as K-anonymity [10–14], L-diversity [15, 16], and T-tight [32], which cannot resist the privacy leakage caused by strong background knowledge.

Dwork [33, 34] proposed a differential privacy method based on Laplace and exponential mechanism in 2006 and 2008. Though this method effectively resists the privacy leakage caused by background knowledge, the error between protected data and original data is large. Dwork et al. [35] proposed an equal-width histogram privacy protection method based on differential privacy that is called LP algorithm, and the performance is better in small noise and small range query. However, when the noise is larger or long-range query, too much noise accumulation leads to poor data availability. To improve the accuracy of long-range query, Xu et al. [36, 37] proposed a differential privacy protection method based on Noise First and Structure First, as well as used the idea of V-optimization histogram to optimize the noise histogram and obtain the high accuracy query results. Whatever, this method cannot balance the noise error and reconstruction error, meanwhile the high computation cost of the postoptimization process. Xiao et al. [38] transformed the original histogram into a binary tree of wavelet coefficient to support long-range query. However, this algorithm is not conducive to the practical application due to the high sensitivity of query. Hay et al. [39] proposed a personalized privacy protection method based on constraint inference for unattributed histogram and universal histogram. The histogram with noise is optimized by constraint inference, but the data accuracy does not have an advantage in low noise and small-range query. Piao et al. [18] proposed the MDHP algorithm to implement privacy protection of governmental data publishing. The MDHP algorithm combined the LP algorithm with the grouping reconstruction algorithm based on the maximum difference scaling to achieve privacy protection which satisfies  $\epsilon$ -differential privacy. The MDHP algorithm effectively improved the published data availability for small-range query and low noise. However, the performance of data availability was poor in the large noise or long-range query.

This paper is inspired by the paper [18, 39]. We improve the MDHP algorithm by the order inference and linear estimation. The algorithm in this paper not only satisfies the privacy requirement of the published multimodal histogram but also effectively balances the privacy and availability of the published data.

### 3. Preliminaries

#### 3.1. Information Entropy

*Definition 1 Information entropy* [25]. The information entropy is the self-information expected value for each dis-

crete event and is denoted by  $H(X)$ .

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i). \quad (1)$$

$p(x_i)$  is probability when  $X$  is equal to  $x_i$ .

*Definition 2 Maximum entropy* [25]. The maximum entropy is when the probability of each discrete event  $x_i$  is equal. The maximum entropy formula is as follows.  $n$  is the number of discrete events.

$$H_{\max}(X) = \log n. \quad (2)$$

*Definition 3 Conditional entropy* [25]. When we set the condition  $X$  and the uncertainty of the random variable  $Y$ ,  $p(x, y)$  is joint distribution  $n$  probability, and  $p(y | x)$  is conditional probability.

$$H_n(Y | X = x) = - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(y | x). \quad (3)$$

*Definition 4 Information gain*. The information gain is when we give the condition  $X$ , calculating the reduction in the uncertainty of  $Y$ . The information gain formula is as follows.  $H(Y)$  is information entropy, and  $H(Y | X)$  is conditional entropy.

$$IG_i = H(Y) - H(Y | X). \quad (4)$$

#### 3.2. Differential Privacy

*Definition 5  $\epsilon$ -Differential privacy* [30]. A randomized mechanism  $S$  is differentially private for any pair of neighboring data set  $I$  and  $I'$  and for any set of possible sanitized output  $R$ .

$$\Pr [S(I) \in R] \leq \exp(\epsilon) \times \Pr [S(I') \in R]. \quad (5)$$

The privacy budget is denoted by  $\epsilon$  which used to represent the privacy level. The smaller  $\epsilon$  represents the higher privacy protection level. The privacy level is inversely proportional to privacy budget  $\epsilon(\epsilon \in [0, 1])$ .

*Definition 6 Global sensitivity* [30]. Given a random query function  $f : D \rightarrow R^d$ ,  $D_1$  and  $D_2$  are neighbor data sets at most one different data record. The global sensitivity formula is as follows:

$$\Delta f = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1. \quad (6)$$

*Definition 7 Laplace mechanism* [38]. Suppose a random query sequence  $f$  with a query of length  $n$ . Given a function  $f : D \rightarrow R^n$ , the global sensitivity is  $\Delta f$  and the privacy budget is  $\epsilon$ . The formula for  $\epsilon$ -differential privacy is as

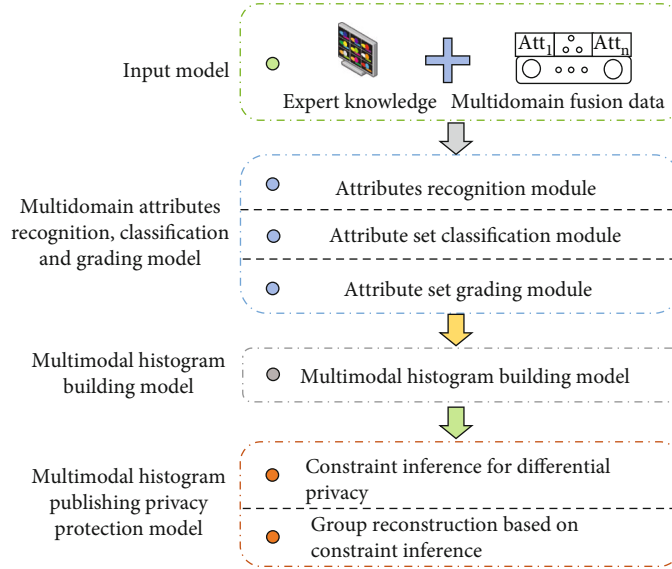


FIGURE 1: Multidomain fusion data privacy security framework.

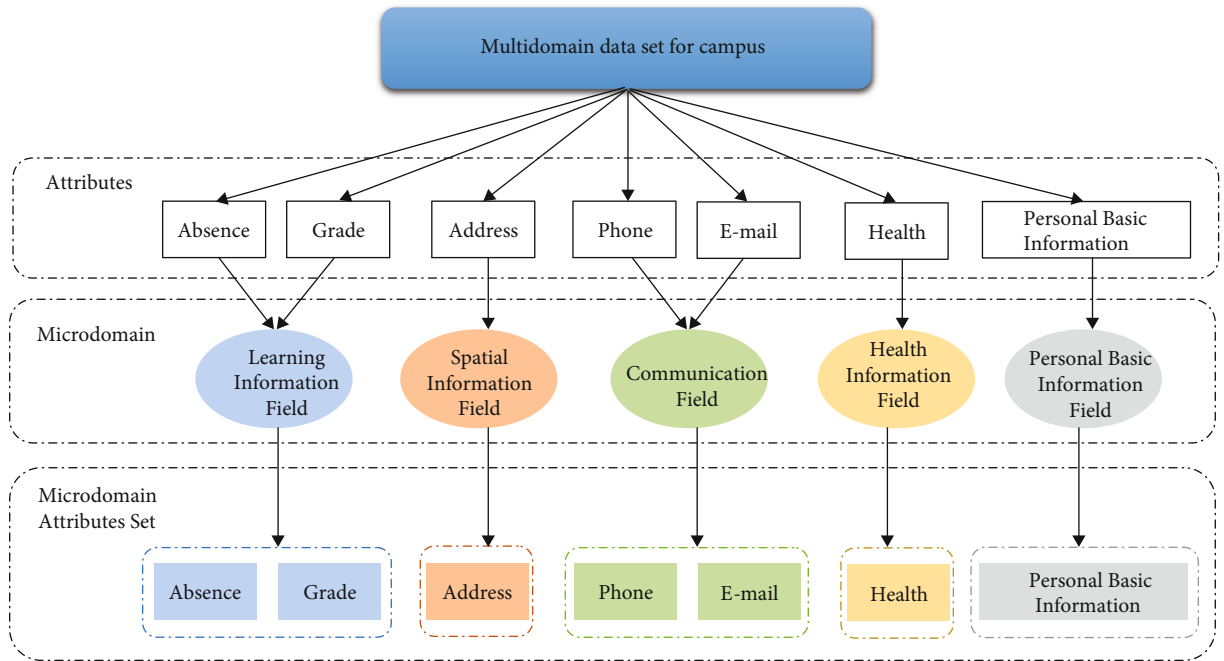


FIGURE 2: Recognition results of the multidomain data set for campus.

TABLE 1: Attribute sensitivity level table.

Grade	Value range
Higher	$0 < SV(Att_i) < 0.6$
Moderate	$0.6 \leq SV(Att_i) < 0.8$
Low	$SV(Att_i) \geq 0.8$

TABLE 2: Grading condition table for SPA sets.

Grade	Condition
Higher	$1/2 < \text{Num}(IG(X_n) > \gamma \times H(X_i)) / \text{Num}(DMA) \leq 1$
Moderate	$1/3 \leq \text{Num}(IG(X_n) > \gamma \times H(X_i)) / \text{Num}(DMA) \leq 1/2$
Low	$0 < \text{Num}(IG(X_n) > \gamma \times H(X_i)) / \text{Num}(DMA) < 1/3$

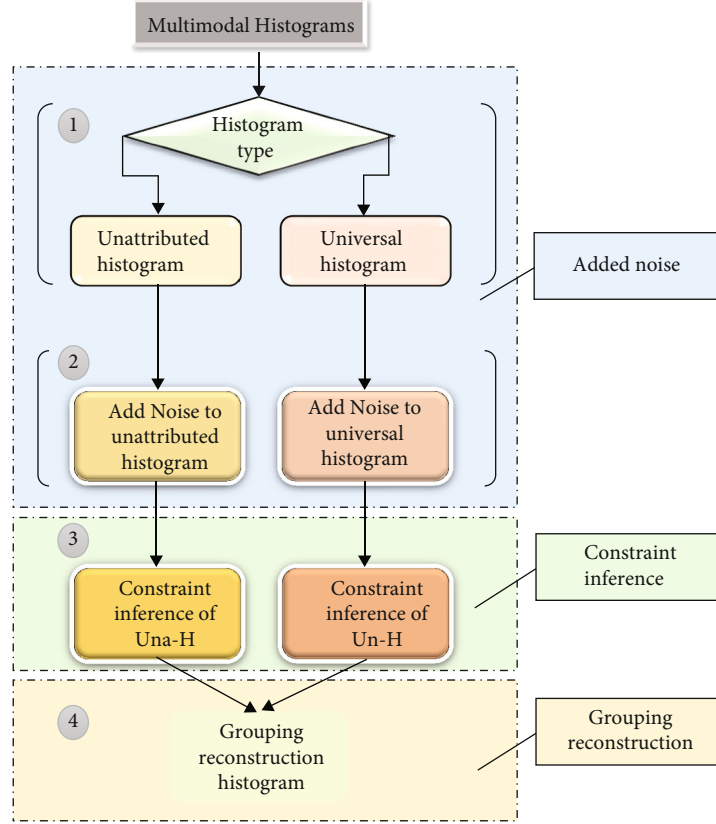


FIGURE 3: Privacy protection model for multimodal histogram data publishing.

follows:

$$\widetilde{f(D)} = f(D) + \langle \text{Lap}\left(\frac{\Delta f}{\varepsilon}\right) \rangle^n. \quad (7)$$

#### 4. Multidomain Fusion Data Privacy Security Framework

To solve the personal privacy security problem caused by multidomain personal privacy fusion data set containing strong background knowledge, we proposed a multidomain fusion data privacy security framework shown in Figure 1 and the framework includes four models: (1) the input model includes multidomain fusion data sets and expert knowledge; (2) Multidomain Fusion Data Recognition, Classification, and Grading model (MRCG); (3) Multimodal Histogram Building model (MHB) according to the result of recognition, classification, and grading builds multimodal histograms; and (4) Multimodal Histogram Publishing Privacy Protection model (MHPP), through constraint inference algorithm and grouping reconstruction algorithm to achieve the multimodal publishing histogram privacy protection.

**4.1. Multidomain Fusion Data Recognition Classification Grading Model.** In the MRCG model, we present the definition of microdomain and microdomain attribute set and combine them with expert knowledge, information entropy,

condition entropy, information gain to realize recognition classification and grading for the multidomain fusion data.

**4.1.1. Microdomain Attribute Recognition Module.** If data set  $D$  contains communication, location tracking, personal information, health and other personal privacy information, we call data set  $D$  multidomain personal privacy fusion data set. The different domains' personal privacy information in data set  $D$  is called microdomain of the multidomain fusion data set. The set of attributes that make up a microdomain is called microdomain attribute set. The definition is as follows.

**Definition 8 Microdomain.** Define a multidomain personal privacy fusion data set  $D = \{\text{Att}_1 \cdots \text{Att}_n\} (n > 0)$  and a domain expert knowledge set  $\text{ES} = \{F_1 \cdots F_n\}$ . According to the domain expert knowledge  $\text{ES}$ , the data set  $D$  is transformed into  $D = \{F_1 \cdots F_n\} (F_{(k \in n)} \in \text{ES}, n > 1)$ , which is a collection of different subfields. In this case, any subdomain  $F_{m \in n}$  in  $D$  is called the microdomain, i.e.,  $\widehat{D} = \{MF_1, \cdots, MF_n\} (MF_{m \in n} = F_{m \in n})$ .

**Definition 9 Microdomain attribute set.** Define a multidomain personal privacy fusion data set  $D = \{\text{Att}_1 \cdots \text{Att}_n\} (n > 0)$ . By Definition 8, get  $\widehat{D} = \{MF_1 \cdots MF_{n-1}, MF_n\}$ , and the set of attributes that make up microdomain  $MF_{k \in D}$  is called microdomain attribute set, i.e.,  $\text{MFAS}_k = \{\text{Att}_m \cdots \text{Att}_{m-1}, \text{Att}_m\} (\text{Att}_m \in D, 0 \leq m \leq n)$ .



```

Input:  $\widetilde{SQ}[i]$ : the noise frequency;
Output:  $\overline{SQ}$ : the constraint inference frequency
1 define listB, array[i], flag=false, start =0, end=0, sum=0, avg=0;
2 for  $i \in [0, i)$  // Line2-13 Judgment whether  $\widetilde{SQ}[i]$  is a order query sequence
3   if  $i=0$ 
4     listB  $\leftarrow \widetilde{SQ}[i]$ ; //add the  $\widetilde{SQ}[i]$  to listB
5   else
6     listB  $\leftarrow \widetilde{SQ}[i]$ ;
7     if  $\widetilde{SQ}[i] < \text{listB.get}(i-1)$ 
8       if flag = false;
9         start = i-1;
10        end = i;
11        flag =!flag;
12      else
13        End = i;
14  Handle(); //The recursive method of the order inference
15 end for
16 Handle() // Lines16-26 the order inference
17 for  $i \in [\text{start}, \text{end}]$ 
18   sum=sum+listB.get(i);
19 end for
20 avg = sum/(end - start +1);
21 for  $i \in [\text{start}, \text{end}]$ 
22   listB.set(i,avg);
23 end for
24 if start  $>=1$  && listB.get(start-1) $>$ avg;
25   start = start - 1;
26   Handle();
27  $\overline{SQ}[i] \leftarrow \text{arr}[i]$ ;
28 return the constraint inference frequency  $\overline{SQ}[i]$ ;

```

ALGORITHM 1: Constraint inference for unattributed histogram.

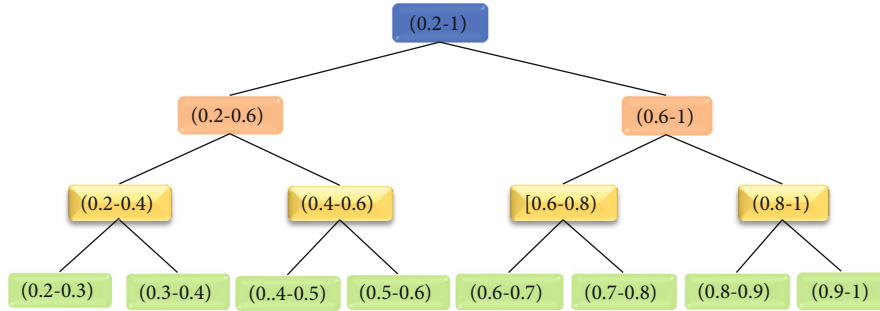


FIGURE 4: Query sequence tree based on the full binary tree.

In this section, take the multidomain personal privacy fusion data set for the campus as an example. The data set contains seven attributes: grade, absences, phone, E-mail, address, health, and personal basic information. Figure 2 shows the recognition result of data attributes based on Definitions 9 and 10 and expert knowledge.

**4.1.2. Microdomain Attribute Set Classification Module.** When we select any microdomain attribute set  $MFAS_k$  to publishing, the personal privacy in  $MFAS_k$  can directly represent the privacy characteristics of the microdomain  $MF_k$ .

Therefore, the attributes of the published microdomain attribute set  $MFAS_k$  are called direct privacy attribute. The definition is as follows.

*Definition 10 Direct privacy attribute (DPA).* Set the data attribute recognition result is  $\widehat{D} = \{MF_1 \cdots MF_{n-1}, MF_n\}$ ,  $MFAS_k = \{Att_m \cdots Att_{m-1}, Att_m\}$ . When we select the attribute set  $MFAS_k$  of any microdomain  $MF_k$  in  $\widehat{D}$  to publish, the attribute in  $MFAS_k$  is called the direct privacy attribute of microdomain  $MF_k$ .

```

Input:  $\widetilde{NL}[i]$ : the noise frequency array,  $h[i]$ : the height of each node in the tree array
Output:  $\overline{NL}$ : the constraint inference frequency
1 define array  $CV[j]$  storing the estimated value;
2 for each  $i$  in  $\widetilde{NL}$  // The full binary tree from the bottom-up
3 if ( $h[i]=0$ )  $i$  is the leaf node
4    $CV[j] \leftarrow \widetilde{NL}[i]$ ;
5 else
6   calculating  $CV[j]$  according to formula (11);
7 end for
8 for each  $j$  in  $CV[j]$  // The full binary tree from the top-down
9 if ( $h[j]=\log_2 n+1$ )  $j$  is the root node;
10   $\overline{NL}[i] = CV[j]$ ;
11 else
12  calculating  $\overline{NL}[i]$  according to formula (12);
13 end for;
14 return  $\overline{NL}$ ;

```

ALGORITHM 2: Constraint inference for universal histogram.

We get the DPA set through Definition 10, and the rest of the microdomain attributes set up other attribute set ( $OA, DPA \cup OA = D$ ), by calculating information gain between each attribute in  $OA$  and each attribute in  $DPA$  to implement attribute classification in  $OA$ . When the information gain is larger, it means that attributes in  $OA$  are more important to the attributes of the  $DPA$ ; otherwise, it is less important to attributes in  $DPA$ . The attributes in  $OA$  are classified as sensitive privacy attributes and implicit privacy attributes for  $DPA$  set.

According to Definition 10, the microdomain is divided into  $DPA$  set and other attribute sets ( $OA, DPA \cup OA = D$ ). This section introduces information entropy and personalization parameter  $\gamma$  to realize the attribute classification in the  $OA$  set. The value of parameter  $\gamma$  is any value within the range of effective classification. The larger the information gain value, the stronger the correlation between the attributes in  $OA$  and the attributes in  $DPA$ . Therefore, the attributes in  $OA$  include two categories: sensitive privacy attributes ( $SPAs$ ) and implicit privacy attributes ( $IPAs$ ). The definitions are as follows.

**Definition 11 Sensitive privacy attribute (SPA).** If  $IG(X_n) \geq \gamma \times H(X_i) (X_i \in DPA, X_n \in OA)$ , the attributes in the MFAS are called sensitive privacy attribute, where the parameter  $\gamma$  is threshold.

**Definition 12 Implicit privacy attribute (IPA).** If  $IG(X_n) < \gamma \times H(X_i) (X_i \in DPA, X_n \in OA)$ , the attributes in the MFAS are called implicit privacy attribute, where the parameter  $\gamma$  is a threshold.

For example, let the data set be  $D = \{Att_1, Att_2, Att_3, Att_4, Att_5\}$ . Firstly, according to Definitions 9 and 10, get  $\widehat{D} = \{MF_1, MF_2\}$ ,  $MFAS_1 = \{Att_1, Att_5\}$ , and  $MFAS_2 = \{Att_2, Att_3, Att_4\}$  and choose published microdomain attributed set  $MFAS_1$ . Secondly, we combine with Definition 11 to get  $DPA = \{Att_1, Att_5\}$  and  $OA = \{Att_2, Att_3, Att_4\}$ , and

then, according to Definition 4, calculate the information gain. Finally, based on Definitions 11 and 12, realize the attribute classification.

**4.1.3. Microdomain Attribute Set Grading Module.** Information entropy can measure the expectation of the amount of data attribute information. The maximum entropy reflects the data attribute maximum information expectation, so we denote the sensitivity of the attributes by the relative rate of attributes information entropy and maximum entropy. The formula for attribute sensitivity is as follows.

$$SV(Att_i) = \frac{H(Att_i)}{H_{\max}(Att_i)} \quad (SV(Att_i) \in (0, 1]). \quad (8)$$

As the formula is known, the smaller distance between the information entropy and maximum entropy, the stronger attribute sensitivity, otherwise the less sensitive of the attribute.

Sensitivity can effectively describe the importance of attribute privacy. In this section, we use the attributes sensitivity level table to achieve grading of attributes. Based on the personal privacy level grading in Personal Privacy Protection Law, we propose the sensitivity level table for the microdomain attribute set, as shown in Table 1.

Because there is a strong correlation between the attributes in the  $SPA$  set and the  $DPA$  set, we propose the attribute grading process in the  $SPA$  set from the perspective of information gain. And the grading condition for  $SPA$  sets as shown in Table 2.

- (1) Count the number of attributes satisfying  $IG(X_n) > \gamma \times H(X_i)$  in  $SPA$  set, denoted as  $Num(IG(X_n) > \gamma \times H(X_i))$
- (2) Count the number of attributes in  $DPA$  set, denoted by  $Num(DMA)$ ;
- (3) We implement the attribute grading in  $SPA$  set according to the following conditions

```

Input:  $\overline{NL}[i]$ : the constraint inference frequency,  $L[i]$ : the original histogram frequency;
Output: the grouping reconstruction result;
1 define an array DV[i] to store  $|\overline{NL}[i] - \overline{NL}[i+1]|$ ;
2 define the SS, SSer, SSEl store the error between the grouped histogram and the original histogram;
3 define avgR, avgL, avg store the average of the group histogram;
4 for  $i \in [0, k]$  // Lines 4-6 calculate the absolute value of the difference between adjacent numbers;
5 DV[i] =  $abs(\overline{NL}[i] - \overline{NL}[i+1])$ ;
6 end for
7  $avg = \sum_i \overline{NL}[i] / |\overline{NL}[i]|$ ; // Calculate the average frequency when all buckets are combined into a group;
8 calculate the SSE after merging a group;
9 for each i in DV[i] // Line 9-28 Group reconstruction based on SSE
10 if  $(\sum(DV[i])) = -|DV[i]|$ ; // group terminal condition;
11 break;
12 else
13 mid = getMaxId(); // get the max DV location value
14 DV[mid] = -1;
15 left  $\leftarrow$  getLeft(mid); // get the left boundary according to the max DV
16 right  $\leftarrow$  getRight(mid); // get the right boundary according to the max DV
17 flag[mid] = true;
18 for  $i \in [left, mid]$ 
19 avgR =  $\sum_{i=left}^{mid} \overline{NL}[i] / |mid - left + 1|$ ;
20 SSer =  $(L[i] - avgR)^2$ ;
21 end for
22 for  $i \in [mid+1, right]$ ;
23 avgL =  $\sum_{i=mid+1}^{right} \overline{NL}[i] / |mid - right + 2|$ ;
24 SSEl =  $(L[i] - avgL)^2$ ;
25 end for
26 SSE[i]  $\leftarrow$  SSer + SSEl;
27 record the result of the current grouping reconstruction;
28 end for
29 sort(SSE[i]) and select the minimum SSE;
30 return the grouping reconstruction of the minimum SSE

```

ALGORITHM 3: Group reconstruction.

Finally, according to the results of recognition, classification, and grading, combine the unattributed histogram and universal histogram [37] to realize multimodal mathematical statistics based on attribute sensitivity for multidomain fusion data sets.

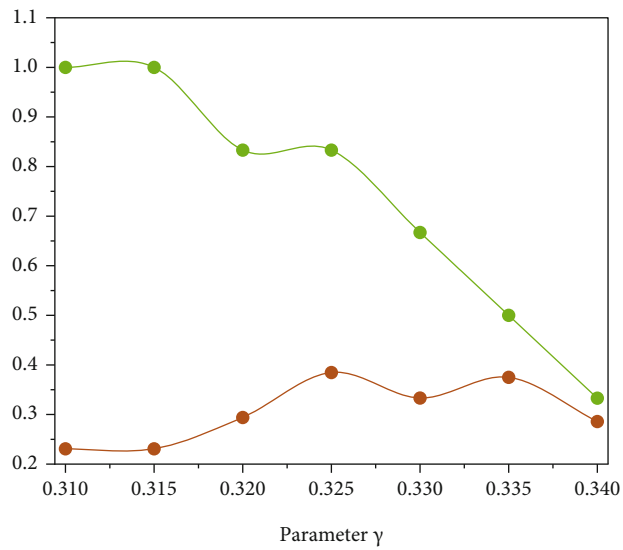
**4.2. Multimodal Histogram Publishing Privacy Protection Model.** To solve the problem of privacy leakage caused by strong background knowledge in multidomain personal privacy fusion data set, in this section, we propose a privacy protection model for multimodal histogram publishing. According to the differences between the unattributed histogram and the universal histogram in paper [39], we use the two different constraint inference algorithms and combine them with grouping reconstruction algorithm to achieve multimodal histogram data publishing privacy protection [35, 36].

**4.2.1. Model Overview.** Figure 3 shows the privacy protection model in this paper, including added noise, constraint inference, and grouping reconstruction.

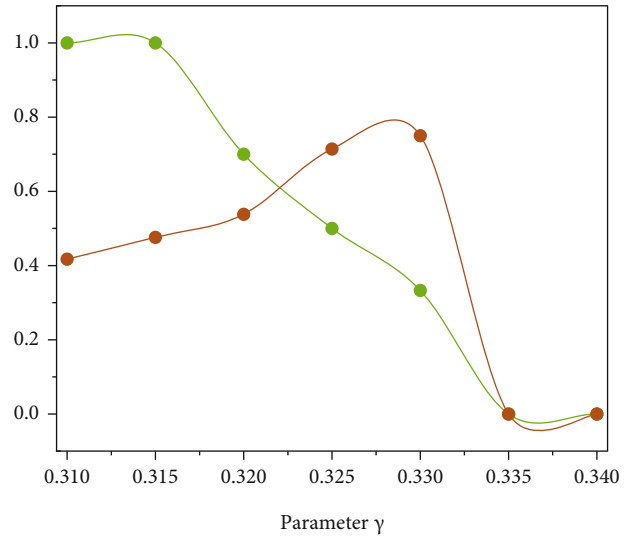
TABLE 3: Microdomain recognition results.

Microdomain	Attribute
Personal	Sex, age, school, guardian, Address type
Family	Pastatus, Fedu, Medu, Fjob, Mjob, Famrel, Famsize
Entertainment	Internet, activities, Freetime, Dalc, Walc, Goout
Campus	Higher, Schoolreason, G1, G2, G3, absence
After-school	Studytime, Schoolsup, paid, Famsup
Health	Health
Spatial	Traveltime
Emotion	Romantic

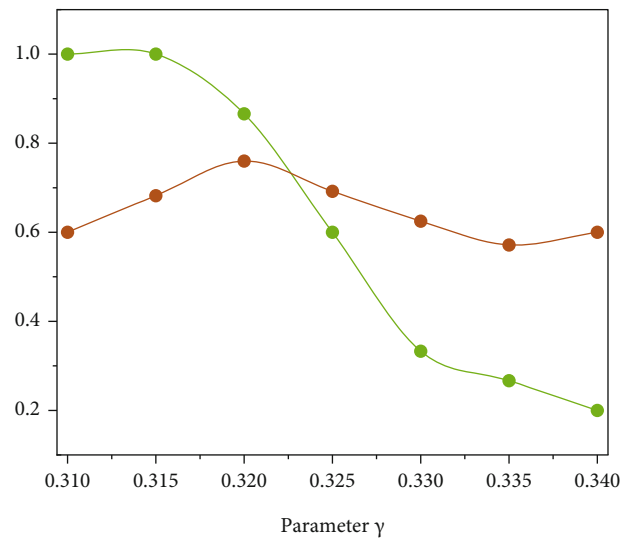
- (1) *Added Noise.* Firstly, we classify histograms by step ① in Figure 3 and add the Laplace noise to histograms by step ②.
- (2) *Constraint Inference for Differential Privacy (CDP).* We use positive-order inference and linear estimation for unattributed histogram and universal histogram with noise through step ③.



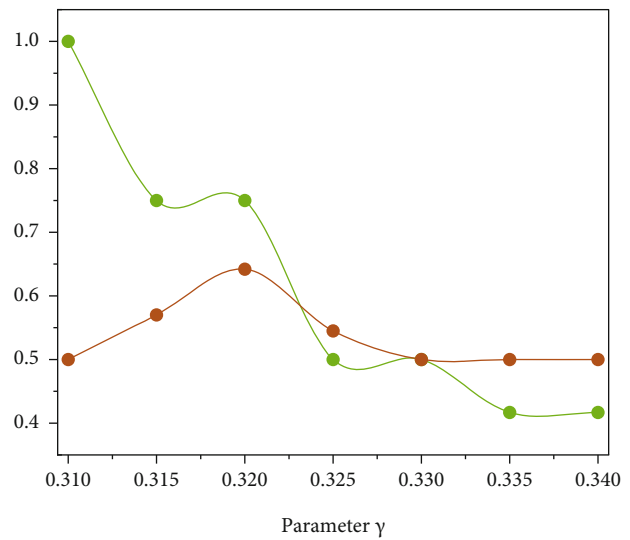
(a) Personal



(b) Family

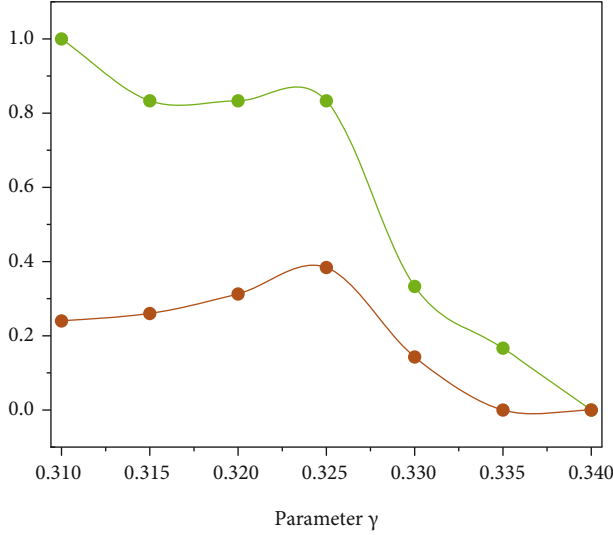


(c) After-school

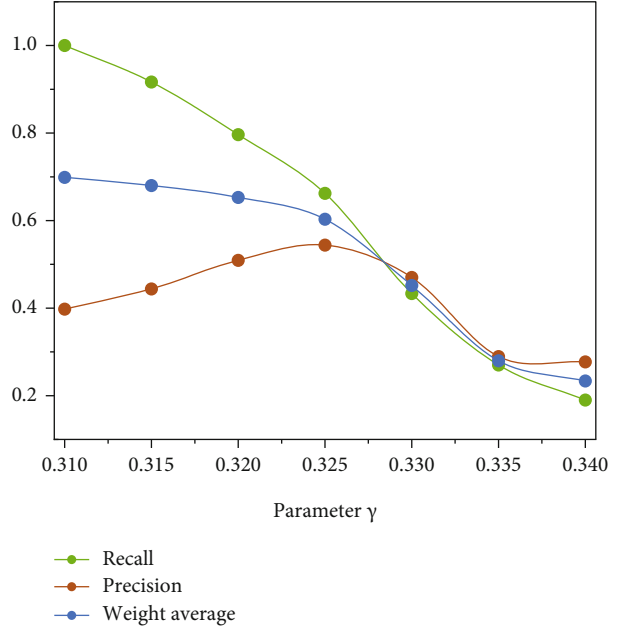


(d) Campus

FIGURE 5: Continued.



(e) Entertainment



(f) Weight average

FIGURE 5: The recall rate and precision rate under different parameter values in 5 microdomains.

(3) *Grouping Reconstruction Based on Constraint Inference (CDPR)*. According to the grouping reconstruction algorithm of step ④, obtain the best grouping of histogram publishing.

4.2.2. *Constraint Inference for Differential Privacy*. Constraint inference makes noise data approach actual data by query constraint conditions such as order and nonnegative. Firstly, add Laplace noise to original query sequence  $Q$ , which has constraint inference  $CQ$  to obtain the query sequence  $q_1 = Q1(I)$ . Then, by the constraint inference rules and L2 distance, calculate  $q_2$ , which is closest to  $q_1$ . The minimum L2 solution is defined as follows.

*Definition 13 Minimum L2 solution* [39]. Let  $Q$  be a query sequence with constraints  $CQ$ . Given a noisy query sequence  $q_1 = Q1(I)$ , a minimum L2 solution and denoted  $q_2$ , that is a vector, satisfy the constraints  $CQ$  and at the same time minimize  $\|q_1 - q_2\|_2$ .

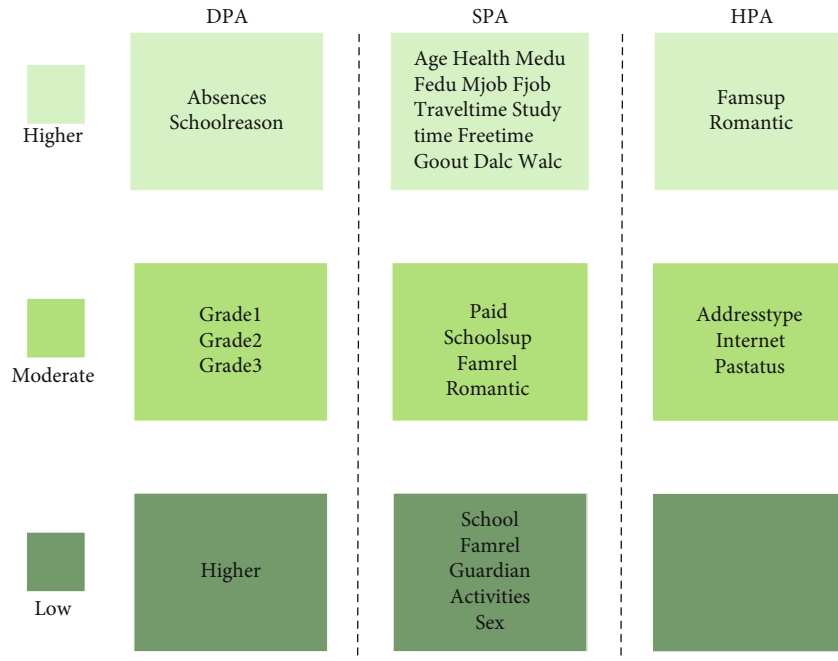
#### (1) Unattributed Histogram

*Add Noise*. Define the original unattributed histogram query sequence  $Q$ . Since we only care about the frequency distribution of the unattributed histogram. Therefore, any sort of query sequence is equivalent. In this section, we use the positive-order query sequence  $SQ$  to replace the original query sequence  $Q$ . For example, if the original query sequence  $Q = \{5, 10, 2, 3, 9\}$ , then the positive-order query sequence  $SQ = \{2, 3, 5, 9, 10\}$ .

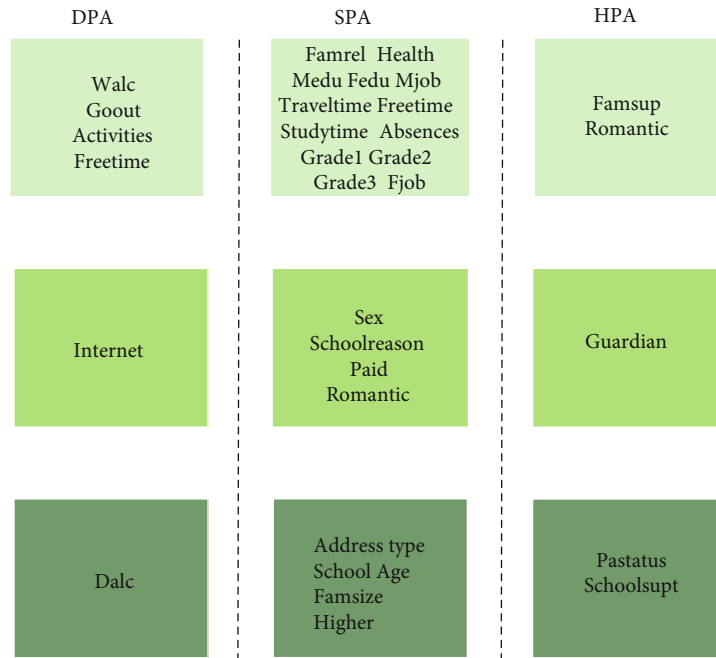
According to Definitions 6 and 7, the global sensitivity of the query sequence is  $\Delta f = 1$ , and the randomized algorithm

TABLE 4: Attribute sensitivity.

Microdomain	Attribute	Sensitivity
Personal	Sex	0.997960
	Age	0.962287
	School	0.519086
	Addressstype	0.765220
	Guardian	0.724476
Family	Pastatus	0.480919
	Fedu	0.872324
	Medu	0.860186
	Fjob	0.726524
	Mjob	0.928905
	Famrel	0.760564
	Famsize	0.866916
Entertainment	Internet	0.651001
	Activities	0.999773
	Freetime	0.869073
	Dalc	0.569988
	Walc	0.915546
Campus	Goout	0.921601
	Higher	0.289079
	Schoolreason	0.933194
	G1	0.735019
	G2	0.735018
	G3	0.739303
	Absence	0.965337
After-school	Studytime	0.850267
	Schoolsup	0.555003
	Paid	0.994959
	Famsup	0.963063

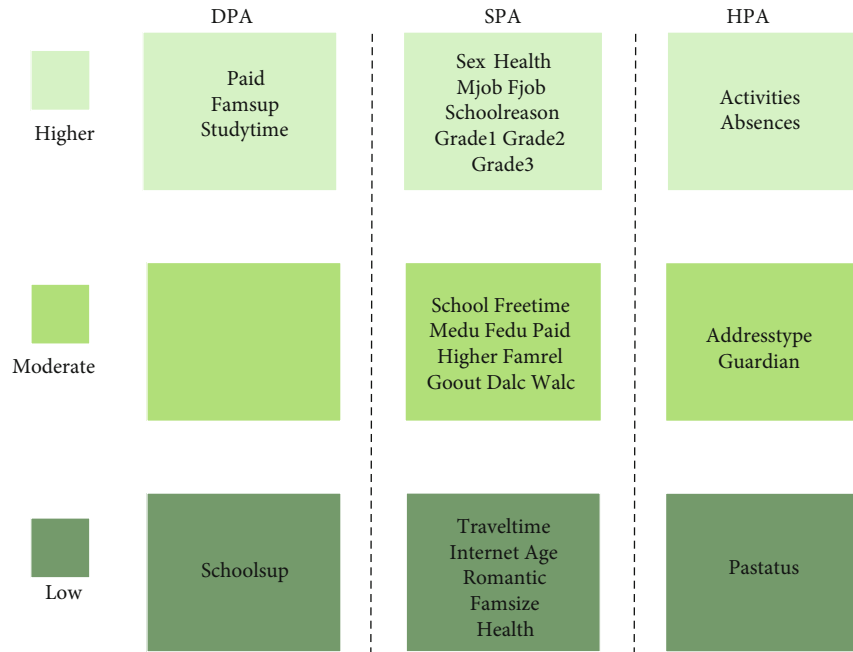


(a) Campus

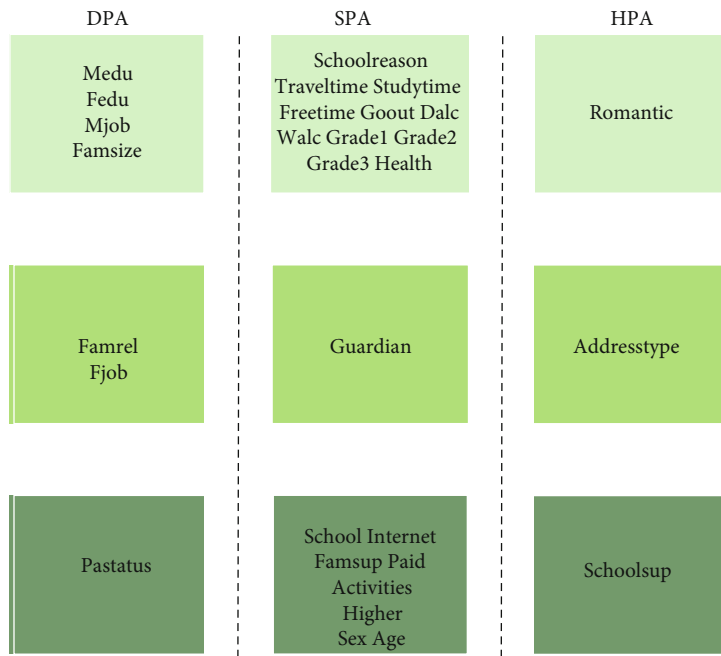


(b) Entertainment

FIGURE 6: Continued.



(c) After-school



(d) Family

FIGURE 6: Continued.

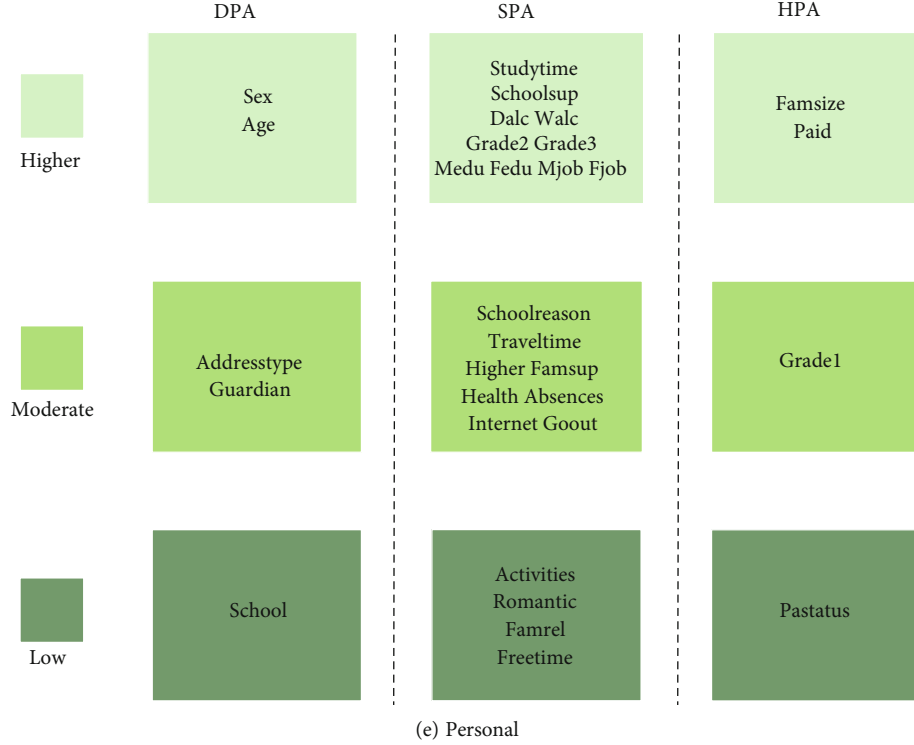


FIGURE 6: Grading results in campus, entertainment, after-school, family, and personal.

$\widetilde{SQ}$ , which satisfies  $\varepsilon$ -differential privacy, is as follows.

$$\widetilde{SQ} = SQ + \langle \text{Lap}\left(\frac{1}{\varepsilon}\right) \rangle^n. \quad (9)$$

The query sequence  $SQ$  satisfies  $SQ[i] \leq SQ[i+1]$ ,  $1 \leq i < n$ .

*Positive-Order Inference.* Given a query sequence  $\widetilde{S} = \widetilde{SQ}(I)$ , this algorithm is aimed at finding a query sequence  $\bar{S}$  that satisfies constraint condition  $\bar{S}[i] \leq \bar{S}[i+1]$ ,  $1 \leq i < n$  and minimizes  $\|\widetilde{S} - \bar{S}\|_2$ .

**Theorem 14.** Let  $L_n = \min_{j \in [n, k]} \max_{i \in [1, j]} M[i, j]$  and  $U_n = \max_{i \in [1, n]} \min_{j \in [i, k]} M[i, j]$ ; then, the result satisfies the positive-order constraint condition is  $\bar{S}[k] = L_n = U_n$ .  $M[i, j] = \sum_{k=i}^j \bar{S}[k] / j - k + 1$ .

We use the two cases to expound. The first case assumes the noise sequence is  $\widetilde{S} = \{2, 6, 8\}$ , it satisfies the positive-order constraint condition, and the final constraint inference sequence result is  $\bar{S} = \widetilde{S}$ . In the second case, if the query sequence  $\widetilde{S} = \{2, 8, 6\}$  is unordered, according to Theorem 14, get the  $\bar{S} = \{2, 7, 7\}$ . The constraint inference is described in detail in Algorithm 1.

Algorithm 1 is a constraint inference for unattributed histograms. Line 1 defines list  $B$ , the array  $\text{arr}[k]$ , the sum variable  $\text{sum}$ , and the average variable  $\text{avg}$  in the recursive method. Lines 2-15 are the sequential check and positive-order constraint inference of the current noise sequence. Lines 7 to 13

are used to check whether the noise sequence is order. In line 14, we use the recursive method of the order inference to realize the order of the sequence. Lines 16 to 26 are the implementation of the order inference. Finally, we obtain an inference query sequence  $\bar{SQ}[i]$  that satisfies constraint condition  $\bar{S}[i] \leq \bar{S}[i+1]$ ,  $1 \leq i < n$  and minimizes  $\|\widetilde{S} - \bar{S}\|_2$ .

#### (2) Universal Histogram

*Add Noise.* Define the original universal histogram query sequence  $L$ . The unit interval of the universal histogram has meaningful. Therefore, overmuch noise accumulation in long-range queries leads to low accuracy and poor availability of the query sequence results. In order to reduce the cumulative error, this section replaces the original query sequence  $L$  by creating a query sequence that supports long-range queries.

Universal histograms support any interval frequency queries, and the query of any interval frequency is based on unit interval frequency statistics. The frequency of the unit interval is the same as the leaf nodes of the tree, and any other interval is the same as other nodes of the tree. In this section, we use the full binary tree to create a long-range sequence  $NL$  replace the original query sequence.

Building a full binary tree of height  $h$ , the node set  $V$  includes the leaf node set  $lv$  and the other node set  $v$ . In a full binary tree, the parent nodes in each layer calculates from the query interval of the corresponding child nodes and the leaf nodes are the unit interval in the original query sequence  $L$ .

For example, the original query sequence is replaced by the full binary tree with  $k = 2$  and  $h = 4$  as shown in Figure 4. The original query sequence is  $L = \{[0.2-0.3], [0.3-0.4], [0.4-0.5], [0.5-0.6], [0.6-0.7], [0.7-0.8], [0.8-0.9], [0.9-1.0]\}$  and the replace query sequence is  $NL = \{[0.2-1.0], [0.2-0.6], [0.6-1.0]\}$



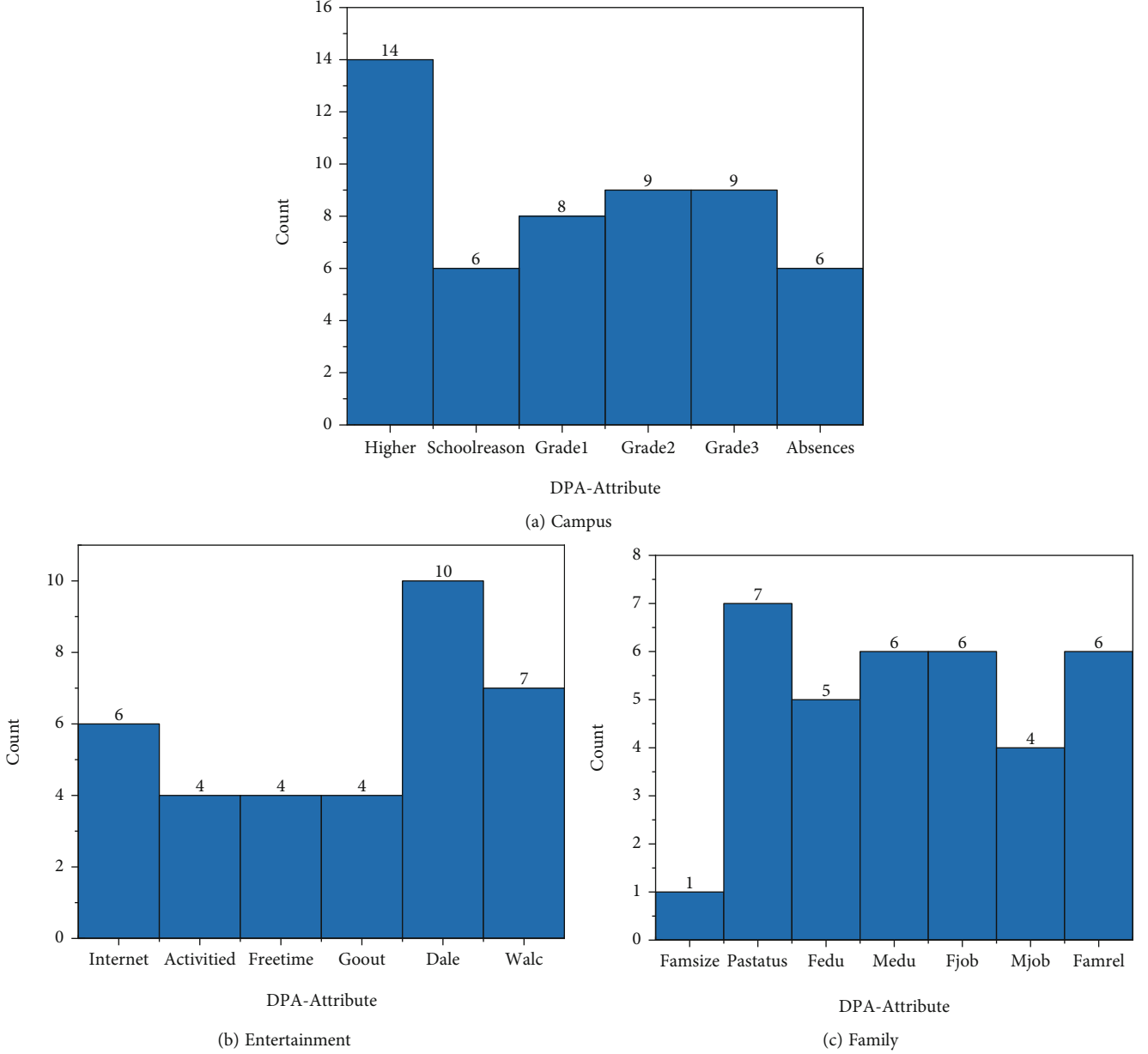


FIGURE 7: Unattributed histograms in campus, entertainment, and family.

, [0.2-0.4], [0.4-0.6], [0.6-0.8], [0.8-1.0], [0.2-0.3], [0.3-0.4], [0.4-0.5], [0.5-0.6], [0.6-0.7], [0.7-0.8], [0.8-0.9], [0.9-1.0]}.

According to Definition 6, the global sensitivity of the query sequence NL is the height of the full binary tree, and we get the randomized algorithm  $\widetilde{\text{NL}}$  which satisfies  $\epsilon$ -differential privacy by Definition 7.

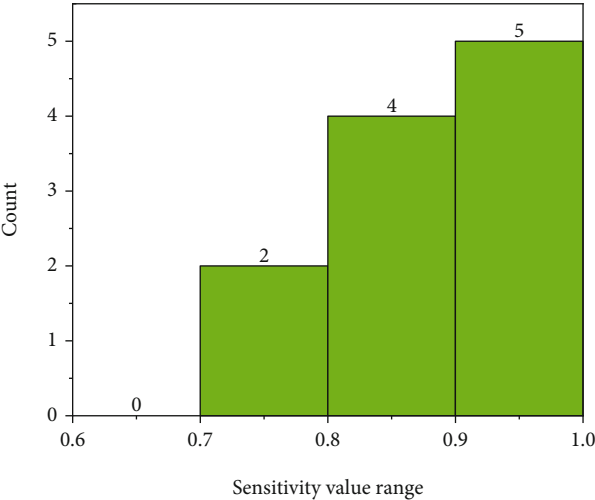
$$\widetilde{\text{NL}} = \text{NL} + \langle \text{Lap}\left(\frac{h}{\epsilon}\right) \rangle^n. \quad (10)$$

*Linear Estimation.* After noise added to query sequence NL, the parent node frequency is not equal to the sum of corresponding child node frequency in the full binary tree, so we use linear estimation to constraint inference. Finally,

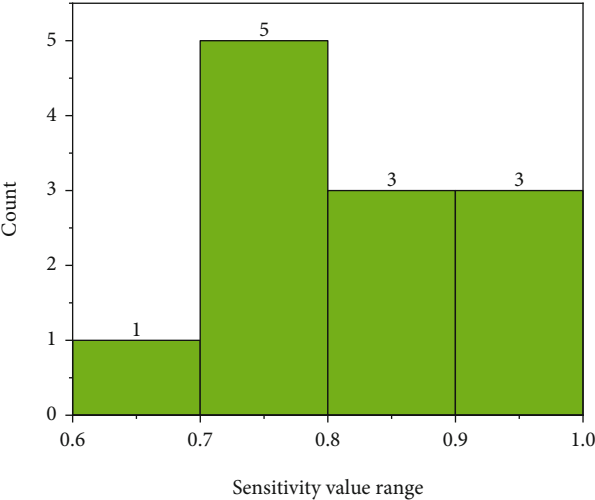
we find a query sequence  $\widetilde{\text{NL}}$ , which satisfies the constraint condition and minimizes  $\|\widetilde{\text{NL}} - \text{NL}\|_2$ .

Firstly, we calculate the linear estimate of the full binary tree from the bottom-up. If the node is leaf node  $e(v) = \widetilde{\text{NL}}(lv)$  and if it is not leaf nodes use the current node noise value  $\widetilde{\text{NL}}(v)$  and its child node linear estimate  $e(v)$  recursive computation, the linear estimation formula is shown below.

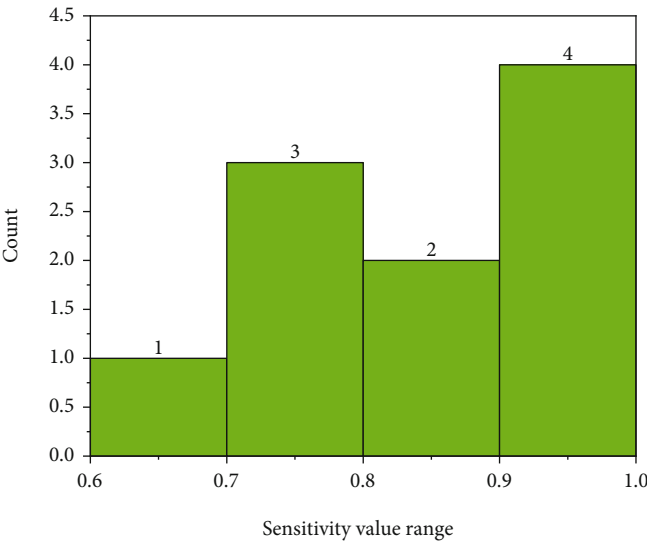
$$e(v) = \begin{cases} \widetilde{\text{NL}}(lv) & \text{if } lv \text{ is leaf node,} \\ \frac{k^h - k^{h-1}}{k^h - 1} \widetilde{\text{NL}}(v) + \frac{k^{h-1} - 1}{k^h - 1} \sum_{m \in s(v_p)} e(m) & v = V - lv. \end{cases} \quad (11)$$



(a) Campus



(b) Entertainment



(c) Family

FIGURE 8: Universal histograms in campus, entertainment, and family.

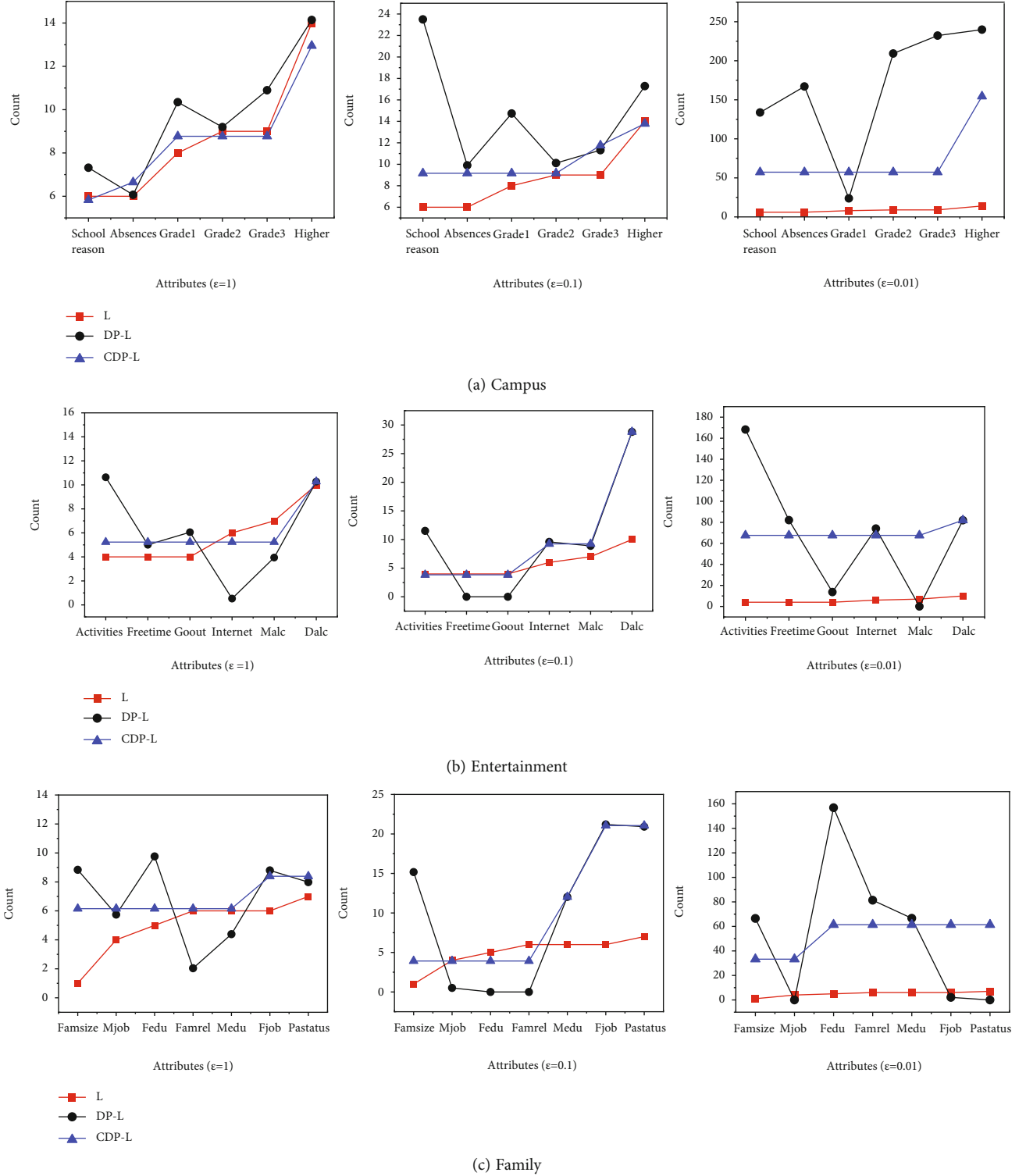


FIGURE 9: Frequency query results in campus, entertainment, and family.

In the formula, the  $k$  denotes full binary tree fan-out,  $h$  is the height of the current node  $v_p$ , and the  $s(v_p)$  represents the child node set of the current node  $v_p$ .

Based on the current estimate  $e(v)$ , we adopt the top-down linear estimation calculation to full binary tree, if the current node is root node  $\bar{NL}(v_{\text{root}}) = e(v_{\text{root}})$ . In the top-

down traversal, if the parent node frequency is not equal to the sum of child node frequency, we use the following formula for linear estimation to constraint inference. The details of the algorithm are shown in Algorithm 2.

Defining the noise query sequence  $\widetilde{NL} = \widetilde{NL}(I)$ , the bottom-up estimation value is  $e(v)$ ; then, the bottom-up

TABLE 5: Mean absolute error table.

Privacy budget	Algorithm	Campus	Entertainment	Family
$\varepsilon = 1$	CDP	0.6665	0.8331	0.5713
	LP	0.9919	0.9976	1.0017
$\varepsilon = 0.1$	CDP	8.3316	7.1652	6.4272
	LP	9.9958	9.9897	9.9796
$\varepsilon = 0.01$	CDP	58.9882	46.1574	44.8481
	LP	99.7955	99.1678	100.1241

linear estimation formula is as follows.

$$\overline{NL}(v) = \begin{cases} e(v_{\text{root}}), & \text{if } v \text{ is the root,} \\ e(v) + \frac{1}{k} \left[ \overline{NL}(v) - \sum_{m \in s(v_p)} e(m) \right], & v = V - v_{\text{root}}. \end{cases} \quad (12)$$

Algorithm 2 is a constraint inference for universal histograms. The formulas in lines 1 and 2 are the core of Algorithm 2. Line 3 defines an array CV to store the estimates by the formula on line 1. Lines 4 to 9 are the bottom-up calculation of the linear estimation and the top-down linear estimation calculation in lines 10 through 15. Finally, we get a query sequence  $\overline{NL}$  that satisfies the constraint condition and minimizes  $\|\widetilde{NL} - \overline{NL}\|_2$ .

**4.2.3. Grouping Reconstruction Algorithm Based on Constraint Inference.** After constraint inference, the published histogram error is caused by noise error (NE), and the total error of grouping reconstruction based on constraint inference includes noise error (NE) and reconstruction error (CE). The sum of squares due to error (SSE) can measure the total error between the published histogram with privacy protection and the original histogram. The sum of squares error (SSE) formula is as follows:

$$SSE = \sum_{i=1}^n \sum_{j=1}^n (D_i - ND_j)^2. \quad (13)$$

$D_i$  is the original data and the  $ND_j$  is the noise data. When the SSE is smaller, the absolute error value becomes smaller and the published histogram data availability becomes better. We find the minimum SSE to implement the best the grouping reconstruction. The core steps of grouping reconstruction algorithm are as follows.

The idea of this algorithm is to find the best grouping strategy by calculating the SSE between the group reconstruction histogram and the original histogram. The input is constrained inference sequence  $\overline{NL}[i]$  and the original query sequence  $L[i]$ . The output is the best group strategy and minimum SSE. Lines 1 to 3 define some variables to support the algorithm. According to lines 4-6, we calculate the absolute value of the difference values between adjacent buckets for the constraint inference result  $\overline{NL}[i]$  and stored

in  $DV[i]$ . Lines 7 and 8 calculate the SSE when all buckets are combined into a group. The core algorithm of grouping reconstruction was based on SSE from lines 9 to 28. From lines 13-17, find the maximum value of the DV to group the sequence; then, according to lines 18-28, calculate the SSE of the current grouped sequence until satisfied conditions of the lines 10 and 11 to stop grouping. We sort the results of SSE to find the minimum SSE and get the best group strategy in lines 29 and 30.

The CDPR algorithm includes the CDP algorithm and the grouping reconstruction algorithm. In the CDP algorithm, both the attribute-free histogram and the general histogram are added with  $\text{Lap}(\Delta f/\varepsilon)$  noise through mathematical inferences based on conditional constraints to improve data availability. The grouping reconstruction algorithm structurally optimizes the data availability of the published histogram based on the CDP algorithm and still satisfies the  $\varepsilon$ -differential privacy. The grouping reconstruction algorithm realizes data optimization and improves the availability of the published histogram data based on the histogram structural characteristics. Moreover, the grouping reconstruction algorithm also satisfies  $\varepsilon$ -differential privacy.

In group reconstruction based on constraint inference algorithm, the time complexity of the histogram group reconstruction algorithm is  $O(n^2)$ . Although the CDPR algorithm has a high time complexity, the time complexity is within an acceptable range, and the CDPR algorithm can improve long-distance query accuracy and data availability.

## 5. Experimental Results and Analysis

In this section, we use the real data set to analyze the multimodal histogram data publishing framework experiment. Firstly, we analyze the microdomain recognition experiments and the microdomain classification grading experiments. Secondly, we build multimodal histograms based on three microdomain and analyze the risk of histograms. Moreover, we use the multimodal histograms for the comparison experiment of the privacy protection model. Finally, comparing the LP algorithm and MDHP algorithm to prove the MHPP model has the advantage of low data error.

### 5.1. Experimental Setting

**5.1.1. Experimental Data Set.** The experimental data comes from a questionnaire filled out by 788 students [40], which contains 37 questions, and the final data set consists of 32 attributes. In the preprocessing of the data set, a valueless attribute was deleted, and the value in the attributes grade 1, grade 2, and grade 3 was converted into five levels: A, B, C, D, and F. Finally, after preprocessing and recognition processing, the data set contains 8 microdomains, 31 attributes, and 395 records.

Table 3 shows the 8 microdomains included in the multidomain fusion data set are as follows: personal, family, entertainment, campus, after-school, health, spatial, and emotion. Among them, campus is the performance of students in school and after-school is the content related to students' spare time.

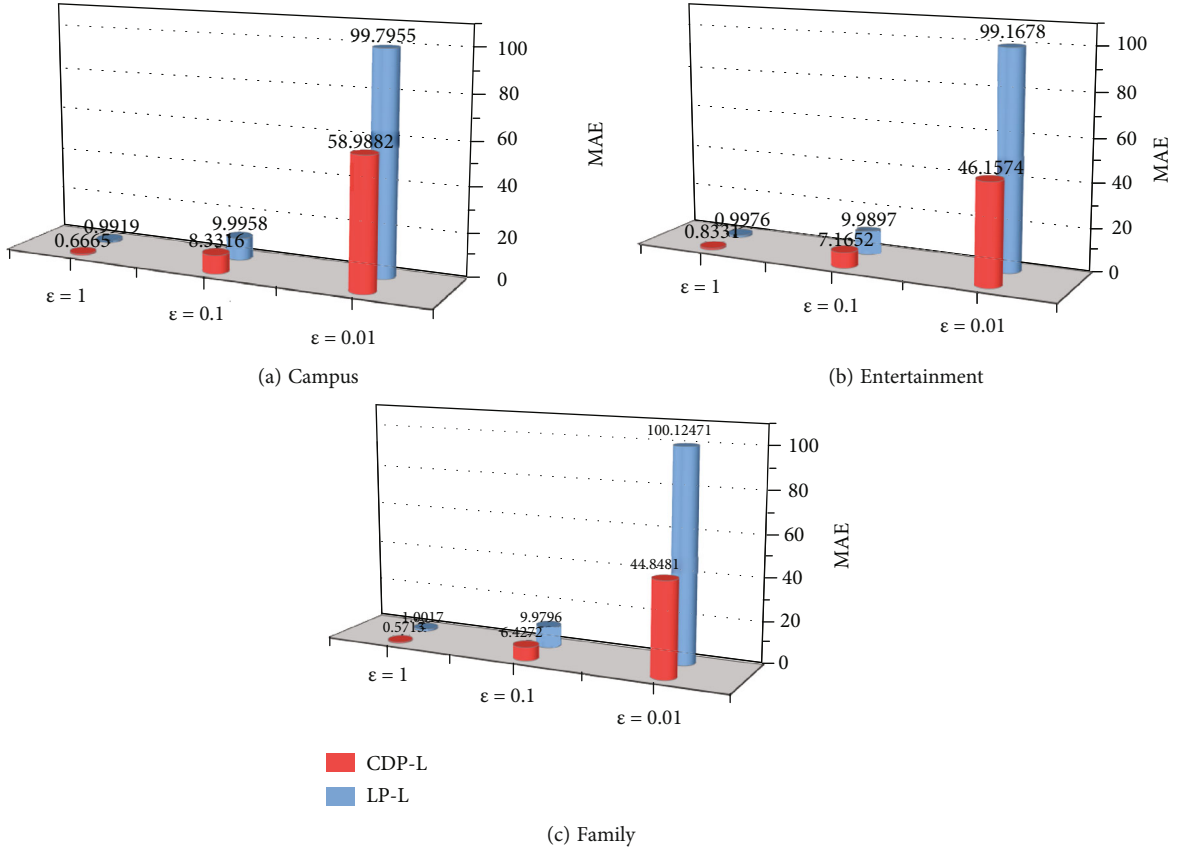


FIGURE 10: Mean absolute error under different privacy budgets.

There is only one attribute in the three microdomains of health, spatial, and emotion which leads to low experimental significance. Therefore, the following experiments in this paper select personal, family, entertainment, campus, and after-school for experimental analysis.

### 5.1.2. Experimental Parameter Set

(1) *Parameter  $\gamma$  Set.* In this section, we compare and analyze the recall rate and precision rate of five microdomains (personal, family, after-school, campus, and entertainment) at different parameters to determine the range of experimental parameters. The value range of the experimental parameters in this section is [0.31, 0.34].

Figures 5(a)–5(e) show the recall rate and precision rate under different parameter values in 5 microdomains. When the parameter  $\gamma = 0.31$ , the recall rate in the five microdomains is 100%. The reason for this problem is that the recall rate is to calculate how many positive samples in the original sample are predicted correctly. Assume that the SPA set is a positive sample, and the HPA set is a negative sample, when the parameter  $\gamma = 0.31$ , the attributes in the five microdomains are basically classified into the positive sample SPA set. However, the HPA set cannot be effectively classified from the OA set under the parameter  $\gamma = 0.31$ . When the parameter  $\gamma = 0.335$ , the precision rate is 0 in the two microdomains of family and entertainment. At the same time,

when the value range is [0.335, 0.34], the recall rate is stored as 0, which indicates that the parameter  $\gamma = 0.335$  cannot effectively classify the OA set. In summary, when the parameter  $\gamma \in (0, 0.31]$  or  $\gamma \in [0.335, 1)$ , the OA set cannot be effectively classified into the SPA set and the HPA set. Therefore, the experimental parameter  $\gamma \in (0.31, 0.335)$  in this paper can effectively classify the OA set into SPA set and HPA set. In the classification process, the recall rate and the precision rate have the same importance. This section realizes the weighted average calculation of different parameter  $\gamma$  based on the average recall rate and average precision rate of different parameter  $\gamma$  in 5 microdomains. According to the weighted average curve shown in Figure 5(f), it can be seen that the weighted average of the parameter  $\gamma = 0.315$  performs better within the effective value range. Therefore, the experimental parameter  $\gamma$  in this paper will be 0.315.

(2) *Parameter  $\epsilon$  Set.* This experiment uses the LP algorithm [25] and the MDHP algorithm [18] to compare and analyze the privacy protection model for multimodal histogram data publishing. In this experiment, we set the privacy budget  $\epsilon$  to 0.01, 0.1, and 1 [30] based on ensuring a reasonable allocation of the privacy budget.

5.2. *Multidomain Attribute Set Classification and Grading Results.* We calculate the attribute sensitivity ( $SV(Att_i)$ ) in the microdomain attribute set by the attribute sensitivity formula (X). When the  $SV(Att_i)$  is larger, the uncertainty of the

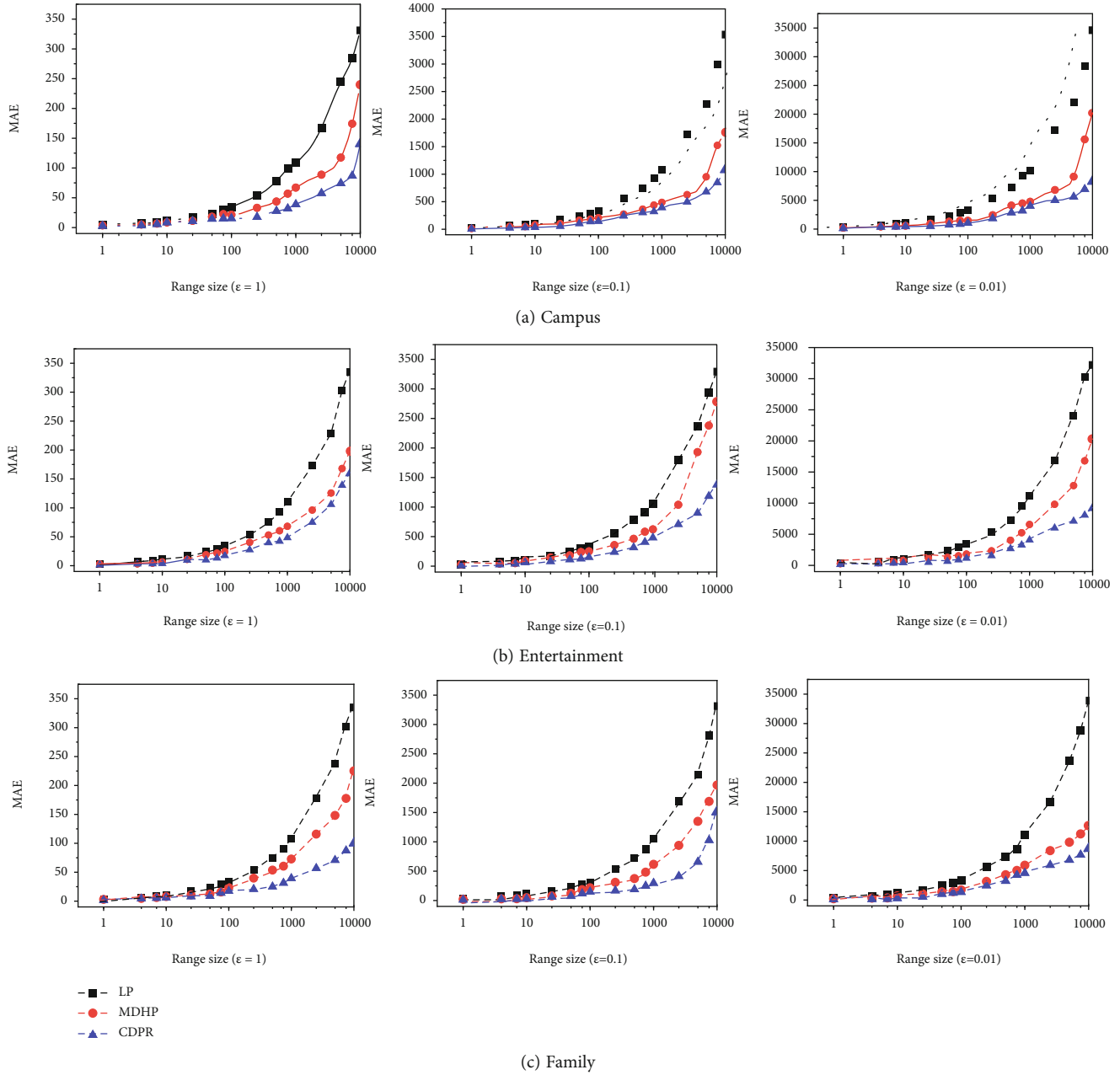


FIGURE 11: Mean absolute error of unattributed histograms in campus, entertainment, and family.

information is more substantial and the information value of the attribute is greater. The attribute sensitivity results are shown in Table 4.

Figure 6 shows the classification results of DPA, SPA, and IPA attribute sets for different microdomains. The attribute set of DPA and IPA is graded by the attribute sensitivity of Table 4 and the attribute sensitivity level of Table 1 in Section 4.1.3. In contrast, the SPA attribute set uses the grading condition of Table 2 in Section 4.1.3.

**5.3. Multimodal Histogram Building.** In the multimodal histogram building experiment, we select campus, entertainment, and family to build multimodal histograms. In

addition, we also analyze the risk of privacy leakage in the multimodal histogram.

**5.3.1. Unattributed Histogram.** Figure 7 shows the frequency of attribute distribution that satisfies  $IG(Att_i) > 0.315 \times H(Att_x \in DPA)$  and  $SV(Att_i \in SPA) \geq 0.8$  in the SPA set of the campus, entertainment, and family. In the histogram, the abscissa is the attribute of the DPA set, and the ordinate is the frequency distribution of highly sensitive attributes in the DPA attribute sets and give the following formal conditions.

$$\{Att_i \in SPA \mid SV(Att_i) \geq 0.8 \cap IG(Att_i) > 0.315 \times H(Att_x \in DPA)\}. \quad (14)$$

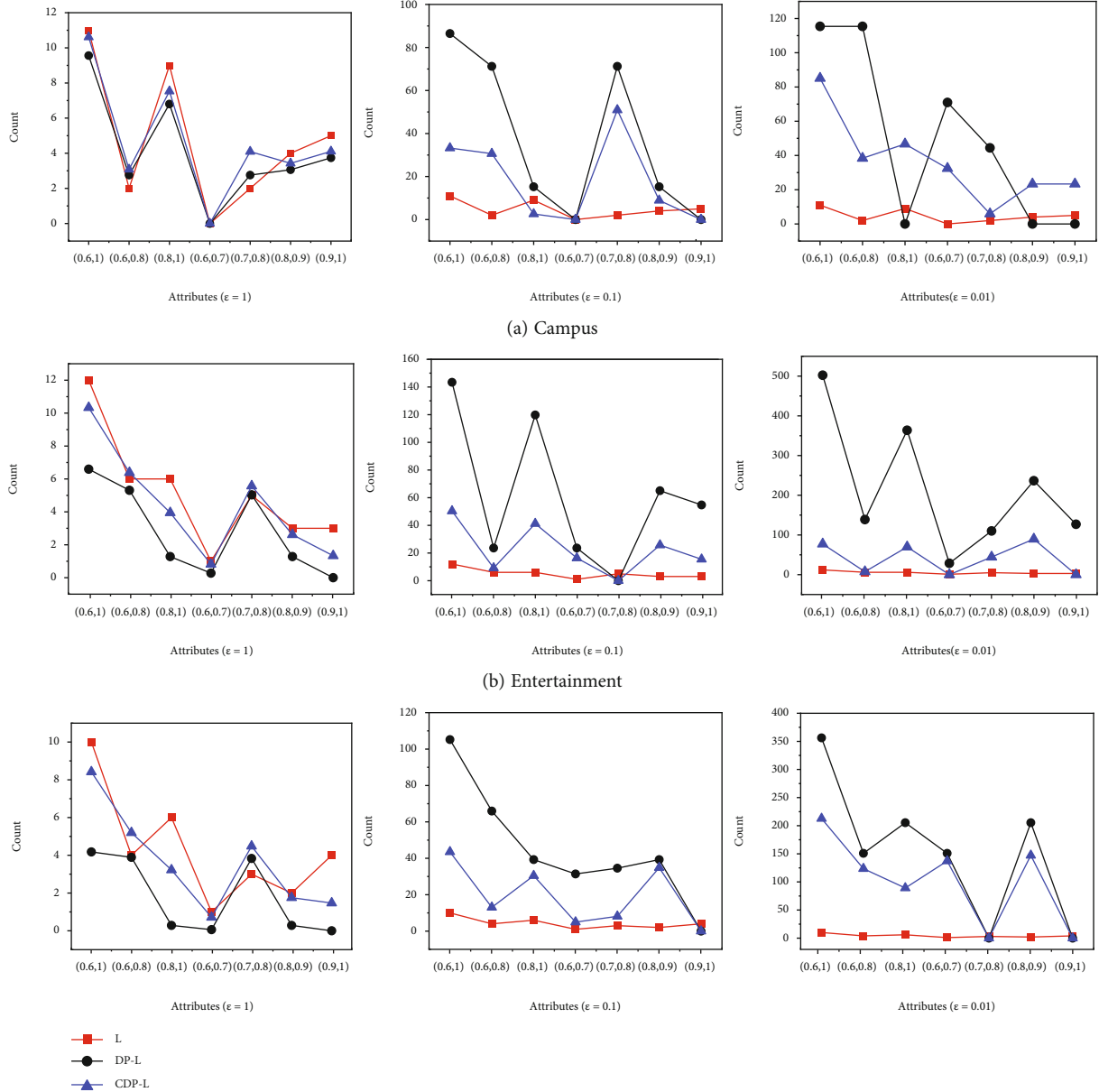


FIGURE 12: Family frequency query results in campus, entertainment, and family.

Take Figure 7(a) as an example, if the attacker grasps the total number of highly sensitive attributes in the DPA attribute sets of campus is 14. Moreover, we still know the 13 attributes in the DPA attribute set and the highly sensitive attributes in higher attribute set. Then, the attacker can infer the remaining highly sensitive attributes by combining the attribute sensitivity in Table 4. At this time, the highly sensitive attributes are not only leaked, but also lead to privacy leaks and even malicious recommendations through the attacker’s data mining.

5.3.2. *Universal Histogram.* We select campus, entertainment, and family to build universal histograms. Figure 8 shows the distribution of attributes with a sensitivity level of higher and over 0.6 in the SPA attribute set. The sensitivity range of the abscissa in the universal histogram is [0.6, 1],

and the ordinate is the number of SPA collection attributes satisfying the level of higher in the corresponding sensitivity range.

In Figure 8(a), assume that the attacker knows the frequency of campus in the sensitivity range of [0.9, 1] is 5 and knows the names of the 4 attributes. At this time, the attacker combines the attributes of the higher level in the SPA set with the attribute sensitivity in Table 4 can infer more private information and lead to private leaks. Due to the microdomain attribute set contains strong privacy information, they face a greater risk of privacy leakage.

5.4. *Microdomain Privacy Data Publishing of Privacy Protection Result Analysis.* In this experiment, the mean absolute error (MAE) is used to calculate the error between

the original frequency and the frequency after privacy protection. The MAE can reflect the frequency availability of multimodal histograms published after privacy protection. The formula of MAE is as follows.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |Q_i(m) - \tilde{Q}_i(m)|^2. \quad (15)$$

In the formula,  $n$  is the number of buckets of the histogram,  $Q_i(m)$  represents the original histogram frequency, and  $\tilde{Q}_i(m)$  is the histogram frequency value after privacy protection. According to the formula, if the value of MAE is smaller, the distance between the histogram frequency value after privacy protection and the original histogram frequency value is closer, which shows that the availability of frequency data is better.

*5.4.1. Unattributed Histogram.* The core algorithms of the privacy protection model for multimodal publishing histograms proposed in this paper are the constraint inference for differential privacy algorithm (CDP) and the grouping reconstruction based on constraint inference (CDPR). First, we compare and analyze the query accuracy and the mean absolute error of the CDP algorithm and the LP algorithm in this section experiment. Second, we choose the LP algorithm, the MDHP algorithm, and the CDPR algorithm proposed in this paper to compare and analyze the mean absolute error of the query. Through the analysis of the above comparative experiments, it proved that the privacy protection model for multimodal publishing histograms proposed in this paper can not only effectively guarantee the privacy of publishing histograms but also improve the availability of data.

*(1) Analysis of the Query Results.* In this experiment, the original unattributed histogram frequency set  $L$  is used as the baseline, and observe the distance from the baseline  $L$ . In the differential privacy protection process, since the noise added to the frequency is random with negative numbers and decimals, this experiment uses nonnegative processing and rounding processing for the noise frequency. In the experiment, the privacy budget  $\epsilon$  is set to 1, 0.1, and 0.01. As the privacy budget  $\epsilon$  decreases, the requirement for privacy protection is stronger, and it shows that more random noise was added. We define the frequency results of the CDP algorithm and the LP algorithm as CDP-L and LP-L.

Figure 9 shows the frequency query results of unattributed histograms protected by CDP and LP algorithms in different microdomains. We observe that under different privacy budget  $\epsilon$ , and the frequency of unattributed histograms published based on the CDP algorithm is closer to the baseline  $L$ . By observing that when the privacy budget  $\epsilon = 1$ , the CDP-L and the LP-L are both close to the baseline  $L$ . The reason is that when  $\epsilon = 1$ , there is less random noise added, which makes the frequency disturbance of the original histogram smaller. With the privacy budget  $\epsilon$  decreases, the distance of CDP-L and LP-L from the baseline  $L$

TABLE 6: Mean absolute error table.

Privacy budget	Algorithm	Campus	Entertainment	Family
$\epsilon = 1$	CDP	2.1547	2.1127	2.1518
	LP	3.9398	3.9252	3.9509
$\epsilon = 0.1$	CDP	21.3226	21.2638	21.1145
	LP	39.5529	39.8937	39.1742
$\epsilon = 0.01$	CDP	214.8347	211.2819	212.6625
	LP	391.8511	396.3250	388.6677

increases. When the privacy budget is 0.1 and 0.01, the noise added to the original frequency gradually increases, causing the deviation between the query result after privacy protection and the baseline  $L$  to gradually increase. As the privacy budget decreases, the histogram frequency published by the CDP algorithm is closer to the baseline  $L$  than the LP algorithm.

*(2) Analysis of the CDP Algorithm.* In this experiment, the value of the privacy budget  $\epsilon$  is still 1, 0.1, and 0.01. Then, take the mean absolute error of 100 random queries under different privacy budgets and calculate the average value after repeating the experiment 50 times. Finally, we obtain the mean absolute error table and histogram shown in Table 5 and Figure 10.

Figures 10(a)–10(c), respectively, show the mean absolute error of the CDP algorithm and the LP algorithm under different privacy budgets for campus, entertainment, and family. From the analysis of the mean absolute error results in Table 5, there are two reasons for the closer error when the privacy budget  $\epsilon = 1$ . One is that the added noise is small, resulting in small frequency disturbance of the original histogram, and the other is that the frequency of the buckets in the original histogram is relatively close, and the number of buckets is small. From the mean absolute error and privacy budget, the noise error in the CDP algorithm is significantly smaller than the error in the LP algorithm with the privacy budget decreases. It shows that the unattributed histograms published by the CDP algorithm have more higher accuracy under the same privacy budget.

*(3) Analysis of the CDPR Algorithm.* In the experiment, we take the mean absolute error of 50 samples under the same query range to calculate the average, and the privacy budget is set to 1, 0.1, and 0.01.

Figures 11(a)–11(c), respectively, show the mean absolute error trend of the CDPR algorithm, MDHP algorithm, and LP algorithm under different privacy budgets for campus, entertainment, and family. As the query range increases under the same privacy budget, the CDPR algorithm has a lower error than the MDHP algorithm and the LP algorithm, which shows that the CDPR algorithm proposed in this paper satisfies the  $\epsilon$ -differential privacy and improves the data availability. By observing the average absolute error curves under three different privacy budgets, it is found that



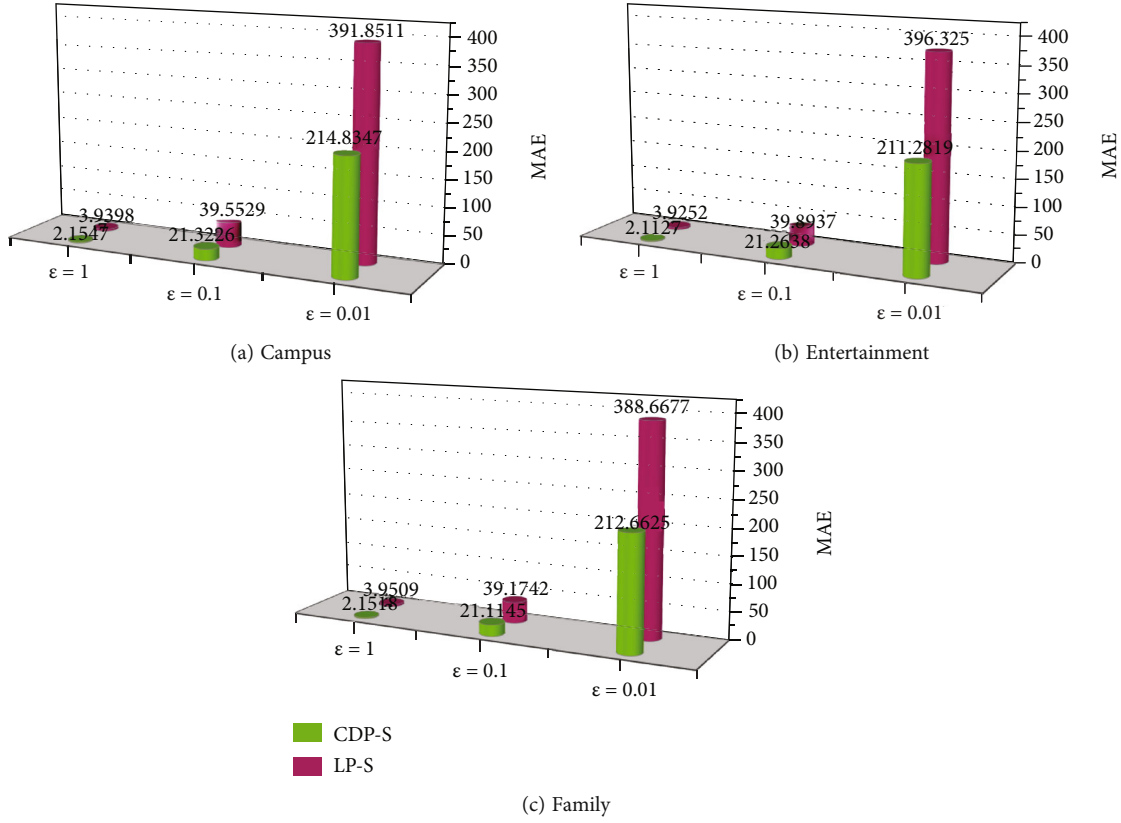


FIGURE 13: Mean absolute error under different privacy budgets.

when the privacy budget decreases, the random noise increases, which leads to the increase of the mean absolute error of the CDPR algorithm, MDHP algorithm, and LP algorithm. However, the mean absolute error of the CDPR algorithm is still smaller than the mean absolute error of the MDHP algorithm and the LP algorithm. When the privacy budget decreases or the number of queries increases, the CDPR algorithm not only satisfies the  $\epsilon$ -differential privacy but also publishes unattributed histograms with low error.

It can be observed from Figure 11 that there are two reasons why the CDPR algorithm proposed in this paper has no obvious advantages compared with the LP algorithm and the MDHP algorithm in a small range of queries. One reason is due to the frequency of the unattributed histogram experimental cases is similar, and the number of buckets is fewer, and when adding small random noise to the unattributed histogram, the frequency values of the unattributed histograms published by the CDP algorithm and the LP algorithm are similar. On the other hand, since the frequency of the unattributed histogram fluctuates greatly after noise is added, there may be cases of no merging in the grouping reconstruction stage, causing the results of the CDPR algorithm to be similar to the LP results. The last reason is that the frequency of adjacent buckets in the unattributed histogram published by the CDP algorithm is the same, which causes the CDPR algorithm to directly merge adjacent buckets with the same frequency.

#### 5.4.2. Universal Histograms

(1) *Analysis of the Query Results.* In this experiment, the universal histogram original frequency set  $S$  used as the baseline. The histogram frequency sets published by the CDP algorithm and the LP algorithm are defined as CDP-S and LP-S. The accuracy of histogram data published based on the CDP algorithm was verified by comparing the distance between CDP-S, LP-S, and the baseline  $S$  when the privacy budget  $\epsilon$  is 1, 0.1, and 0.01.

Figure 12 shows the results of frequency query under different privacy budgets. When the privacy budget  $\epsilon = 1$ , the noise added to the original frequency is small, making the frequency inferred by the CDP algorithm similar to the frequency published by the LP algorithm, so the distance advantage between CDP-S and baseline  $S$  is not apparent. With the privacy budget is set to 0.1 and 0.01, the noise content of the original frequency increases. The frequency set CDP-S is closer to the baseline  $S$  than the frequency set LP-S, which shows that the CDP algorithm has lower error than the LP algorithm.

(2) *Analysis of the CDP Algorithm.* According to the mean absolute error results in Table 6, we draw the mean absolute error distribution as shown in Figure 13. In this experiment, the privacy budget  $\epsilon$  is still selected as 1, 0.1, and 0.01, and take the average of the mean absolute error after 100 random queries for the count results of any interval. The whole

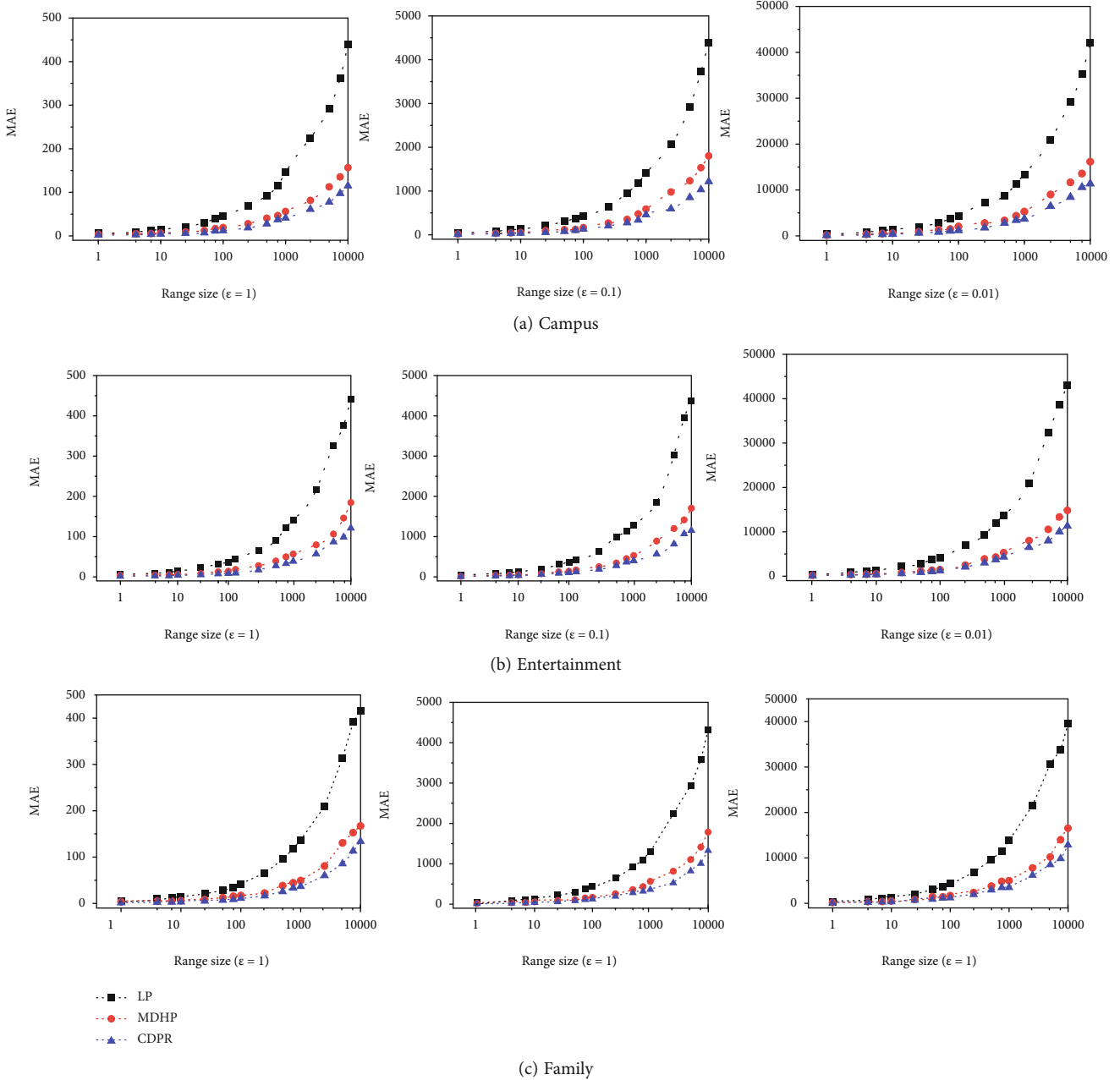


FIGURE 14: Mean absolute error of universal histograms in campus, entertainment, and family.

experiment was repeated 50 times, and the average value was taken.

According to the Table 6 and Figure 13, in the same microdomain, no matter the privacy budget is 1, 0.1, or 0.01, the average absolute error produced by the CDP algorithm is smaller than the average absolute error produced by the LP algorithm. However, since the random noise added by  $\epsilon = 1$  is small, and the original frequency value in the experimental case is small, resulting in a small frequency fluctuation range after adding noise, the mean absolute error of the CDP algorithm is close to that of the LP algorithm. When the privacy budget is 0.1 or 0.01, more noise will be

added to the original frequency. The mean absolute error of the CDP algorithm is significantly smaller than that of the LP algorithm. With the privacy budget decreases, the histogram published by the CDP algorithm has a lower error than the histogram published based on the LP algorithm.

(3) *Analysis of the CDPR Algorithm.* The long-range query frequency results of the general histogram are to calculate the mean absolute error of the CDPR algorithm, MDHP algorithm, and LP algorithm under different privacy budgets and compare and analyze the three algorithms. During the experiment, the interval size taken as  $2^h$ , where  $h$  is the tree's height. For the same interval size, we take the average of the

same number of times and repeat the entire experiment 50 times to get the average.

Observing Figures 14(a)–14(c), it is found that under the same privacy budget, as the query range increases, the mean absolute error curve of the CDPR algorithm is lower than the mean absolute error curve of the LP algorithm and the MDHP algorithm. When the privacy budget decreases, the mean absolute error curve of the CDPR algorithm is still lower than the mean absolute error curve of the LP algorithm and the MDHP algorithm. It shows that the universal histogram published based on the CDPR algorithm not only supports long-range queries, but also with the privacy budget decreases, the CDPR algorithm has lower error than the LP algorithm and the MDHP algorithm.

From the trend of the mean absolute error curve in Figure 14, it was found that the mean absolute error curve of the LP algorithm increases sharply with the increase of the query number. The reason is that due to the increase in the number of queries, too much noise is accumulated in the frequency of the unit interval, so more noise is accumulated when calculating the frequency of any interval. With the query range increases, the mean absolute error curve of the MDHP algorithm compared to the LP algorithm does not have a sharp increase significantly. Although the MDHP algorithm optimizes the noise error accumulated in the unit interval by merging adjacent buckets, the other arbitrary interval frequency is still calculated based on the unit interval frequency, so more errors are accumulated in the face of long-range queries or small privacy budgets. However, this paper replaces the original query sequence by building a full binary tree before adding noise, to avoid the noise accumulated in the unit interval from affecting the frequency of other arbitrary intervals.

(4) *CDPR Algorithm Time Complexity.* The CDPR algorithm is composed of constraint inference for differential privacy algorithm and group reconstruction based on constraint inference algorithm. In constraint inference for differential privacy algorithm, the time complexity of the differential privacy algorithm is  $O(n)$ . Then, execute the constraint inference algorithm, where the time complexity of the unattributed histogram positive sequence inference algorithm is  $O(n^2)$ , and the time complexity of the linear inference algorithm of the universal histogram is  $O(\log 2n)$ . In group reconstruction based on constraint inference algorithm, the time complexity of the histogram group reconstruction algorithm is  $O(n^2)$ .

In summary, the algorithm proposed in this paper has high time complexity, but the time complexity is within an acceptable range. At the same time, the method proposed in this paper can support long-range queries and improve the availability of data. The method proposed in this paper is suitable for small-scale, small-span, and small-change statistical data, such as government statistics, traffic statistics, and other related data. It is reasonable to improve the availability of data by sacrificing acceptable time complexity.

## 6. Conclusions

Due to the multidomain fusion data based on the collaborative collection of the IoT has rich background knowledge and attribute characteristics, it causes problems such as privacy leakage based on background knowledge and difficulty in multimodal data publishing. To address these problems, we propose a multidomain fusion data privacy security framework that includes three models such as the MRCG model, MHB model, and MHPP model. In the MRCG, first, we perform microdomain recognition through the proposed microdomain and microdomain attribute set definitions. Second, we use information entropy, conditional entropy, and information gain to realize the microdomain attribute set classification and grading. Finally, based on the results of MRCG, determine the attributes sensitivity and relationship between attributes to solve the problem of multimodal data statistics difficulties in multidomain fusion data sets. In the MHB, according to the results of multimodal data statistics, combine unattributed histogram and universal histogram to build multimodal histograms for multidomain fusion data set. In the MHPP, we improve the MDHP algorithm by positive-order inference and linear estimation to implement privacy protection for multimodal histogram data publishing.

Based on the real data set, the experimental results show that the multidomain fusion data privacy security framework not only realizes the recognition, classification, and grading of microdomain attribute set but also builds multimodal histograms by combining multimodal data statistics with the unattributed histogram and the universal histogram. Furthermore, the CDPR algorithm in the MHPP model ensures the privacy security of the published multimodal histogram data. Compared with the LP algorithm and the MDHP algorithm, the CDPR algorithm has a lower frequency query error and improves the data availability of the published histogram.

In the future, we will research the recognition, classification, and grading model for streamed data and the privacy protection model for dynamic data publishing.

## Data Availability

The data (multidomain fusion structured data) used to support the findings of this study are included within the article (“Using data mining to predict secondary school student performance”).

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This paper is supported by (1) the National Natural Science Foundation of China under Grant nos. 61672179 and 61370083, (2) the Project funded by China Postdoctoral Science Foundation under Grant no. 2019M651262, and (3) the Youth Fund Project of Humanities and Social Sciences

Research of the Ministry of Education of China under Grant no. 20YJCZH172.

## References

- [1] Y. Liu and T. Zhang, "Personal privacy protection in the era of big data," *Journal of Computer Research and Development*, vol. 52, no. 1, pp. 229–248, 2015.
- [2] T. Ya, Y. Lin, J. Wang, and J.-U. Kim, "Semi-supervised learning with generative adversarial networks on digital signal modulation classification," *CMC-Computers Materials & Continua*, vol. 55, no. 2, pp. 243–254, 2019.
- [3] J. Xiong, J. Ren, L. Chen et al., "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2019.
- [4] Q. Li, B. Xia, H. Huang, Y. Zhang, and T. Zhang, "TRAC: traceable and revocable access control scheme for mHealth in 5G-enabled IIoT," *IEEE Transactions on Industrial Informatics*, p. 1, 2021.
- [5] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: recent advances and future challenges," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 790–807, 2021.
- [6] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
- [7] J. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.
- [8] Y. C. Yang, L. F. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.
- [9] M. Ammar, G. Russello, and B. Crispo, "Internet of things: a survey on the security of IoT frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.
- [10] L. Sweeney, "K-anonymity: a model for protecting privacy," *International Journal on Uncertainty, Fuzziness and Knowledge Based Systems*, vol. 10, no. 5, pp. 557–570, 2002.
- [11] W. Q. Zhang, G. S. Yin, Y. Sha, and J. Yang, "Protecting the moving user's locations by combining differential privacy and -anonymity under temporal correlations in wireless networks," *Wireless Communications and Mobile Computing*, vol. 2021, 12 pages, 2021.
- [12] J. B. Xiong, M. F. Zhao, M. Z. A. Bhuiyan, L. Chen, and Y. Tian, "An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2021.
- [13] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2008.
- [14] K. El Emam and F. K. Dankar, "Protecting privacy using k-anonymity," *Journal of the American Medical Informatics Association*, vol. 15, no. 5, pp. 627–637, 2008.
- [15] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkatasubramanian, "l-diversity: privacy beyond k-anonymity," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pp. 24–35, Atlanta, Georgia, USA, 2006.
- [16] B. Zhou and J. Pei, "The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks," *Knowledge and Information Systems*, vol. 28, no. 1, pp. 47–77, 2011.
- [17] X. Zhang, "An accurate method for mining top-k frequent pattern under differential privacy," *Journal of Computer Research and Development*, vol. 51, no. 1, pp. 104–114, 2014.
- [18] C. Piao, Y. Shi, J. Yan, C. Zhang, and L. Liu, "Privacy-preserving governmental data publishing: a fog-computing-based differential privacy approach," *Future Generation Computer Systems*, vol. 90, pp. 158–174, 2019.
- [19] X. Yang and L. Gao, "Balanced correlation differential privacy protection method for histogram publishing," *Chinese Journal of Computers*, vol. 43, no. 8, pp. 1415–1432, 2020.
- [20] K. Li and C. Wang, "High-precision histogram publishing method based on differential privacy," *Journal of Computer Applications*, vol. 40, pp. 3242–3248, 2020.
- [21] Z. Chong and W. Ni, "A privacy-preserving data publishing algorithm for clustering application," *Journal of Computer Research and Development*, vol. 47, no. 12, pp. 2083–2089, 2010.
- [22] H. Li, J. T. Cui, X. Meng, and J. Ma, "IHP: improving the utility in differential private histogram publication," *Distributed and Parallel Databases*, vol. 37, no. 4, pp. 721–750, 2019.
- [23] S. Zhang, L. Liu, Z. Chen, and H. Zhong, "Probabilistic matrix factorization with personalized differential privacy," *Knowledge-Based Systems*, vol. 183, p. 104864, 2019.
- [24] D. Wu, Z. Yang, B. Yang, R. Wang, and P. Zhang, "From centralized management to edge collaboration: a privacy-preserving task assignment framework for mobile crowdsensing," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4579–4589, 2020.
- [25] W. He and C. Pen, "Sensitive attribute recognition and classification algorithm for structure dataset," *Application Research of Computers*, vol. 37, no. 10, pp. 3077–3082, 2020.
- [26] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proc. of the 2nd Int'l Conf. on Privacy Enhancing Technologies*, R. Dingledine and P. Syverson, Eds., pp. 54–68, Springer-Verlag, Berlin, Heidelberg, 2002.
- [27] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proc. of the 2nd Int'l Conf. on Privacy Enhancing Technologies*, R. Dingledine and P. Syverson, Eds., pp. 41–53, Springer-Verlag, Berlin, Heidelberg, 2002.
- [28] C. Pen, "Information entropy models and privacy metrics methods for privacy protection," *Journal of Software*, vol. 27, no. 8, pp. 1891–1903, 2016.
- [29] Y. Yihan, "Metric and classification model for privacy data based on Shannon information entropy and BP neural network," *Journal on Communications*, vol. 39, no. 12, pp. 11–17, 2018.
- [30] B. Krishnamurthy and C. E. Wills, "Characterizing privacy in online social networks," in *Proceedings of the First Workshop on Online Social Networks*, pp. 37–42, Seattle, WA, USA, 2008.
- [31] X. Yong and X. Qin, "A QI weight-aware approach to privacy preserving publishing data set," *Journal of Computer Research and Development*, vol. 49, no. 5, pp. 913–924, 2012.
- [32] N. Li and T. Li, "T-closeness: privacy beyond k-anonymity and l-diversity," in *Proceedings of the 23rd International*

- Conference on Data Engineering (ICDE)*, pp. 106–115, Istanbul, Turkey, 2007.
- [33] C. Dwork, “Differential privacy,” in *Proceedings of the 33rd International Colloquium on Automata Languages and Programming (ICALP)*, pp. 1–12, Venice, Italy, 2006.
  - [34] C. Dwork, “Differential privacy: a survey of results,” *Theory and Applications of Models of Computation Proceedings*, vol. 4978, pp. 1–19, 2008.
  - [35] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proceedings of the 3th Theory of Cryptography Conference (TCC)*, pp. 363–385, New York, USA, 2006.
  - [36] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett, “Differentially private histogram publication,” in *Proceedings of IEEE 28th International Conference on Data Engineering (ICDE)*, pp. 32–43, Washington, DC, USA, 2012.
  - [37] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett, “Differentially private histogram publication,” *International Journal of Very Large Database (VLDBJ)*, vol. 22, no. 6, pp. 797–822, 2013.
  - [38] X. Xiao, G. Wang, and J. Gehrke, “Differential privacy via wavelet transforms,” *IEEE Transactions on Knowledge Data Engineering (TKDE)*, vol. 23, no. 8, pp. 1200–1214, 2011.
  - [39] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, “Boosting the accuracy of differentially private histograms through consistency,” in *Proceedings of the 36th Conference of Very Large Databases (VLDB)*, pp. 1021–1032, Istanbul, Turkey, 2010.
  - [40] P. Cortez and A. Silva, “Using data mining to predict secondary school student performance,” in *Conference Committee of the ECEC-FUBUTEC’2008 Conference*, Porto, 2008.

## Research Article

# VarDefense: Variance-Based Defense against Poison Attack

Mingyuan Fan , Xue Du , Ximeng Liu , and Wenzhong Guo 

College of Computer and Data Science, Fuzhou University, 350108, China

Correspondence should be addressed to Wenzhong Guo; [guowenzhong@fzu.edu.cn](mailto:guowenzhong@fzu.edu.cn)

Received 4 August 2021; Revised 18 October 2021; Accepted 12 November 2021; Published 9 December 2021

Academic Editor: Lei Chen

Copyright © 2021 Mingyuan Fan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The emergence of poison attack brings a serious risk to deep neural networks (DNNs). Specifically, an adversary can poison the training dataset to train a backdoor model, which behaves fine on clean data but induces targeted misclassification on arbitrary data with the crafted trigger. However, previous defense methods have to purify the backdoor model with the compromising degradation of performance. In this paper, to relieve the problem, a novel defense method VarDefense is proposed, which leverages an effective metric, i.e., variance, and purifying strategy. In detail, variance is adopted to distinguish the bad neurons that play a core role in poison attack and then purifying the bad neurons. Moreover, we find that the bad neurons are generally located in the later layers of the backdoor model because the earlier layers only extract general features. Based on it, we design a proper purifying strategy where only later layers of the backdoor model are purified and in this way, the degradation of performance is greatly reduced, compared to previous defense methods. Extensive experiments show that the performance of VarDefense significantly surpasses state-of-the-art defense methods.

## 1. Introduction

Recently, deep neural networks (DNNs) have obtained impressing achievements over a broad spectrum of domains, such as image recognition [1], natural language process [2], recommendation system [3], and others [4–6]. Unfortunately, when DNNs are used in security-critical or data-sensitive scenarios, potential benefits arouse the adversaries to attack the models.

Over the past decade, many attack methods have been developed. For example, model extraction attacks [7] expose the valuable information of the model, model inversion attacks [8] reconstruct the secret data, and evasion attack [9] makes the model output false predicts. Despite the fact that those attacks are dangerous, poison attack still is the most threatening attack due to its high attack feasibility compared to those attacks. Poison attack can be effortlessly conducted by mixing a small volume of poison data into the training dataset, and a backdoor will be planted in any model trained with the poison dataset. Subsequently, this backdoor is activated during model inference by a trigger crafted by the attacker which, whenever and wherever present in a given image, enforces the DNN to predict the poison

label. The capacity of manipulating the output of the DNN can produce a severe consequence in many fields. For instance, it is extremely terrible that the autopilot system [10] mistakenly recognizes the pedestrian as something else and similar events happen in face recognition [11], etc. In short, poison attack makes DNNs unreliable and greatly drags the application of DNNs in the physical world.

For promoting applications of DNNs in scenarios with high-security requirements, several relatively effective defense approaches have been proposed to conquer poison attack over the past years. These defense methods generally are two-stage methods, including the detection stage and purifying stage. The detection stage is aimed at determining whether a DNN contains a backdoor. In detail, optimization is adopted to derive a trigger for each class and then inspecting if there is a trigger with exceptional distinctness than other triggers. The existence of a trigger with exceptional distinctness probably suggests that the model is poisoned. By adopting the idea, the detection stage obtains an impressive capacity of detecting the backdoor. In contrast, the performance of such methods focused on purifying stage is disappointing. Specifically, with such methods, despite the backdoor being fairly erased from the model, the induced

loss in performance of the model is always compromising, extremely lowering the effectiveness of such approaches. This paper proposes a novel defense method, dubbed variance-based defense against poison attack (VarDefense), to reduce the impairment induced by removing the backdoor from the model.

VarDefense is motivated by the fact that the backdoor embedded in the model essentially is the neurons responding strongly to the trigger (named bad neurons), which contribute the most of the attack effectiveness. Hence, we desire to identify and then remove the bad neurons, and in this way, the performance of the model can be reserved as possible while the backdoor is quite removed. However, the information about the trigger pattern does not access in advance, and then, it is intractable to recognize the bad neurons directly. To circumvent the problem, we embrace the approach with a similar function that reserves the neurons that produce a great effect for clean data (named benign neurons). Moreover, in Approach, it is intuitively justified that the way can achieve the identical performance to the method of directly identifying the poison neurons. Extensive experiments show that the performance of VarDefense significantly surpasses state-of-the-art defense methods. Our contributions are summarized as follows:

- (i) We suggest adopting variance as the purifying metric, which is more efficient compared to previous purifying metrics. Then, based on the variance, we propose a novel defense method VarDefense against the poison attack
- (ii) We propose an insightful perspective to understand the mechanism of the poison attack and design a simple yet efficient purifying strategy that can greatly reduce the degradation of the model's performance
- (iii) We conduct extensive experiments on four widely used benchmark datasets, i.e., MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100, to validate the effectiveness of VarDefense, where the experiment results demonstrate the superior performance of VarDefense than state-of-the-art defense methods

## 2. Related Works

**2.1. Backdoor Attack.** In the life cycle of DNNs, there are many potential threat factors used to attack, and poison attack is conducted during the training process. The seminal work of poison attack was proposed by Gu et al. [12], where authors just added a simple trigger pattern, e.g., a small white rect, into a small portion of training data, and then a backdoor is strongly injected into the model. Specifically, the model trained with the poison dataset classified poison input into a poison label over 99% success rate, without degradation of performance. Following Gu et al. [12], Chen et al. [13] argued that, for a practical attack, the images with the crafted trigger should be indistinguishable compared to its benign version. To do so, Chen et al. [13] proposed a

blended strategy in which poison inputs are produced by blending the specific random noise with clean images. It greatly decreases the risk of being checked out by humans. Xie et al. [14] and Bhagoji et al. [15] extend the poison attack into the federated learning scenario, which further increases the threat of poison attack.

**2.2. Backdoor Defense.** With the emergence of poison attacks, several countermeasures were proposed, and such countermeasures can be roughly classified as detection methods and purifying methods.

Detection methods focus on recognizing whether a model is poisoned. Neural cleanse [16] has taken the first step for detecting whether a model contains a backdoor. In detail, the possible triggers for each class are reversed and then  $L_1$  norm as the metric to inspect the trigger with exceptional features. The model is infected if there is a trigger with exceptional features. Despite the impressive performance of neural cleanse, it still requires a clean dataset. To further alleviate the constraint, Chen et al. [17] proposed DeepInspect to detect the backdoor without resort to a clean dataset, which obtains competitive performance compared to neural cleanse. There are some ambiguous approaches that are aimed at detecting the poison examples, and we also classify the approaches into the detection methods. For example, Gao et al. [18] proposed to filter poison examples through overlapping example patterns and inspect the output. If the randomness of the output is too small, the examples probably are poisoned.

In contrast to detection methods, purifying methods are aimed at removing the backdoor while maintaining the performance as possible. Liu et al. [19] proposed a simple way against poison attack, where the model is fine-tuning with benign examples. But the performance of the method is fairly low. Similar to it, Qiao et al. [20] proposed fine-tuning the model with the mixed dataset, which consisted of the clean data and the poison data (synthesized by the generative model) with the clean label other than the poison label. To remove the backdoor, Liu et al. [21] proposed fine-pruning, a model compression method. Fine-pruning is done by removing the neurons with low activation value w.r.t. clean data, but some bad neurons remain in the model. Besides, based on the idea that the model is not poisoned, definitely, if it is trained from scratch over clean data, Yoshida and Fujino [22] proposed knowledge distillation for removing backdoor, where clean labels can be predicted by the teacher (poison) model. Furthermore, Li et al. [23] proposed neural attention distillation to improve the effectiveness of vanilla knowledge distillation, becoming the state-of-the-art purifying method currently. However, the effectiveness of such methods heavily depends on the number of clean data and performs unstably over different datasets.

## 3. Scenario

This section gives an overview of the attack and defense scenario, beginning with attack scenario, followed by defense assumptions and goals.

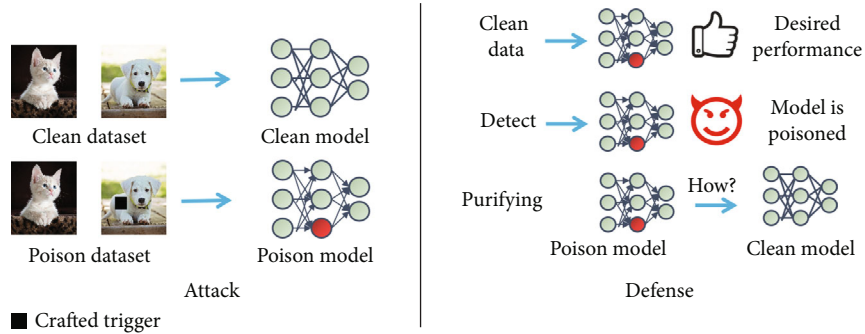


FIGURE 1: The illustration of the attack and defense scenario.

**3.1. Attack Scenario.** We assume that attackers poison the training dataset in some certain ways, and then, the model is trained with the poison dataset, as shown in Figure 1(a). Then, the (poisoned) model performs well on clean data but exhibits targeted misclassification on the presence of poison data, i.e., a backdoor is injected into the model. This attack scenario is fairly common in practice because the training dataset is always not fully controlled by users. To illustrate, there are three typical cases. The first case is that the collected dataset may contain poison data crafted and spread by the attackers. Another case is that, in federated learning, the malicious participants are desired to poison their own dataset to manipulate the shared model. The third case is that, due to deficiency of computing resource, the users outsource the training process to an untrusted party and the party probably poisons the dataset to potential benefits. In short, the assumed attack scenario covers most of the attack scenario of the poison attack.

**3.2. Defense Scenario.** Now, the defenders (users) receive the (poisoned) model and there are two required tasks for defenders: validating the performance and detecting whether the model is poisoned. To validate the performance of the model, it is commonly supposed that the defenders have a small volume of clean data, i.e., test set. Moreover, we highlight that the test set is identically distributed with the test size, which is a widely used assumption in most backdoor defense papers [19–23] because it is impossible to adopt a test set outside the distribution of training set to measure the performance of the model. Then, as demonstrated in Figure 1(b), the defender attempt begins to validate the performance and the model produces desired performance on the dataset available to the defenders. Besides, we assume that the defenders has been ensured that a backdoor is embedded into the model (the step can be smoothly implemented with previous detection methods), and the remaining problem is how to purify the model. Specifically, the purifying method should fulfill two goals: (1) rendering the backdoor disable with slight performance loss and (2) without resort any extra resources except the holding dataset. In the next section, we will develop a method to meet the two goals.

**3.3. Approach.** In this section, we design the purifying approach VarDefense to address the poison attack, starting

with the core idea of VarDefense; implementation details are revealed subsequently.

**3.4. Overview.** The backdoor embedded in the model essentially is the bad neurons that are implicitly coopted by the attack to recognize the crafted trigger. According to the idea, removing the backdoor from the model basically equals removing those bad neurons. Besides, the bad neurons are dormant in the presence of clean data, suggesting that recognizing the bad neurons can be realized by using clean data. Therefore, we assume that the defenders possess a small volume of clean data, which also is a widely used hypothesis in most defense scenarios as forementioned in Scenario. The leaving problem is how to implement the idea leveraging the clean data. Intuitively, the idea can be implemented as follows: (1) feeding a batch of clean data into the network, (2) adopting a certain predefined selection metric to measure the importance of neurons to the data, and (3) based on it, determining the neurons that should be purified, i.e., recognizing the bad neurons. In the next paragraphs, we first define the selection metric and then introduce the purifying strategy and implement details.

**3.5. Variance-Based Selection Metric.** In general, there are two types of selection metrics that are available to evaluate the importance of neurons: weight-based metrics and output-based metrics. Weight-based metrics utilize the weights of neurons for assessing the importance of neurons, while output-based metrics adopt the outputs of neurons to evaluate the importance of neurons. However, weight-based metrics are unsuitable to our method. Specifically, notice that the bad neurons must not be associated with small weights, which implies only removing the neurons with small weight being not enough to erase the backdoor.

In contrast to weight-based metrics, output-based metrics are favorable. In fact, the neurons of the outputs with the relatively small change to the deviation of the clean data generally contribute less in terms of clean data, since the outputs are approximately constant to the variation in the inputs, suggesting the neurons are probably redundant, i.e., the bad neurons. Hence, such metrics based on sensitivity can be exploited to distinguish the beneficial neurons from the bad neurons. There are two such representative metrics: variance and information entropy. Variance assesses how divergent a set of contiguous numbers is while information



entropy measures the redundancy of a set of numbers. But, when vanilla information entropy is adopted to measure the divergence of contiguous data, discretization is the necessary data preprocessing procedure. Discretization involves how to bin that is not being well-handled till now. In sharp contrast, variance can inherently estimate the divergence of the contiguous variables without resort to any preprocessing trick. Moreover, the computation overhead required to variance is significantly lower than information entropy, considering the complicated process flow and log function being the high computation cost involved in information entropy. Therefore, variance is a better metric than information entropy.

### 3.6. Purifying Strategy and Implementation Details

**3.6.1. Purifying Strategy.** Recall that our goal is removing the backdoor while maintaining the performance as possible, and to fulfill the goal as possible, we have to bespeak a proper purifying strategy. The purifying strategy can be divided based on four factors: (1) the number of purifying layers, (2) purifying earlier layers or later layers, and (3) layer-wise purifying or network-wise purifying. It should be clarified that layer-wise purifying and network-wise purifying differ in purifying the neurons with the metric below the predetermined threshold within one certain layer or the network. For VarDefense, purifying multiple and later layers, and layer-wise purifying are shown in Figure 2. There are reasons for it. A piece of shared knowledge is that shallow and deep layers of DNNs are responsible for recognizing general and specific features, respectively. Besides, the trigger is a specific feature, which implies the backdoor is the neurons located in later layers. Therefore, purifying the later layers is enough for fairly removing the backdoor. Another benefit is to avoid the huge damage to the model induced by pruning earlier layers (all of the general features matter). For the third question, layer-wise purifying is advisable because the weights and outputs of neurons in different layers generally have different orders of magnitude. In other words, network-wise purifying is prone to purify the neurons in the layers with higher output magnitude.

**3.6.2. Implementation Details.** Now, the remaining problem is to determine the threshold for classifying neurons and how to purify the neurons below the threshold. We begin with solving the first problem. Intuitively, the fixed threshold strategy is not feasible since it is difficult to ensure a specified threshold where the backdoor is completely removed with desired loss of performance of the model. To conquer the dilemma, the dynamic incremental threshold strategy is introduced in VarDefense: initializing the threshold from a small value and iteratively raising it, until the threshold meets the predetermined maximum value. To overcome the second problem, we design an appropriate loss function involving crossentropy loss and  $L_1$  regularization term-associated weights of neurons for purifying the bad neurons. On the one hand, the regularization term is adopted to remove the backdoor fairly. On the other hand, the crossentropy loss function is aimed at fine-tuning for decreasing the

damage induced by purifying, i.e., avoiding compromising performance loss. Specifically, the loss function is defined as follows:

$$L = \frac{1}{n} \sum_{i=1}^n \left( \alpha \cdot \text{Cross}(y_i, F(x_i)) + \sum_{j \in \mathcal{M}} \|w^j\|_1 \right), \quad (1)$$

where  $\alpha$  is the hyperparameter that balances two loss terms,  $\mathcal{M}$  is purifying layers set, and  $w^j$  denotes all parameters of the  $j$ th layer.

**3.6.3. Momentum Trick.** The effectiveness of VarDefense heavily depends on the quality of clean data. In fact, to accurately evaluate the effect of a neuron for clean data, the importance of the neuron should be measured by all clean data simultaneously. But the solution has to consume a huge cost and requires advanced equipment, which is a harsh condition to normal users. To overcome the issue, we utilize the widely used momentum trick. In detail, by accumulating the previous values, the momentum trick can effectively reduce the bias of minibatch data, greatly improving the effectiveness of the purifying process.

## 4. Experiments

### 4.1. Experiment Setting

**4.1.1. Attacks.** Poison attacks mainly differ in the characteristic of the crafted trigger, e.g., size and position. To comprehensively evaluate the effectiveness of VarDefense to various poison attacks, both the size and position of the crafted trigger are covered in the experiments. For trigger size, we consider the four classic forms, including  $3 \times 3$ ,  $5 \times 5$ ,  $7 \times 7$ , and global noise. Meanwhile, for trigger position, the fixed position and random position are involved. Finally, the poison ratio is set to widely used 0.05, which is the fraction of poison data added per minibatch.

**4.1.2. Baselines.** To validate the performance of VarDefense, three mainstream types of state-of-the-art defense method is considered, including fine-pruning [21], GAN-based defense (the authors did not name their defense method, and thus, we use GAN-based defense to denote it) [20], and neural attention distillation (NAD) [23].

**4.1.3. Datasets and Model Architecture.** The performance of defense methods against attacks is measured in four benchmark datasets, i.e., MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100, while ResNet18 is the base model throughout the experiments.

**4.1.4. Evaluation Metric.** Followed by previous studies [20, 21, 23], we adopt the two widely used metrics to evaluate the effectiveness of VarDefense, i.e., accuracy (ACC) and attack success rate (ASR). ACC refers to the accuracy of the model to clean data, measuring the degradation of the performance of the model. ASR is the portion of poison data classified into the poison label, evaluating the attack effectiveness. In short, the higher the ACC is, the lower the ASR is, and the better the defense method is.

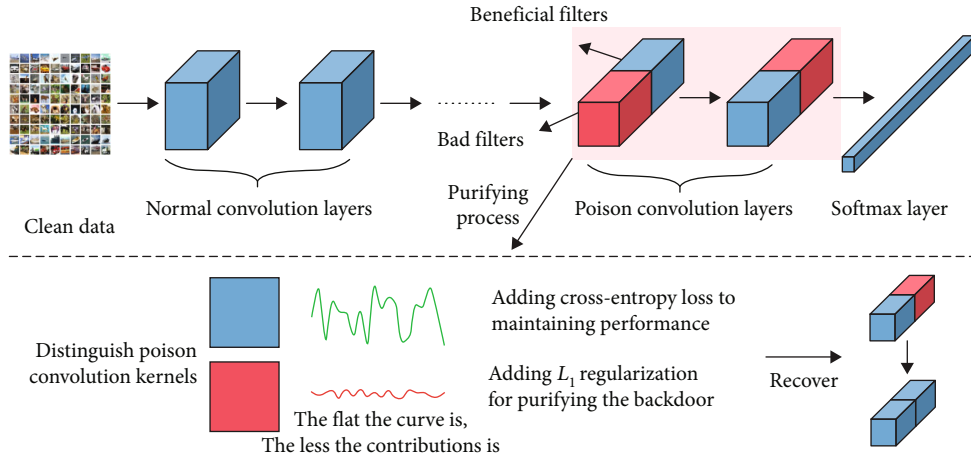


FIGURE 2: The running process of VarDefense. The images (clean data) are extracted from public dataset CIFAR-10 that can be referred to <http://www.cs.toronto.edu/~kriz/cifar.html>.

TABLE 1: The performance of four defense methods against different attack settings in MNIST and Fashion-MNIST. The numbers in *italic* are the best result.

Dataset	Trigger size	Position	Before		Fine-pruning		NAD		GAN-based defense		VarDefense		
			ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	
MNIST	$3 \times 3$	Fixed	99.05	98.67	11.47	96.70	1.86	98.24	7.48	97.59	<i>0.58</i>	<i>97.25</i>	
		Random	99.52	98.87	11.53	96.42	2.14	98.05	7.36	98.00	<i>0.54</i>	<i>98.52</i>	
	$5 \times 5$	Fixed	99.24	98.74	10.95	97.52	2.43	97.74	6.79	96.08	<i>0.57</i>	<i>97.91</i>	
		Random	99.53	98.59	11.02	97.33	3.11	97.13	6.66	96.31	<i>1.10</i>	<i>97.16</i>	
	$7 \times 7$	Fixed	99.17	99.72	13.70	96.94	2.26	97.36	11.75	95.54	<i>0.71</i>	<i>98.06</i>	
		Random	99.87	99.00	13.98	97.42	2.64	97.35	11.32	95.01	<i>0.80</i>	<i>97.60</i>	
	Global noise	—	98.75	98.83	10.33	97.19	2.65	98.12	7.13	95.91	<i>0.93</i>	<i>97.79</i>	
Fashion-MNIST	$3 \times 3$	Fixed	99.57	99.02	16.86	94.32	2.81	97.77	11.65	96.41	<i>0.57</i>	<i>98.30</i>	
		Random	99.86	99.25	16.42	95.12	2.23	97.61	11.32	96.31	<i>1.28</i>	<i>98.08</i>	
	$5 \times 5$	Fixed	99.68	99.23	14.25	95.24	1.95	97.60	12.40	95.80	<i>0.33</i>	<i>98.19</i>	
		Random	99.14	99.49	15.10	94.67	<i>1.69</i>	97.06	12.35	96.02	1.95	<i>98.97</i>	
	$7 \times 7$	Fixed	98.94	99.83	17.85	94.30	3.53	95.78	14.32	95.63	<i>0.92</i>	<i>98.46</i>	
		Random	99.08	99.15	18.32	94.51	3.51	96.34	13.57	96.52	<i>0.41</i>	<i>98.51</i>	
		Global noise	—	99.12	99.23	18.79	94.48	4.14	96.70	13.07	97.06	<i>1.14</i>	<i>98.36</i>

4.1.5. *Others.* Following the standard setup, we used SGD with a learning rate of 0.01 and batch size 128. All experiments were conducted in one machine with Ubuntu 18.04 system equipped with two Tesla V100s.

4.2. *Comparison with State-of-the-Arts.* Tables 1 and 2 summarizes the detailed performance of four defense methods over attacks with varying trigger sizes and positions on four datasets (CIFAR-10, CIFAR-100, MNIST, and Fashion-MNIST).

Overall, VarDefense dominates the previous state-of-the-art defense approaches in most settings (demonstrated in Tables 1 and 2). For instance, for VarDefense, ASR drops dramatically by more than 99% with negligible degradation of ACC (less than 1%) in MNIST, while other defense

methods, e.g., fine-pruning and GAN, only obtain about 90% drop, implying the superior performance of VarDefense compared to other methods. Besides, it is illustrated in Tables 1 and 2 that either of ACC and ASR to NAD is lightly better than VarDefense ( $\approx 1\%$  in such cases). But, it is highlighted that VarDefense still achieves better performance compared to NAD. In such cases, compared to VarDefense, NAD only obtains fairly minimal improvement ( $\approx 1\%$  in either of ACC or ASR, whereas VarDefense attains notable improvement ( $\approx 10\%$  on ACC and  $\approx 2\%$  on ASR) in another metric compared to NAD.

As introduced earlier, trigger is the most critical factor in poison attack, and hence, we investigate the effectiveness of VarDefense to varying triggers as demonstrated in Figure 3. The surprising aspect is that the lines in Figure 3

TABLE 2: The performance of four defense methods against different attack settings in CIFAR-10 and CIFAR-100.

Dataset	Trigger size	Position	Before		Fine-pruning		NAD		GAN-based defense		VarDefense	
			ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC
CIFAR-10	3 × 3	Fixed	98.36	83.33	32.51	78.54	11.65	69.48	10.21	78.97	<b>8.46</b>	<b>80.41</b>
		Random	98.80	83.81	32.07	77.96	11.24	69.00	9.52	78.95	<b>8.70</b>	<b>80.34</b>
	5 × 5	Fixed	97.28	83.28	37.25	78.33	<b>7.98</b>	69.58	9.29	78.46	9.10	<b>80.27</b>
		Random	97.99	83.64	37.41	78.34	<b>7.68</b>	69.25	9.32	78.51	8.83	<b>79.88</b>
	7 × 7	Fixed	98.93	82.96	35.15	76.32	8.92	70.63	10.79	78.30	<b>8.15</b>	<b>80.44</b>
		Random	98.01	82.89	35.24	76.29	8.95	70.49	10.77	78.32	<b>8.68</b>	<b>81.02</b>
	Global noise	—	98.08	84.27	33.08	78.76	12.33	70.00	10.42	78.58	<b>8.01</b>	<b>79.83</b>
CIFAR-100	3 × 3	Fixed	97.77	54.98	47.06	47.69	2.56	33.51	19.88	46.93	<b>2.14</b>	<b>46.23</b>
		Random	98.06	54.40	47.58	48.12	3.09	33.33	20.19	47.37	<b>2.75</b>	<b>46.45</b>
	5 × 5	Fixed	97.65	54.87	51.07	44.24	4.75	31.95	13.84	46.91	<b>1.93</b>	<b>46.83</b>
		Random	98.44	54.79	51.28	44.64	4.56	31.89	13.75	46.98	<b>1.78</b>	<b>46.45</b>
	7 × 7	Fixed	97.60	55.65	46.93	43.51	5.36	34.52	18.13	46.05	<b>2.27</b>	<b>47.83</b>
		Random	97.94	55.17	46.31	43.57	4.73	34.27	17.71	46.11	<b>1.81</b>	<b>46.98</b>
	Global noise	—	97.38	54.34	47.79	48.80	<b>2.60</b>	32.94	20.24	47.88	2.66	<b>46.02</b>

The numbers in bold mean that this method is the best compared with other methods, under the corresponding experiment settings.

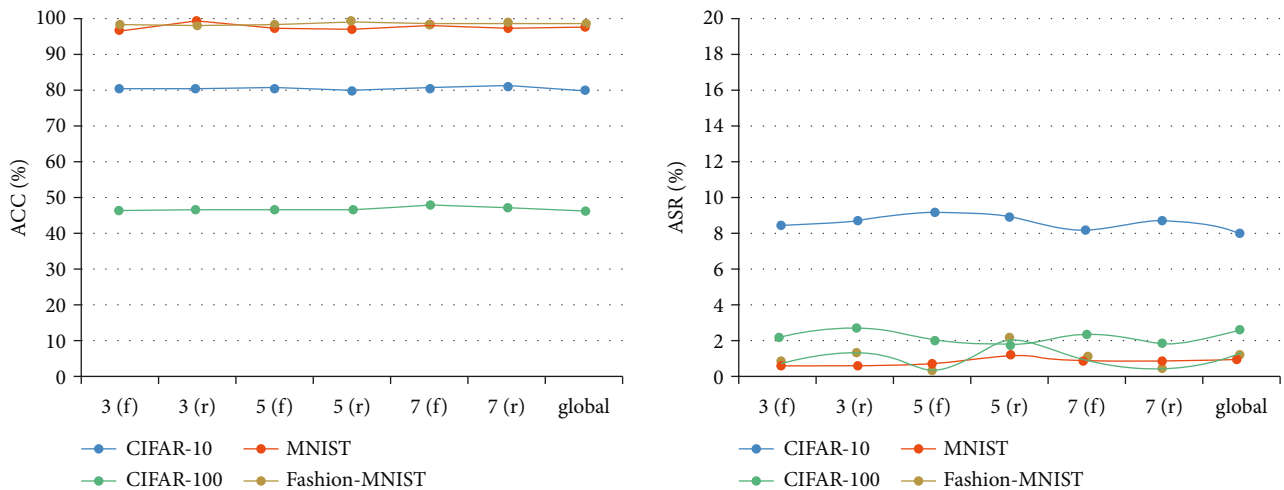


FIGURE 3: The performance of VarDefense over various trigger settings in four datasets. The f and r in this figure denote the fixed and random position while 3, 5, and 7 denote the trigger with 3 × 3, 5 × 5, and 7 × 7 resolution, respectively. The global means that the trigger is global imperceptible noise.

are fairly flattened, which suggests excellent and consistent resistance of VarDefense over various triggers.

In practice, the stability of performance over different tasks (i.e., datasets) is crucial and Figure 4 shows the average performance of different defense methods over various datasets. For ACC, we observed that VarDefense consistently performs well over four datasets, and the NAD performs unsteadily over four datasets. Meanwhile, for ASR, both fine-pruning and GAN have quite a few fluctuations over four datasets, and VarDefense still consistently works well. In a nutshell, the consistent and great performance of VarDefense over various datasets is more favored in practice.

#### 4.3. Further Exploration to VarDefense

**4.3.1. Impact of  $\alpha$ .** We are also interested in hyperparameter  $\alpha$  that controls the purifying magnitude, and Table 3 demonstrates the results of coarse tuning  $\alpha$  in four benchmark datasets.

It is noticed that lower  $\alpha$  induces lower ASR, i.e., more effective against poison attack, but, lower  $\alpha$  also results in bigger degradation of ACC. For instance, in CIFAR-10, when  $\alpha = 0.0001$ , despite the fact that the ASR drops to 4.5%, the ACC also degrade minimum (76.47%) and such cases also happen in MNIST, Fashion-MNIST, and CIFAR-100. In

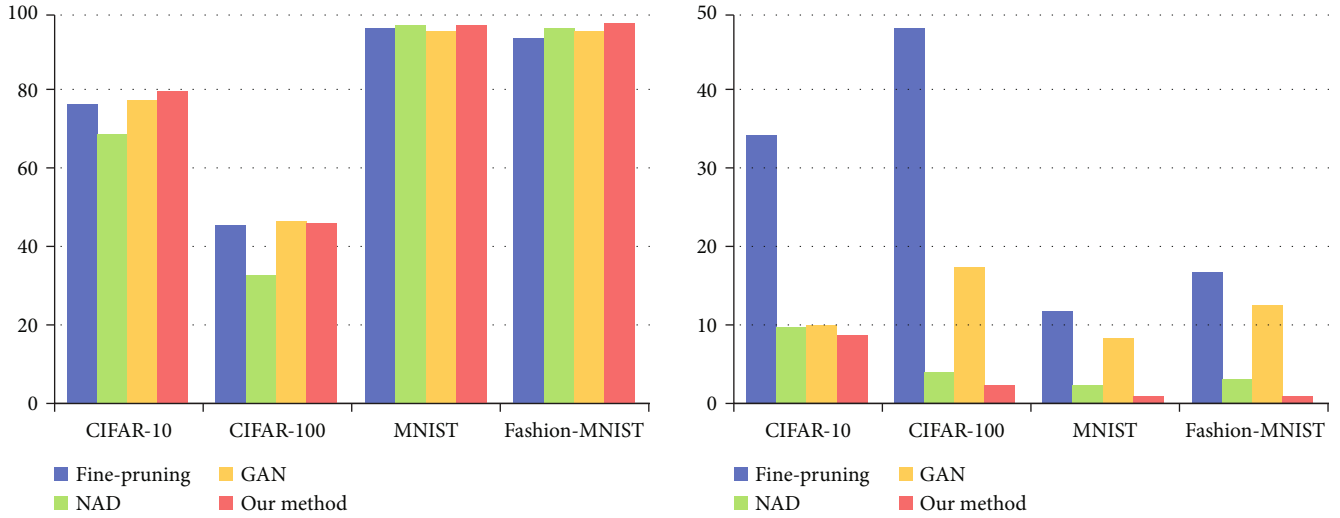


FIGURE 4: Average performance of four defense methods on CIFAR-10, CIFAR-100, MNIST, and Fashion-MNIST.

TABLE 3: The performance of VarDefense over different  $\alpha$ .

$\alpha$	MNIST		Fashion-MNIST		CIFAR-10		CIFAR-100	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
0.0001	95.96	<b>0.22</b>	95.96	<b>0.19</b>	76.47	<b>4.50</b>	45.96	<b>0.50</b>
0.001	96.15	0.28	96.15	0.21	80.25	4.90	46.04	0.94
0.01	97.37	0.28	96.37	0.22	80.27	5.80	46.05	1.01
0.1	97.85	0.39	96.85	0.38	80.29	7.59	46.05	1.19
1	98.30	0.57	97.25	0.58	80.41	8.46	46.23	2.14
10	97.43	0.59	97.47	0.56	80.97	12.50	46.29	3.03
100	97.52	0.63	97.79	0.67	81.03	15.53	46.80	3.73
1000	97.95	0.69	97.88	0.63	81.04	16.85	46.92	5.07
10000	98.06	0.77	98.02	0.83	81.59	17.50	47.97	5.65
100000	<b>98.19</b>	1.92	<b>98.07</b>	1.87	<b>81.72</b>	19.41	<b>48.27</b>	5.84

The numbers in bold mean that this method is the best compared with other methods, under the corresponding experiment settings.

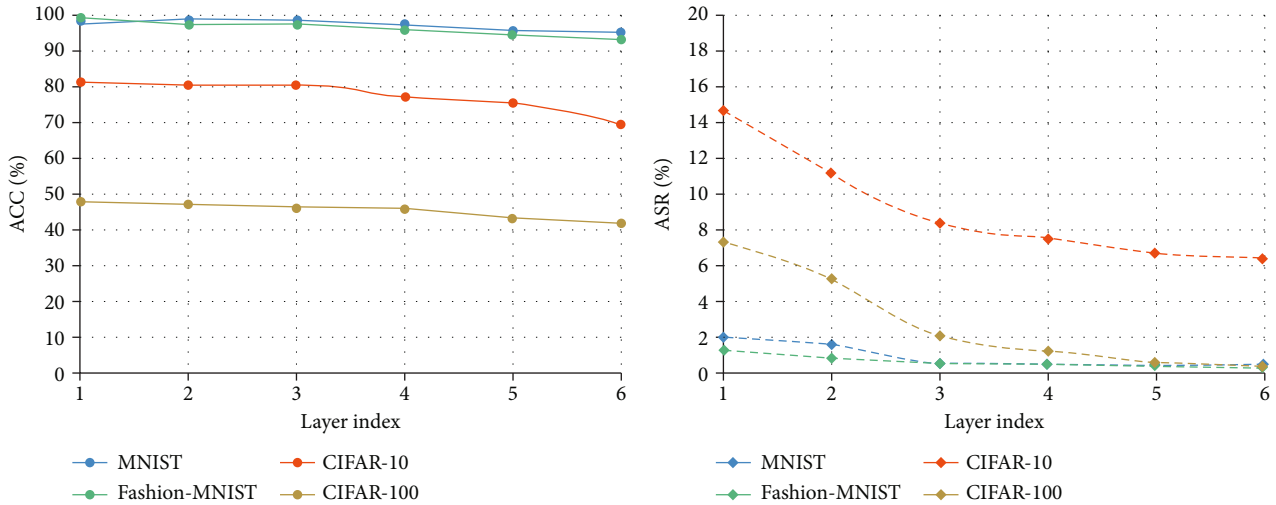


FIGURE 5: The performance of VarDefense with different purified layers.

short, there is a trade-off between maintaining model performance and removing the backdoor, and  $\alpha$  can control the trade-off. In practice, to search proper  $\alpha$ , it is suitable that  $\alpha$  iteratively increases from the small value until the ACC degrades below a predetermined threshold.

**4.3.2. Impact of Purifying Layers.** Selecting which layers to purify also is a critical factor, and Figure 5 presents the performance of VarDefense with purifying different layers. The layer index  $n$  denotes purifying the last  $2 \times n$  convolutional layers. We observe that, as the number of convolutional layers being purified (layer index) increases, the trend of performance change (ACC and ASR) is consistent with the inference as mentioned in Approach: purifying more layers induces lower ASR but with raising performance loss.

In conclusion, purifying layers play a similar role to  $\alpha$ , in which both of them adjust the balance between ACC and ASR. Therefore, the defenders should carefully tune the layer index and  $\alpha$  to achieve the desired trade-off between ACC and ASR.

## 5. Conclusions

In this paper, we point out the existence of bad neurons that causes the poison attack and identify removing the bad neurons as the main challenge in backdoor defense. Moreover, we design an effective defense method VarDefense which distinguishes the bad neurons based on variance and adopts a great purifying strategy. To solve unstable training of VarDefense, the momentum trick is introduced in VarDefense, remarkably increasing the efficiency of purifying. Extensive evaluations on four benchmark datasets show that the proposed defense method can reliably remove the backdoor with a slight performance loss compared to three state-of-the-art defense methods.

## Data Availability

The dataset of MNIST is available on the site <http://yann.lecun.com/exdb/mnist/>. The dataset of Fashion-MNIST is available on the site <https://github.com/zalando-research/fashion-mnist>. The dataset of CIFAR-10 and CIFAR-100 is available on the site <http://www.cs.toronto.edu/~kriz/cifar.html>.

## Conflicts of Interest

We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62072109, No. U1804263) and the Natural Science Foundation of Fujian Province (No. 2021J06013).



## References

- [1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: a survey," 2019, <http://arxiv.org/abs/1905.05055>.
- [2] A. Torfi, R. A. Shirvani, Y. Keneshloo, N. Tavaf, and E. A. Fox, "Natural language processing advancements by deep learning: a survey," 2020, <http://arxiv.org/abs/2003.01200>.
- [3] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 1–37, 2019.
- [4] J. Xiong, R. Bi, Y. Tian, X. Liu, and D. Wu, "Towards light-weight, privacy-preserving cooperative object classification for connected autonomous vehicles," *IEEE Internet of Things Journal*, pp. 1–15, 2021.
- [5] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
- [6] J. Xiong, M. Zhao, M. Bhuiyan, L. Chen, and Y. Tian, "An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2021.
- [7] X. Hu, L. Liang, L. Deng et al., "Neural network model extraction attacks in edge devices by hearing architectural hints," 2019, <http://arxiv.org/abs/1903.03916>.
- [8] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li, and D. Song, "The secret revealer: generative model-inversion attacks against deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 253–261, Seattle, WA, USA, 2020.
- [9] J. Zhang and C. Li, "Adversarial examples: opportunities and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 7, pp. 2578–2593, 2020.
- [10] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, Z. Zhong, and T. Wei, "Fooling detection alone is not enough: first adversarial attack against multiple object tracking," 2019, <http://arxiv.org/abs/1905.11026>.
- [11] K. Xu, G. Zhang, S. Liu et al., "Adversarial T-shirt! evading person detectors in a physical world," in *European Conference on Computer Vision*, pp. 665–681, Springer, Cham, 2020.
- [12] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.
- [13] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, <http://arxiv.org/abs/1712.05526>.
- [14] C. Xie, K. Huang, P. Y. Chen, and B. Li, "DBA: distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, New Orleans, LA, USA, 2019.
- [15] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International conference on machine learning. PMLR*, pp. 634–643, Long Beach, CA, USA, 2019.
- [16] B. Wang, Y. Yao, S. Shan et al., "Neural cleanse: identifying and mitigating backdoor attacks in neural networks," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, San Francisco, CA, USA, 2019.

- [17] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "DeepInspect: a black-box Trojan detection and mitigation framework for deep neural networks," *IJCAI*, pp. 4658–4664, 2019.
- [18] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: a defence against Trojan attacks on deep neural networks," in *Proceedings of the 35th annual computer security applications Conference*, pp. 113–125, San Juan, Puerto Rico, December 2019.
- [19] Y. Liu, Y. Xie, and A. Srivastava, "Neural Trojans," in *2017 IEEE international conference on computer design (ICCD)*, pp. 45–48, Boston, MA, USA, 2017.
- [20] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," 2019, <http://arxiv.org/abs/1910.04749>.
- [21] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: defending against backdooring attacks on deep neural networks," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 273–294, Springer, Cham, 2018.
- [22] K. Yoshida and T. Fujino, "Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks," in *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pp. 117–127, Virtual Event, USA, 2020.
- [23] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Neural attention distillation: erasing backdoor triggers from deep neural networks," 2021, <http://arxiv.org/abs/2101.05930>.
- [24] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xia, "Rethinking the trigger of backdoor attack," 2020, <http://arxiv.org/abs/2004.04692>.
- [25] B. Chen, W. Carvalho, N. Baracaldo et al., "Detecting backdoor attacks on deep neural networks by activation clustering," 2018, <http://arxiv.org/abs/1811.03728>.
- [26] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, <http://arxiv.org/abs/1503.02531>.
- [27] K. Yoshida and T. Fujino, "Countermeasure against backdoor attack on neural networks utilizing knowledge distillation," *Journal of Signal Processing*, vol. 24, no. 4, pp. 141–144, 2020.
- [28] L. Muñoz-González, B. Biggio, A. Demontis et al., "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 27–38, Dallas, TX, USA, 2017.
- [29] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, <http://arxiv.org/abs/1611.06440>.

## Research Article

# Enhancing Packet-Level Wi-Fi Device Authentication Protocol Leveraging Channel State Information

Yubo Song <sup>1,2</sup> Bing Chen,<sup>1,2,3</sup> Tianqi Wu,<sup>1,2</sup> Tianyu Zheng,<sup>1,2</sup> Hongyuan Chen <sup>1,2</sup>  
and Junbo Wang<sup>2,3</sup>

<sup>1</sup>Key Laboratory of Computer Network Technology of Jiangsu Province, School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China

<sup>2</sup>Purple Mountain Laboratories, Nanjing 211189, China

<sup>3</sup>School of Information Science and Engineering, Southeast University, Nanjing 211189, China

Correspondence should be addressed to Yubo Song; songyubo@seu.edu.cn

Received 10 May 2021; Accepted 18 October 2021; Published 17 November 2021

Academic Editor: Jinbo Xiong

Copyright © 2021 Yubo Song et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Wi-Fi device authentication is crucial for defending against impersonation attacks and information forgery attacks. Most of the existing authentication technologies rely on complex cryptographic algorithms. However, they cannot be supported well on the devices with limited hardware resources. A fine-grained device authentication technology based on channel state information (CSI) provides a noncryptographic method, which uses the CSI fingerprints for authentication since CSI can uniquely identify the devices. But long-term authentication based on CSI fingerprints is a challenging work. First, the CSI fingerprints are environment-sensitive, which means that the local authenticator should be updated to adapt to the changing channel state. Second, the local authenticator trained with old CSI fingerprints is outdated when users reconnect to the network after being offline for a long time, thus, it needs to be retrained in the access phase with new fingerprints. To tackle these challenges, we propose a CSI-based enhancing Wi-Fi device authentication protocol and an authentication framework. The protocol helps to collect new CSI fingerprints for authenticator's training in access phase and performs the fingerprints' dispersion analysis for authentication. In the association phase, it provides packet-level authentication and updates the authenticator with valid CSI fingerprints. The authenticator consists of an ensemble of small-scale autoencoders, which has high enough time efficiency for packet-level authentication and authenticator's update. Experiments show that the accuracy of the framework is up to 98.7%, and the authenticator updating method can help the framework maintains high accuracy.

## 1. Introduction

Nowadays, Wi-Fi has become one of the most important wireless associated technologies [1]. However, there are serious security issues with Wi-Fi networks. Attackers can obtain the identity information of a valid device through wireless sniffing and then use this information to disguise themselves as legitimate devices [2, 3]. Attackers can steal confidential data or attack the internal websites after getting authorization [4, 5], or they can control other devices by sending spurious instructions [6]. Due to the widespread use of Wi-Fi networks in a range of critical services such as financial transactions and business management, attackers based on the identity of Wi-Fi networks can cause damage

to public and private property and disrupt social order. Therefore, device authentication for Wi-Fi network security is indispensable [7, 8]. IEEE 802.11i provides cryptographic-based device authentication methods, but it has been proven to lack security [9–12]. What is more, Wi-Fi devices with limited hardware resources cannot support these authentication schemes well, which brings challenges to the usage of cryptographic-based device authentication technology.

In recent years, attempts have been made to use wireless channel characteristics for device authentication. One approach is to extract fine-grained fingerprints that can identify the device from the CSI and use fingerprint matching to authenticate the device [3, 13–16]. CSI refers to the channel frequency response (CFR) of a wireless multipath

channel, including the amplitude attenuation and phase shift of the channel. In a fast fading channel for indoor wireless communication, each radio path has different amplitude attenuation and time delay. As the radio signals arrive at the receiver along different paths, signals of different amplitudes and phases are superimposed to form the final signal, and the CSI reflects the amplitude fading and phase shift of the multipath channel. Because CSI reflects the amplitude and phase on all subcarriers, fine-grained device fingerprints can be generated with CSI. In a complex indoor environment, devices in different locations have different multipath channel states, and CSI is also expected to be different accordingly [17–23]. This feature allows CSI to be used for fingerprint extraction and device identification.

However, there are challenges to the application of this technology. It is well known that device authentication is usually a long-term task, and the channel state tends to change over time during the authentication process. As a result, the CSI fingerprint of the target device can also change, rendering the local authenticator ineffective. This will result in the rejection of legitimate devices. Therefore, it is significant to maintain a valid authenticator in an authenticated device during long-term operation for device authentication. More importantly, the authenticator often fails when a user logs back into the network after a long absence. This means that an alternative authentication method should be found to authenticate users at the access stage.

What is more, packet-level device authentication has high requirements on the computational complexity of the authentication algorithms. Packet-level device authentication requires extracting CSI fingerprints from each new data packet, data preprocessing, and fingerprint matching. Most of the CSI-based authentication technologies use large-scale neural networks such as ANN and CNN for fingerprint matching. Clustering algorithms such as  $K$ -means and SVM with high computational complexity are also used in authentication. Although these machine learning frameworks improve the accuracy of fingerprint matching, they are hard to apply for packet-level authentication in high throughput networks.

In this paper, we propose an enhanced Wi-Fi device authentication protocol based on CSI and authentication framework. We focus our attention on two working phases, including authentication in the access phase and authentication in the association phase. The protocol helps to collect CSI fingerprints in the access phase, and our framework performs fingerprint dispersion analysis for authentication. If the authentication in the access phase is successful, the newly collected CSI fingerprints are used to retrain the local authenticator. In the association phase, it provides packet-level authentication and updates the authenticator with valid CSI fingerprints. The local authenticator consists of a small-scale autoencoder rather than a neural network with high input dimensionality, which is computationally small enough for packet-level authentication and authenticator updates.

The main contributions of this paper are as follows:

- (i) We propose an enhancing Wi-Fi device authentication protocol based on CSI, including the authenti-

cation procedure in access phase and association phase. It combines our authentication technology with the Wi-Fi communication protocol so that it can be applied in the existing Wi-Fi networks

- (ii) We develop a Wi-Fi device authentication framework. In access phase, the framework is capable of the device authentication and the authenticator's retraining. In the association phase, it offers packet-level authentication service and updates the authenticator to make it fits the current channel state
- (iii) We present an authentication method based on the CSI measurements' dispersion degree in access phase. We also provide a novel machine learning architecture for packet-level authentication in association phase. It consists of an ensemble of small-scale autoencoders, which has high enough time efficiency and accuracy for authentication and updating authenticator at a packet level
- (iv) We implement our framework on Raspberry Pi and conduct real experiments in laboratory. Experimental results show that our framework can detect attackers and authenticate Wi-Fi devices with high accuracy under prolonged authentication. We also verify the effectiveness of the authenticator update method by conducting comparison tests

This paper is organized as follows: Section 2 shows the related work. Section 3 gives an overview of the authentication framework. Section 4 provides the authentication methods in access phase and association phase based on CSI fingerprints. Section 5 introduces the enhancing Wi-Fi device authentication protocol. Section 6 gives the evaluation. Section 7 concludes the paper.

## 2. Related Work

Recently, many wireless device authentication schemes based on CSI have been proposed. [17] applies  $K$ -means algorithm for authentication in access phase. It uses the number of clusters in the CSI fingerprints for judging whether there are fingerprints from attackers. [21] uses the correlation of neighbouring CSI fingerprints to determine whether the new CSI fingerprint comes from the valid device. [19] combined MIMO with CSI-based wireless device authentication technology, trying to use higher-dimensional CSI to improve authentication accuracy. Similarly, [18] also uses MIMO-CSI as the device fingerprint, but it proposes an authentication algorithm with a higher accuracy rate based on the LOF algorithm, which can achieve good accuracy even in an environment with low SNR.

Machine learning is widely used in wireless device authentication systems based on CSI. [13] uses CNN to classify new fingerprints and uses the classification results for subsequent device authentication. [17] uses  $K$ -means algorithm for access phase authentication. [24] leverages the SVM to obtain the similarity between the unknown CSI fingerprint and the local user profile, and the similarity is



used to determine whether the unknown fingerprint comes from a legitimate device. In addition, data dimensionality reduction algorithms are also used for the classification of high-dimensional data. Principal component analysis can extract the main feature components of high-dimensional data, thus, reducing the computational complexity of subsequent machine learning by downscaling the high-dimensional data to a low-dimensional space with the best linear combination.  $T$ -distributed stochastic neighbour embedding is also a dimensionality reduction algorithm. It can reduce the dimensionality of high-dimensional data to below 3 dimensions for data visualization.

However, all the abovementioned authentication systems do not provide a method for updating the authenticator during the authentication process. In this paper, we provide a method for updating the authenticator in real time, which can improve the performance of the authentication framework in the long run. In addition, none of the systems mentioned above show the time complexity of the authentication algorithms, and there is no guarantee that these techniques can be applied to high-throughput networks. However, the ensemble learning-based authenticator provided in this paper improves the time efficiency of authentication.

### 3. Framework Overview

We design a Wi-Fi device authentication framework that provides device authentication services in both the access phase and association phase, as shown in Figure 1. The framework can be implemented at the Wi-Fi access point (AP). Some work steps that require signaling interaction between the access point and the workstation (STA) can be accomplished by installing our authentication protocol on both devices, which will be shown in Section 5.

As shown in Figure 1, the Wi-Fi device authentication framework consists of two main parts, including authentication in the access phase and authentication in the association phase. The authenticator in between plays an important role in these two parts. The authentication framework will be described in terms of the device authentication workflow in the access phase and association phase as follows.

**3.1. Authentication Workflow in Access Phase.** According to the access process of the Wi-Fi device, STA needs to send an authentication request to AP. AP starts the CSI collection after receiving the request by sending a collection request (named collection REQ) to STA. STA then returns  $M$  measuring packets, which are used to extract CSI fingerprints. After extracting  $M$  CSI fingerprints, AP analyzes the dispersion degree of the fingerprints and judges whether the fingerprints are valid. If the fingerprints are invalid, the access request of STA will be rejected. Otherwise, the fingerprints are used to train the local authenticator for packet-level authentication in association phase. Before training the authenticator, the fingerprints should be preprocessed for noise reduction, which includes outlier elimination and smoothing. AP uses the processed fingerprints to train the local authenticator.

**3.2. Authentication Workflow in Association Phase.** For each data packet from STA, AP extracts the CSI fingerprint and sends it to the local authenticator. Then, AP judges whether the fingerprint is valid according to the output of the authenticator. It can be seen from Figure 1 that if the CSI fingerprint is judged to be valid, the AP accepts this data packet and uses the CSI fingerprint to train the authenticator. If the fingerprint fails the authentication, the data packet is dropped.

AP keeps a counter called F-Counter. It records the number of the packets that continuously fail the authentication. When the authentication fails, the F-Counter is incremented, and AP determines whether the connection should continue according to whether the threshold  $f$  is reached. Otherwise, the F-Counter will be cleared.

## 4. Authentication Leveraging CSI Fingerprints

In this section, we focus on the device authentication based on CSI fingerprints. We first explain the basic idea of how to do authentication using CSI fingerprints, and then we describe the device authentication algorithms used in access phase and association phase.

**4.1. Basic Idea.** CSI is the CFR of a wireless multipath channel, including the amplitude attenuation and phase shift. In a Wi-Fi network, the wireless channel between each STA and AP is unique, which means that the CSI of the channel between each STA and AP is also unique. In the light of this idea, the AP can uniquely mark a STA through the CSI obtained from the packets sent by this device. Therefore, we refer to the CSI collected from the STA as CSI fingerprints. Figure 2 shows the amplitude image of the CSI fingerprints obtained from four STA placed in different positions. We can see that the fingerprints' distributions of the same device are concentrated, while the fingerprints' distributions between different devices are different. Considering that there is often interference in real environment, we collected CSI fingerprints under three interference conditions, and the results are shown in Figure 3. Even if there is interference, the CSI fingerprint is still stable. Therefore, it is reasonable to use CSI fingerprints for Wi-Fi device authentication.

In access phase, AP sends CSI fingerprints collecting requests to the target STA. If there is no attacker or the attacker does not do any response, the fingerprint set will only contain fingerprints with the same distribution, which belong to the target STA. However, if the identity-based attackers make response to AP, the fingerprint set will contain fingerprints of at least two different distributions. The framework takes advantage of this distinction to perform authentication. The framework quantifies the number of distributions by the dispersion degree of the fingerprint set.

In the association phase, the authenticator has been trained with new CSI fingerprints obtained in access phase. Our framework extracts CSI fingerprint from each packet received and perform authentication at the packet level. Autoencoders are used to build the local authenticator. The trained autoencoder is expected to rebuild the fingerprints of the target STA with low error, but it cannot rebuild the

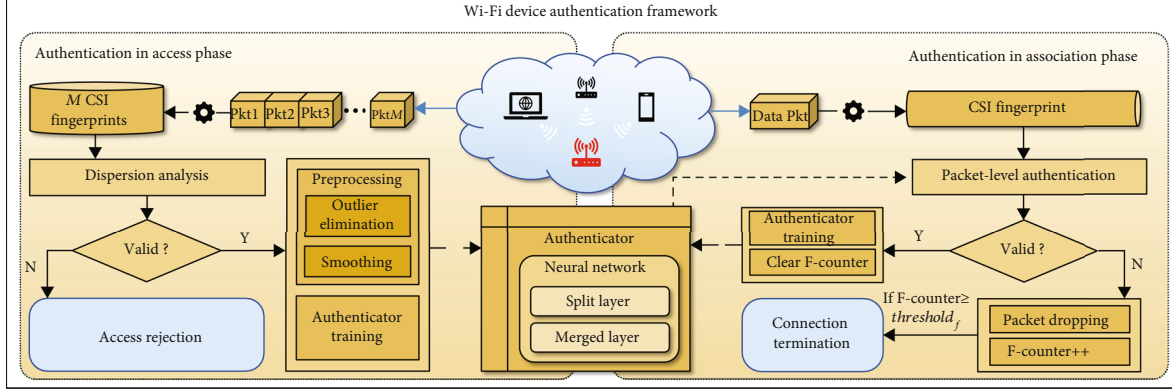


FIGURE 1: The framework of our Wi-Fi device authentication system.

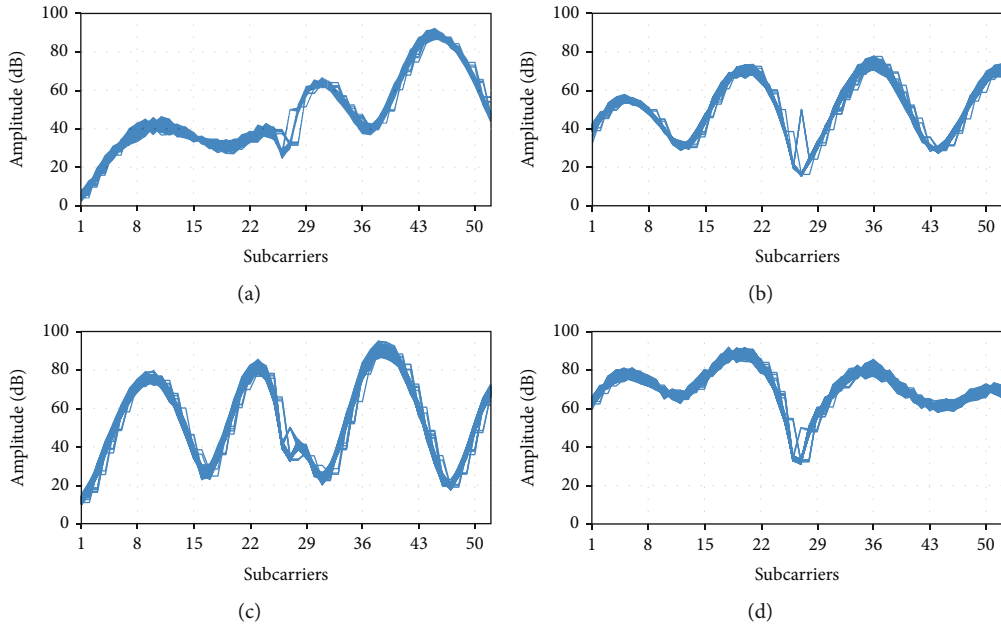


FIGURE 2: The amplitude of CSI collected in different locations. (a) Location 1. (b) Location 2. (c) Location 3. (d) Location 4.

unknown fingerprints collected from the attackers well. The framework makes use of this characteristic for authentication. To keep the authenticator adapt to the channel state, each valid fingerprint is used to train the authenticator. Moreover, we use an ensemble of small-scale autoencoders instead of a large-scale autoencoder, because the time efficiency of training and executing an ensemble of small-scale autoencoders is higher than that of a large-scale autoencoder. It is confirmed in Section 4.4 by theoretical derivation and Section 6.2 by evaluation results.

#### 4.2. Authentication Scheme in Access Phase

**4.2.1. Dispersion Analysis.** Let  $C_D$  stands for the fingerprints set collected from the STA  $D$  and the  $k$ -th CSI fingerprint in  $C_D$  indicated by  $C_D^k = \{C_{D,1}^k, \dots, C_{D,N}^k\}$  ( $k = 1, \dots, K$ ), where  $N$  is the number of subcarrier, and  $K$  is the number of samples obtained from  $D$ .

After collecting CSI fingerprints from STA, the framework uses standard deviation of  $C_D$  to quantify the dispersion degree:

$$\sigma = \left( \frac{1}{K} \sum_{k=1}^K (C_D^k - \bar{C}_D)^2 \right)^{1/2}, \quad (1)$$

where  $\bar{C}_D$  is the mean of the fingerprints in  $C_D$ .

If the standard deviation exceeds the predefined threshold, the STA is judged to be invalid and denied access to the network. Otherwise, the STA successfully access to the network, and the fingerprints are used for training the local authenticator. We designate the threshold as  $\text{threshold}_{ac}$ , which means the threshold used in access phase.

**4.2.2. Fingerprint Preprocessing.** We found that there are often fingerprints that far exceed the expected numerical fluctuation range, as shown in Figure 4(a). They do not

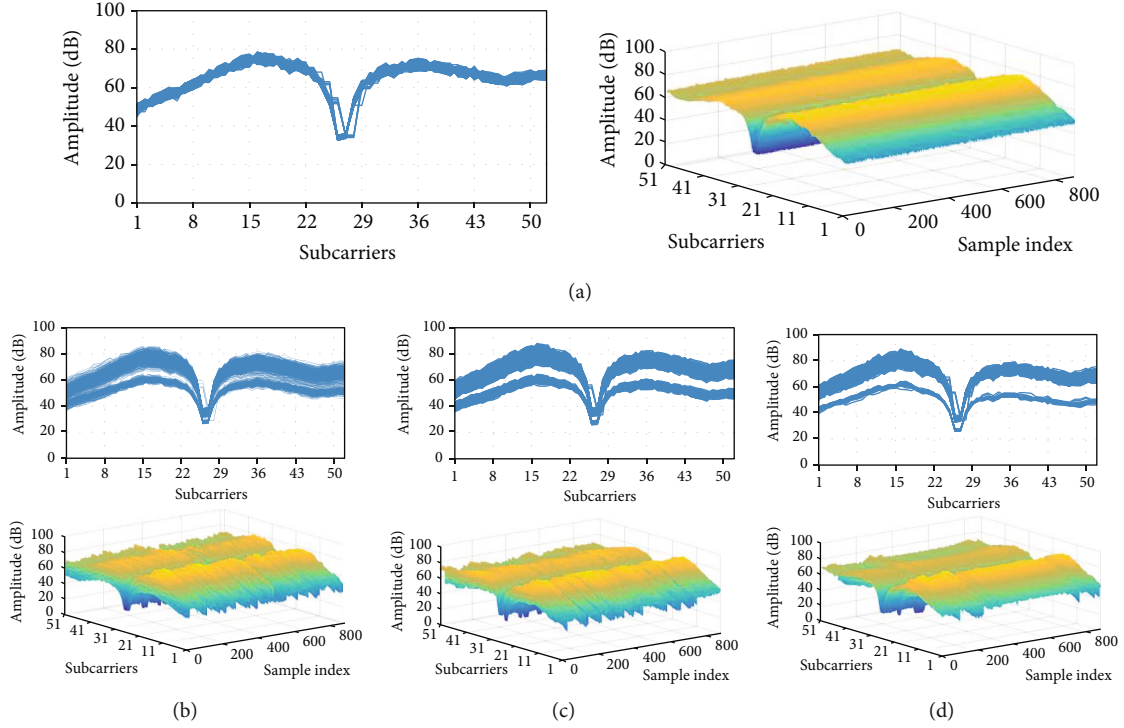


FIGURE 3: The amplitude of CSI collected in different interference environments. (a) Non-interference. (b) Interference -1 m. (c) Interference -2 m. (d) Interference -3 m.

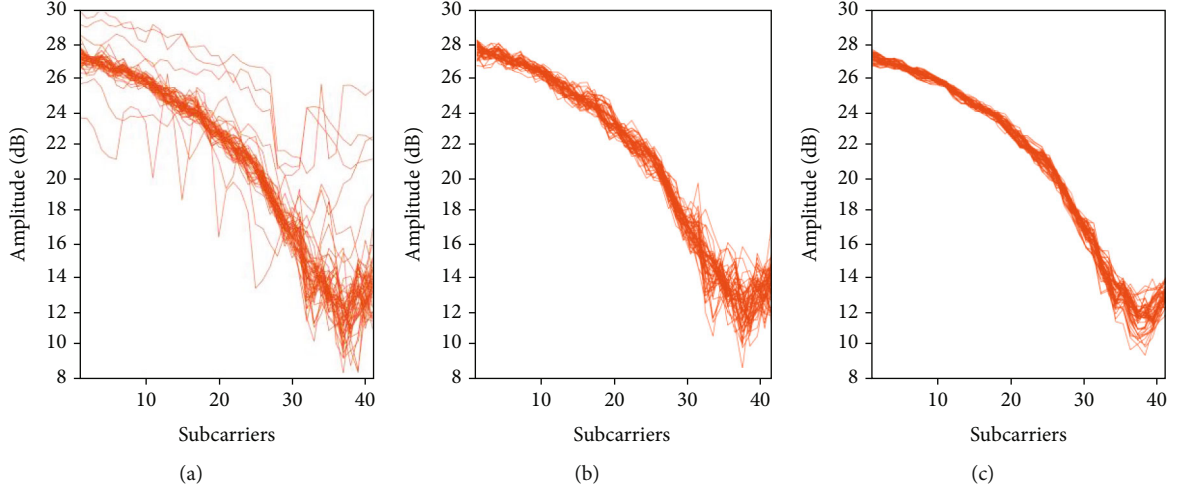


FIGURE 4: The amplitude of CSI in different preprocessing stages. (a) Before preprocessing. (b) After removing outliers. (c) After smoothing.

contain any information of device identity and will reduce the accuracy of system. Therefore, we use the Hampel identifier to exclude these outliers. In addition, the CSI fingerprints of the same device will change under noise interference. Even though the CSI images of the same device still have the same trend, but the dispersion increases. The local outlier factor describes the difference between fingerprints based on the Euclidean distance. It means the noise will cause the reference sample group become discrete and reduce the probability of abnormal CSI being detected.

Therefore, we smooth the CSI fingerprints in the time domain. The fingerprint preprocessing consists of two stages, including the outlier elimination and the smoothing.

(1) *Outlier Elimination.* We perform outlier detection on each subcarrier individually. First, we arrange the amplitude samples of  $n$ -th subcarrier in chronological order, which can be presented as  $L_{D,n} = (C_{D,n}^1, \dots, C_{D,n}^K)$ . Then,  $C_{D,n}^i$  ( $i = l + 1, \dots, K - l$ ) and its  $2l$  surrounding amplitude samples form a window  $W_i$ . In  $W_i$ , the absolute deviation

of each amplitude sample is used to estimate the standard deviation of the amplitude samples. The standard deviation is shown as follows:

$$\sigma_{D,n}^I = \frac{1}{\gamma} \text{median}(|C_{D,n}^i - C_{D,n}^I|), \quad (2)$$

where  $C_{D,n}^I$  indicates the median of  $W_i$  and  $\gamma = \sqrt{2} \text{erfinv}(0.5)$ ,  $\text{erfinv}$  on behalf of the inverse error function.

We then perform the following numerical substitution on each amplitude sample in  $W_i$  to exclude the outlier:

$$C_{D,n}^i = \begin{cases} C_{D,n}^i, & |C_{D,n}^i - C_{D,n}^I| \leq \eta \sigma_{D,n}^I \\ C_{D,n}^I, & |C_{D,n}^i - C_{D,n}^I| > \eta \sigma_{D,n}^I \end{cases} \quad (3)$$

where  $\eta$  is a threshold used to judge whether the sample is an outlier.

We repeat the above work for  $W_i$  ( $i = l + 1, \dots, K - l$ ) on each subcarrier. Figure 4(b) shows the CSI fingerprints without the outliers.

(2) *Smoothing*. We smooth the amplitude of each subcarrier in the time domain to reduce this effect. The smoothing process is described as follows:

$$\tilde{C}_{D,n}^k = \frac{1}{\omega} \sum_{\max(0, k - \lfloor \frac{\omega}{2} \rfloor) \leq i \leq \min(K, k + \lfloor \frac{\omega}{2} \rfloor)} C_{D,n}^i, \quad (4)$$

where  $\omega$  indicates the length of smoothing window. Figure 4(c) shows the CSI fingerprints after smoothing.

**4.3. Packet-Level Authentication Scheme in Association Phase.** We assume that the STA has been successfully connected and starts to transmit data. The successful access of the STA means that the fingerprints collected by the AP is valid, which means that we can successfully authenticate the data sent by the STA in association phase.

Before authenticating the STA in association phase, the authenticator should be trained with the processed fingerprints obtained in access phase.

**4.3.1. Authenticator Training.** Figure 5 shows the organization of the authenticator. The authenticator consists of two layers, including split layer and merged layer. We can see from Figure 5 that the basic functional unit of each layer is autoencoder, which is shown on the right of Figure 5. It consists of three layers of fully connected neurons. The input and output layers have the same number of neurons, and the middle layer, named hidden layer, has a smaller number of neurons. The autoencoder first compresses the input samples and then reconstructs them into the original dimensions. It tries to learn the distribution characteristics of the input sample set for reducing the reconstruction error.

In this paper, the autoencoders in split layer learn the distribution characteristics of the input CSI fingerprints

and reconstruct them. The reconstruction error of each autoencoders is sent to the output autoencoder in merged layer. The root mean squared error (RMSE) between input and output is used for indicating the reconstruction error.

Before entering a new CSI fingerprint into split layer, authenticator first normalizes the fingerprint. The normalization algorithm is shown as follows:

$$S_{D,i}^k = \frac{C_{D,i}^k - C_{D,\min}^k}{C_{D,\max}^k - C_{D,\min}^k}, \quad (5)$$

where  $C_{D,\min}^k$  and  $C_{D,\max}^k$  represent the minimum and maximum values of  $C_{D,i}^k$  ( $i = 1, \dots, N$ ).

The normalized fingerprints  $S_D^k$  are equally divided into  $I$  subfingerprints ( $S_D^k = \{S_D^{k(1)}, \dots, S_D^{k(I)}\}$ ), and they are sent to the autoencoders in split layer in order (pictured in Figure 5). Next, we focus on the training process of a single autoencoder.

Next, we focus on the training process of the authenticator. We initialize each autoencoder in two layers with random numbers with the distribution  $\mathcal{U}(-1/\dim(x), 1/\dim(x))$  ( $x$  is the input of the autoencoder) before training. For clearly presenting the training process, we use  $L1$  and  $L2$  to represent split layer and merged layer. We use  $\theta_i$  for the  $i$ -th autoencoder  $L1$  and  $\theta_0$  for the autoencoder in  $L2$ . The training algorithm is shown in Algorithm 1, where  $v$  is the input of  $\theta_0$ ,  $S_D^{k(i)'}$  is the output of  $\theta_i$ , and  $w'$  is the output of  $\theta_0$ .

Authenticator's update is important in association phase, because if the authenticator become outmoded, the data packets sent from STA will be refused. The updating method of the authenticator presented in this paper can improve this problem without compromising the safety of the system.

**4.3.2. Packet-Level Authentication.** For a new data packet from the STA, the framework first extracts the CSI fingerprint  $C_D$  from the packet and then normalizes it using equation (5). The normalized CSI fingerprint  $S_D$  is sent to the local authenticator, and the authentication algorithm is shown in Algorithm 2, where  $v$  is the input of  $\theta_0$ ,  $S_D^{(i)'}$  is the output of  $\theta_i$  and  $w'$  is the output of  $\theta_0$ . If the CSI fingerprint is judged to be valid, it will be used to train the local authenticator.

**4.4. Packet-Level Authentication Complexity.** This subsection shows the time complexity of the packet-level authentication in association phase. To show the improvement of ensemble learning on time efficiency, we make a comparison between the time complexity of our authenticator (an ensemble of small-scale autoencoders) and a single large-scale autoencoder by mathematical calculations.

In the authentication framework,  $N$  is the number of subcarriers, and the number of subfingerprints (also the number of autoencoders in split layer) is  $I$ . Thus, the input dimension of the autoencoder in  $L1$  and  $L2$  is, respectively, equal to  $N/I$  and  $I$ . We use  $\alpha$  as the dimensionality reduction

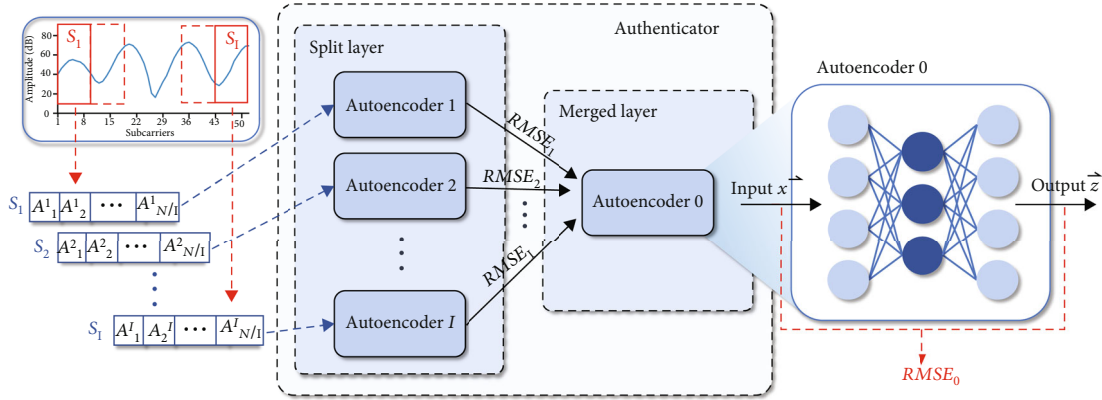


FIGURE 5: The local authenticator for authentication in association phase.

```

Input:  $S_D^k$ 
1:  $\mathbf{v} \leftarrow \text{zeros}(I)$ 
   //train Split layer
2: For  $\theta_i$  in  $L1$  do
3:  $\theta_i, S_D^{k(i)'} \leftarrow \text{SGD}(S_D^{k(i)})$ 
4:  $\mathbf{v}[i] \leftarrow \text{RMSE}(S_D^{k(i)}, S_D^{k(i)'})$ 
5: End for
   //train merged layer
6:  $\mathbf{w} \leftarrow \text{norm}_{0-1}(\mathbf{v})$ 
7:  $\theta_0, \mathbf{w}' \leftarrow \text{SGD}(\mathbf{w})$ 
8: Return  $\text{RMSE}(\mathbf{w}, \mathbf{w}')$ 

```

ALGORITHM 1: The authenticator training algorithm.

```

Input:  $S_D$ 
1:  $\mathbf{v} \leftarrow \text{zeros}(I)$ 
2: For  $\theta_i$  in  $L1$  do
3:  $S_D^{(i)'} \leftarrow h_{\theta_i}(S_D^{(i)})$ 
4:  $\mathbf{v}[i] \leftarrow \text{RMSE}(S_D^{(i)}, S_D^{(i)'})$ 
5: End for
6:  $\mathbf{w} \leftarrow \text{norm}_{0-1}(\mathbf{v})$ 
7:  $\mathbf{w}' \leftarrow h_{\theta_i}(\mathbf{w})$ 
8: If  $\text{RMSE}(\mathbf{w}, \mathbf{w}') < \text{threshold}_{as}$  then
   //train Split layer and merged layer
9: For  $\theta_i$  in  $L1$  do
10:  $\theta_i \leftarrow \text{SGD}(S_D^{(i)})$ 
11: End for
12:  $\theta_0 \leftarrow \text{SGD}(\mathbf{w})$ 
13: End if
14: Return  $\text{RMSE}(\mathbf{w}, \mathbf{w}')$ 

```

ALGORITHM 2: The authenticator execution algorithm.

ratio of the autoencoder's middle layer, thus, the neurons' number in the middle layer in  $L1$  and  $L2$  are  $\alpha N/I$  and  $\alpha I$ .

If the input of an autoencoder is  $u$  and the dimensionality reduction ratio is  $\alpha$ , the activation of the middle layer and the output layer both requires  $u \cdot \alpha u = \alpha u^2$  calculations.

Therefore, the complexity of the activation is  $O(\alpha u^2) = O(u^2)$ . The backward propagation has the same complexity. Before calculating the complexity of the authenticator, we assume that the autoencoders in split layer operate serially in the framework. Therefore, the complexity of the authenticator's training is  $O(N^2/I + I^2)$ . Let  $N = \beta I$ , where  $\beta$  is the length of the subfingerprint. If we limit the size of the autoencoder in split layer to 6 and below, then,  $\beta$  can be regarded as a constant. Thus, the complexity is  $O(\beta^2 I + I^2) = O(I^2)$ . In the association phase, we assume that all packets are authenticated successfully and the authenticator needs to be updated every time. Then, the complexity of the authenticator is  $O(I^2)$ .

If we use a single autoencoder instead, the complexity becomes  $O(N^2)$ . Our authenticator based on ensemble learning reduces the time complexity of the framework from  $O(N^2)$  to  $O(I^2)$ .

## 5. The CSI-Based Authentication Protocol

It can be seen from the previous section that the association and cooperation between the AP and the STA are the key to do device authentication and fingerprint database update in both the access and the association phase. Therefore, we present an enhancing Wi-Fi device authentication protocol in this section, which is designed based on the existing Wi-Fi association process.

**5.1. Authentication Procedure in Access Phase.** This subsection describes the authentication procedure in access phase, which is shown in Figure 6(a). The steps are shown as follows:

- (i) STA sends probe request to request access to the network. AP responds with probe response after listening to the request, indicating that it can provide access service
- (ii) STA then sends authentication request for the authentication. AP returns collection request to inform STA that it is ready for CSI collection

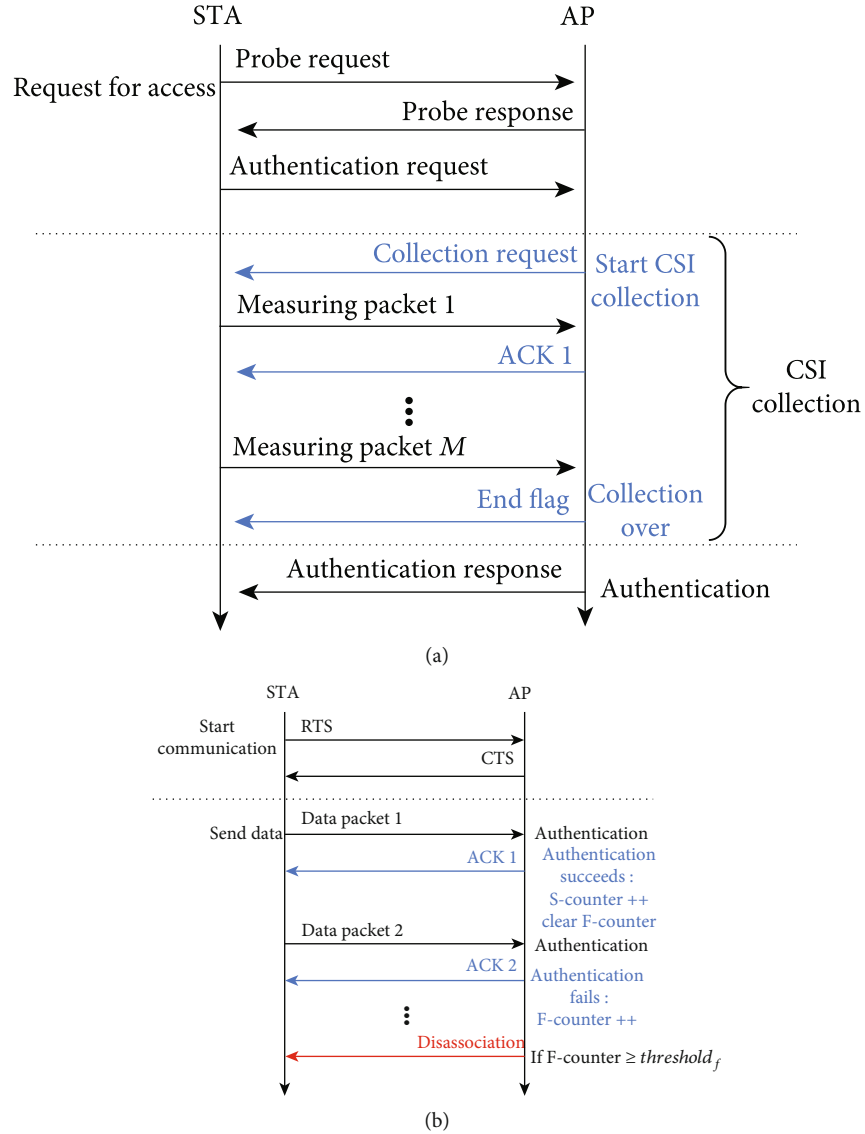


FIGURE 6: The procedure of CSI-based authentication protocol. (a) Access phase. (b) Association phase.

- (iii) After receiving collection request, STA starts to send measuring packets with a fixed data length of 24 bytes. After receiving the data packets, AP responds with ACK to tell STA to continue or stop
- (iv) After collecting enough CSI fingerprints, AP authenticates the target STA and judges whether the STA is valid. The authenticating result is sent to the target STA through authentication response

**5.2. Packet-Level Authentication Procedure in Association Phase.** Figure 6(b) shows the packet-level authentication in association phase. In Wi-Fi association, CSMA/CA is used to avoid the collisions among packets. The STA exchanges RTS/CTS with the AP before starting to transmit data to inform other devices to remain silent. After exchanging RTS/CTS, the AP performs CSI matching on each received data frame, judges whether the current connection is valid

according to the matching situation, and decides whether to continue association or not (as shown in section 2).

The real-time update of the authenticator is based on the active communication between the AP and the STA. When the STA is in sleeping mode, keeping the CSI collection will cause greater power consumption and increase the network burden, so this is not recommended. After a device returns to be active, the local authenticator may become outmoded. To retrain the authenticator, we disconnect after threshold<sub>f</sub> consecutive data packets fail the authentication (see section 2); then, the authentication process will return to access phase.

In addition, we must also consider the existence of the attacker. When an attacker uses a forged identity to send a data packet, the AP will get the failed authentication result and discard the packet. If the AP continuously receives threshold<sub>f</sub> packets from the attacker, the connection is broken. We can find that the attacker cannot attack the AP effectively during this whole process.

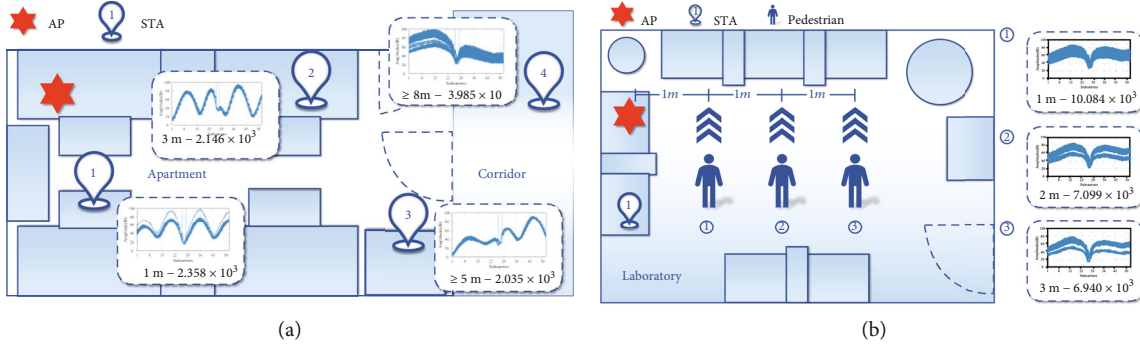


FIGURE 7: The test scenarios used in the evaluation. (a) Apartment. (b) Laboratory.

- (i) STA sends RTS to tell AP that it is going to send data. AP responds with CTS after receiving the request, indicating that it is ready receiving and clearing the channel
- (ii) STA starts to send data packets. When AP receives a data packet, it responds with ACK. AP extracts the CSI of the packet and performs authentication. If the authentication is successful, the S-Counter is increased by 1, and the F-Counter is cleared. Otherwise, the F-Counter is increased by 1
- (iii) If F-Counter reaches threshold $_f$ , AP sends disassociation to STA for disconnection

## 6. Performance Evaluation

In this section, we make an evaluation on the performance of our authentication framework.

**6.1. Experiment Setup and Metrics.** To simulate real application scenarios, we conducted experiments in both apartment and laboratory, which are complex multipath environments and in public networks with high traffic.

A commercial Wi-Fi device, HUAWEI TAS-AN00, works as STA transmits data packets at a rate of 100 pkt/sec in 20 MHz Wi-Fi channel on 2.4 GHz, which runs EMUI 11.0.0. We use the Raspberry Pi 3b+ as AP, which runs Raspbian Buster Lite 4.19.97 and is equipped with BCM43455c0 Wi-Fi card. By modifying the Wi-Fi card driver, it can pack the CSI fingerprints extracted from the specified devices' data packets in UDP datagrams and send it to the application layer. We then use TCPDump to grab the UDP datagrams and read the CSI fingerprints. The plug-in used to extract CSI from the Wi-Fi card of Raspberry Pi 3b+ is available in [25].

In the experiments performed in apartment, we place the STA in four locations fixedly, which is shown in Figure 7(a). The first three locations are distributed in the apartment at different distances, and the fourth location is in the corridor outside the apartment. We use these experiments to analyze the impact of distance on the performance of the authentication framework. In the experiments performed in laboratory, we put the AP and STA in fixed locations and walk back and forth in the positions shown in Figure 7(b). These three

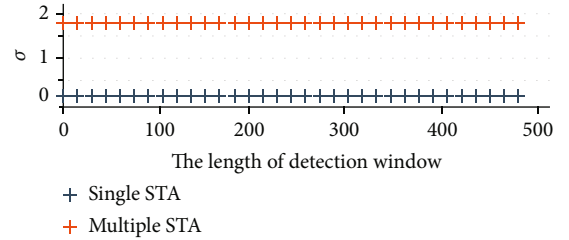


FIGURE 8: The standard deviation of CSI fingerprints with and without attacker.

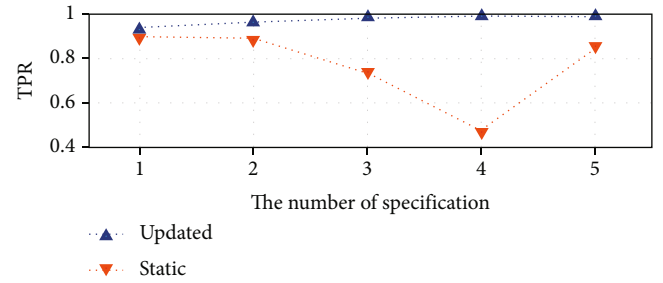


FIGURE 9: The authentication framework's accuracy using different specifications of authenticator.

modes of movement interfere with the collection of CSI to different degrees. These experiments help us observe the effects of different levels' interference on the authentication framework's performance.

In the apartment experiments, we collect 10,000 CSI fingerprints at each location, and each CSI fingerprint contains 52 subcarrier amplitudes. These datasets serve to evaluate the performance of our authentication framework in access phase and association phase. In the laboratory experiments, the same number of fingerprints is obtained in three modes of movement, which are used for evaluation in association phase. Since the AP will receive irrelevant frames from other unknown Wi-Fi devices, we record the MAC of each STA and filter the frames by MAC checking. For each valid frame received, the AP extracts CSI fingerprint and saves it locally.

To evaluate the performance of the authentication framework, we use the true positive rate (TPR) as the specific criteria for judging. TPR is calculated as  $TP/(TP + FN)$ , where TP is the number of true positive samples, and FN is the number of false negative samples.

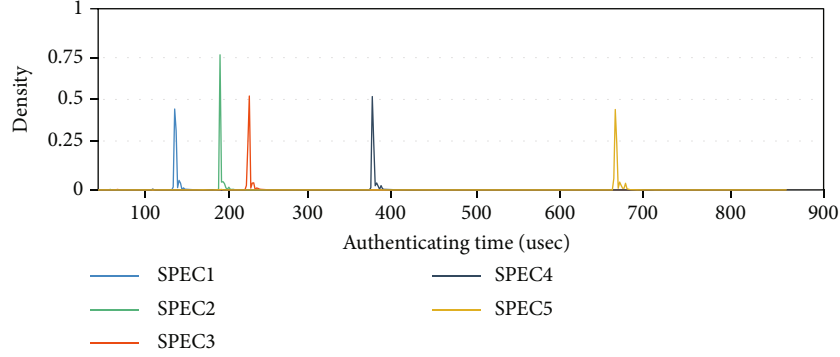


FIGURE 10: Per-packet authenticating time using different specifications of authenticator.

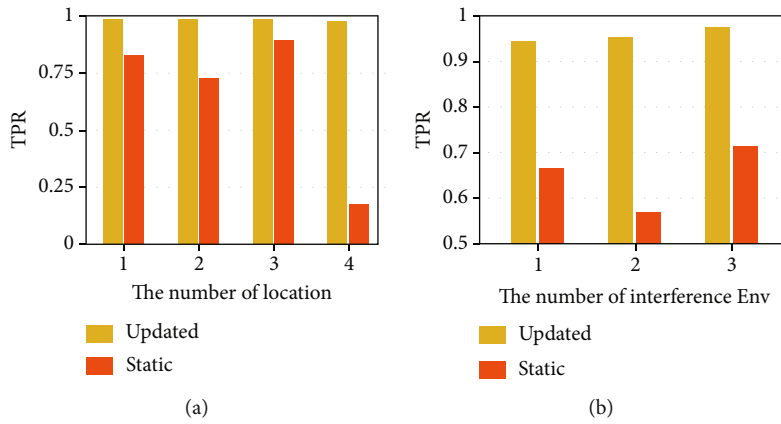


FIGURE 11: The authentication framework's accuracy in two test scenarios. (a) Apartment. (b) Laboratory.

**6.2. Authentication Performance.** In access phase, we need to determine the decision threshold for dispersion analysis. To this end, we use the CSI fingerprints obtained in the apartment experiments to test the degree of dispersion with and without the attacker. The test result is shown in Figure 8. Among them, the blue plus sign marks the standard deviation of the CSI fingerprints when only STA1 requests access, and the red plus sign marks the standard deviation when four STA request access at the same time.  $X$  represents the length of the detection window. Different detection window lengths have a stable standard deviation, and the standard deviation of a single STA is much lower than that of multiple STAs. Therefore, we set the decision threshold to 0.5, which can be used for access phase's authentication in the apartment experiments with an accuracy of 1. We set the detection window length to 100. Because on the one hand it can shorten the CSI collecting time compared with a longer window, on the other hand, it can reduce the probability of misjudgment caused by channel noise compared with a shorter window.

The number of autoencoders used in the authenticator's split layer and merged layer is an important factor affecting the overall performance of the authentication framework. We choose 5 different specifications of authenticators for performance evaluation, their  $I$  are 13, 6, 4, 2, and 1. Figure 9 shows the TPR of each specification. We can see

that the TPR increases from 93.278% to 99.278% as the number of autoencoders in split layer increases. However, the increase in  $I$  means that the number of neurons in autoencoder increases, and the time efficiency of the authenticator will decrease, as analyzed in Section 4.4. To prove our theoretical analysis, we recorded the packet-level authentication time of 10,000 data packets under 5 different specifications and plotted the density map, which is shown in Figure 10. As we can see in the figure, the packet-level authenticating time is around 150 usec when  $I = 13$ . However, the authenticating time up to 660 usec when  $I = 1$ , which is 4.4 times as many as the former. Based on the above factors, we choose to use the third specification authenticator ( $I = 4$ ) for the following evaluation.

In the apartment experiments, the authentication results of the first three locations are similar, and their TPR is higher than 98.6% (depicted in Figure 11(a)). However, the TPR of the position 4 drops to 97.87%. The CSI fingerprints' variance of location is up to  $3.985 \times 10^3$  (as shown in Figure 7(a)), which is much higher than the variance of the first three locations. This shows that the distance between AP and STA is not an essential factor affecting the stability of CSI fingerprints. Although the increase of the distance will increase the possibility of interference, the distance has little effect on the stability of the CSI fingerprint in a stable indoor environment and will not affect the authentication



framework's performance. In the corridor, the CSI fingerprints show relatively poor stability, because in the absence of line of sight (LOS), any small disturbance will have a greater impact on the multipath channel state.

In the laboratory experiments, the third device with the least disturbance has the largest TPR, which is 97.53%. Meanwhile, the first device with the most interference had a TPR of 94.36% (as shown in Figure 11(b)). This is in line with our expectations. The CSI sample variances of the three interference environments are  $10.084 \times 10^3$ ,  $7.099 \times 10^3$ , and  $6.941 \times 10^3$  (Figure 7(b)). We can see that noise will reduce the stability of CSI fingerprints collected from Wi-Fi devices and then decrease the accuracy of the authentication framework. However, the results show that the accuracy of the authentication framework can still reach more than 94% even when there are disturbances such as people walking and objects moving at 1 m, which means that our authentication framework has strong robustness.

In each performance evaluation, we used two different methods for authentication. The first method is updating the authenticator with the update scheme shown in Section 4.3. The other one is keeping a static authenticator in the AP. In the environments with less interference such as locations 1, 2, and 3 in the apartment, the minimum TPR when using the static authenticator is 72.91%, which is 25.77% lower than using the updated authenticator; the maximum is 89.69% and is 9% lower than the second method (Figure 11(a)). In the environments with interference (location 4 in the apartment and laboratory), the minimum TPR when using the first authentication method is 17.37%, which is 80.5% lower than using the updated authenticator; the maximum is 71.49% and is 26.04% lower than the second method (depicted in Figures 11(a) and 11(b)). Therefore, the authenticator update scheme presented in this paper can improve the performance of the authentication framework, and the effect is more significant under interference environment.

However, there are some shortcomings in this scheme. First, because CSI is highly dependent on the environment in which the device is located, it faces frequent disconnections if the device moves quickly. Second, the scheme requires that the legitimate device is always within the range of the wireless network; otherwise, it will not be able to prevent effectively when the attacker appears. We will continue to study and overcome these possible problems in the future work.

## 7. Conclusion

In this paper, we propose an enhancing Wi-Fi device authentication protocol. We also give a complete Wi-Fi device authentication framework. The framework mainly contains two parts, including the authentication in access phase and the authentication in association phase. We provide different authenticating algorithms for both authentication phases. What is more, we present an authenticator composed of small-scale autoencoders and the authenticator update method. The evaluation shows that our authentication methods in both authentication phases have good performance. Also, our authenticator has a higher time effi-

ciency than a single neural network, which can help our framework to do packet-level authentication.

## Data Availability

The data set in this paper includes CSI samples collected in two experiment scenarios, which is available from the corresponding author upon request. The code of our authentication framework is available in <https://github.com/Chinmize/CSIAuthenticator>.

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work is supported by Frontiers Science Center for Mobile Information Communication and Security, Southeast University, Nanjing, China. This work is also supported by the Zhishan Youth Scholar Program of SEU, Nanjing, China. Yubo Song is the corresponding author.

## References

- [1] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
- [2] S. Liu, "Mac spoofing attack detection based on physical layer characteristics in wireless networks," in *2019 IEEE International Conference on Computational Electromagnetics (ICCEM)*, pp. 1–3, Shanghai, China, 2019.
- [3] P. Madani, N. Vljajic, and S. Sadeghpour, "Mac-layer spoofing detection and prevention in IoT systems: randomized moving target approach," in *Proceedings of the 2020 Joint Workshop on CPS & IoT Security and Privacy (CPSIoTSEC'20)*, pp. 71–80, New York, NY, USA, 2020.
- [4] J. S. Alshudukhi, B. A. Mohammed, and Z. G. Al-Mekhlafi, "An efficient conditional privacy-preserving authentication scheme for the prevention of side-channel attacks in vehicular ad hoc networks," *IEEE Access*, vol. 8, pp. 226624–226636, 2020.
- [5] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang, "IoT device fingerprinting for relieving pressure in the access control," in *Proceedings of the ACM Turing Celebration Conference (ACM TURC'19)*, New York, NY, USA, 2019.
- [6] C. Yan and J. Ge, "Synchronous control of master-slave manipulator system under deception attacks," in *2020 Chinese Control and Decision Conference (CCDC)*, pp. 1778–1782, Hefei, China, 2020.
- [7] Y. Tian, Z. Wang, J. Xiong, and J. Ma, "A blockchain-based secure key management scheme with trustworthiness in dwsns," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6193–6202, 2020.
- [8] L. Yang, Y. Song, S. Gao, B. Xiao, and A. Hu, "Griffin: an ensemble of autoencoders for anomaly traffic detection in SDN," in *2020 IEEE Global Communications Conference (GLOBECOM 2020)*, pp. 1–6, Taipei, Taiwan, 2020.

- [9] U. Chatterjee, R. Sadhukhan, D. Mukhopadhyay, R. Subhra Chakraborty, D. M. Mahata, and M. Prabhu, "Stupify: a hardware countermeasure of kracks in wpa2 using physically unclonable functions," in *Proceedings of the 2020 Companion Proceedings of the Web Conference*, pp. 217–221, Taipei Taiwan, 2020.
- [10] A. Elhigazi, S. Abd Razak, M. Hamdan, B. Mohammed, I. Abaker, and A. Elsafi, "Authentication flooding dos attack detection and prevention in 802.11," in *2020 IEEE Student Conference on Research and Development (SCORED)*, pp. 325–329, Batu Pahat, Malaysia, 2020.
- [11] S. Gao, Z. Peng, B. Xiao, A. Hu, Y. Song, and K. Ren, "Detection and mitigation of dos attacks in software defined networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1419–1433, 2020.
- [12] W. Kim, S. Kim, and H. Lim, "Malicious data frame injection attack without seizing association in IEEE 802.11 wireless LANs," *IEEE Access*, vol. 9, pp. 16649–16660, 2021.
- [13] R. Liao, H. Wen, F. Pan, H. Song, A. Xu, and Y. Jiang, "A novel physical layer authentication method with convolutional neural network," in *2019 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pp. 231–235, Dalian, China, 2019.
- [14] Q. Wang, H. Li, D. Zhao, Z. Chen, S. Ye, and J. Cai, "Deep neural networks for CSI-based authentication," *IEEE Access*, vol. 7, pp. 123026–123034, 2019.
- [15] J. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in iiot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.
- [16] J. Xiong, M. Zhao, M. Z. A. Bhuiyan, L. Chen, and Y. Tian, "An ai-enabled threeparty game framework for guaranteed data privacy in mobile edge crowdsensing of iot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2021.
- [17] H. Liu, Y. Wang, J. Liu, J. Yang, and Y. Chen, "Practical user authentication leveraging channel state information (CSI)," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS'14)*, pp. 389–400, New York, NY, USA, 2014.
- [18] M. Liu, A. Mukherjee, Z. Zhang, and X. Liu, "TBAS: enhancing Wi-Fi authentication by actively eliciting channel state information," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1–9, London, UK, 2016.
- [19] N. Xie, J. Chen, and L. Huang, "Physical-layer authentication using multiple channel-based features," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2356–2366, 2021.
- [20] K. S. Germain and F. Kragh, "Multi-transmitter physical layer authentication using channel state information and deep learning," in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–8, Adelaide, SA, Australia, 2020.
- [21] K. S. Germain and F. Kragh, "Physical-layer authentication using channel state information and machine learning," in *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–8, Adelaide, SA, Australia, 2020.
- [22] J. Xiong, J. Ren, L. Chen et al., "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2019.
- [23] J. Zhang, Z. Wang, K. Wang, S. Guo, B. Wang, and M. Guo, "Improving power efficiency for online video streaming service: a self-adaptive approach," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 3, pp. 308–313, 2019.
- [24] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging WiFi-enabled IoT," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '17)*, pp. 1–10, New York, NY, USA, 2017.
- [25] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *Network and Distributed System Security Symposium*, 2018, <http://arxiv.org/abs/1802.09089>.

## Research Article

# Signal Modulation Recognition Method Based on Differential Privacy Federated Learning

Jibo Shi , Lin Qi, Kuixian Li , and Yun Lin 

College of Information and Communication Engineering, Harbin Engineering University, Harbin, China

Correspondence should be addressed to Yun Lin; linyun@hrbeu.edu.cn

Received 19 July 2021; Revised 5 September 2021; Accepted 11 September 2021; Published 4 October 2021

Academic Editor: Jinbo Xiong

Copyright © 2021 Jibo Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Signal modulation recognition is widely utilized in the field of spectrum detection, channel estimation, and interference recognition. With the development of artificial intelligence, substantial advances in signal recognition utilizing deep learning approaches have been achieved. However, a huge amount of data is required for deep learning. With increasing focus on privacy and security, barriers between data sources are sometimes difficult to break. This limits the data and renders them weak, so that deep learning is not sufficient. Federated learning can be a viable way of solving this challenge. In this article, we will examine the recognition of signal modulation based on federated learning with differential privacy, and the results show that the recognition rate is acceptable while data protection and security are being met.

## 1. Introduction

At a time when the volume of information is rapidly increasing, various modulation methods are commonly employed to fully utilize the channel's ability to carry data swiftly and effectively. The modulation method has therefore become one of the essential features to differentiate different sorts of communications. In the military sector, the identification of signal modulation offers an essential basis for information interception and for selecting the best possible interference in electronic warfare systems. It is mostly apparent in the detection of hostile radar types, in the interception of the intelligence of the adversary, and in the recognition of enemy radio sources. In civil matters, modulation recognition is mostly utilized for radio station monitoring and radio platform usage monitoring, and also information provided by nonpartner signals is utilized for monitoring communication spectrum effectively. Wireless communication settings are diverse nowadays, as are modulation schemes. An essential research subject for military and civic usage is how to accurately detect the modular type of a signal in diverse wireless communications settings. Investigations into effective modulation-type recognition algorithms are very important in both the civil and military domains. The modulation rec-

ognition is considered to be an issue with the test of modulation, as long as the density of probability function of the signal is known and classified by comparing the probability function with a set threshold. In recent decades, similar approaches, such as mixed probability ratio tests and mean probability ratio tests, have emerged. Although they are straightforward to implement and function well for clear and well-known signals, the practice of this approach is continually constrained by updated communications technologies and complicated communication settings. Data-driven Chinese learning in the field of signal modulation has recently been actively utilized, and considerable progress has been made. For example, Reference [1] uses GNU radio to create a dataset containing 11 modulation radio signals with  $[-20, +18]$  dB SNR (each dataset sample has two raw data channels  $(I/Q)$  in size  $2 \times 128$ ), and different profounder neural networks including CNN, SVM, and deep neural networks (DNN) are being tested in this set. Reference [2] combines the generation countermeasure network (GAN) with the semisupervised learning network in order to get a more efficient modulation recognition classification. [3] applies robust RPCA to a random forest and obtains the maximum recognition rate of 90%. RPCA is applied to random forests. A deep learning approach which trains and merges two CNN

on distinct training sets has been suggested in [4]. Reference [5] is classified by CV-SVM and achieved by a rate of recognition of 90%. [6] used the RadioML2016.10a dataset to compare and study three different neural network models and their complex-valued counterparts. Their results verified the excellent performance of complex-valued networks in AMC. Reference [7] has developed a framework to convert complex-valued signal waveforms into statistically significant images, called Contour Star Images (CSI), which can convey deep statistical information from the original wireless signal waveforms and represent them in an image data format. Reference [8] proposes a new filter-level pruning technique based on activation maximization (AM) that omits the less important convolutional filter. Semisupervised AMC (TL-AMC) based on transfer learning (TL) is proposed in zero forcing-assisted multiple input multiple output (zf-mimo) systems in [9]. Compared with CNN-based AMC trained on a large number of labeled samples, tl-amc also achieves similar classification accuracy under relatively high signal-to-noise ratio. Reference [10] presents a signal classification method of industrial Internet of things based on feature fusion.

Wireless communication has permeated every part of the work and life of people, and its safety problems cannot be disregarded. Reference [11] and Reference [12] discuss the opportunities and challenges of wireless communication in the 6G era. Because the physiothermal channel of wireless communications is open, the modulation signal containing important information is fully exposed and an attacker can retrieve important signal information by utilizing a blind signal processing technology, which poses a serious threat to legal communication that makes signal data available. For this issue, Reference [13] discusses the performance of a modulation recognition attack method, measures the effectiveness of adversarial attack on signal, and empirically evaluates the reliability of CNN. In particular, privacy and safety are crucial. In parallel, the world is starting to pay more and more attention to data privacy and security through the continuous development of machine learning. In many nations, the security of the data protection has been unparalleled, making it harder to collect data and presenting machine learning with unprecedented problems. No good answers to these issues are available now. Google suggested a distributed learning-federated learning in order to tackle these challenges.

A central server stores and starts sharing global data in a federated learning architecture. The local information is secured, and the local study model is trained based on local data by each client (participant, edge device). Clicking on a specific communication mechanism, the client transfers data like model settings to the central server (the original data of the entire client are not sent). In order to create a global model, the central server collects the data each client uploads. In the whole federated learning process, each client has the same status.

Federated learning has many advantages. First, model training is spread among customers inside the federated learning framework, and each client group updates the gradient autonomously according to their local training data

to reflect the learning model. Based on the fact that the original data is not sent but only the model parameters are changed throughout the training process, federated learning provides data privacy and security, which is also an excellent factor. Secondly, federated learning with the aid of edge computing devices may be deployed with the constant growth of big data and edge computing, which allows the full use of numerous data at the edge without the need for a centralized and efficient data center. And because the model is trained on the user terminal, local data does not leave the "house," reducing communication delay and communication costs due to original data transfer.

These reasons are hot subject federated learning, and several studies about federated learning have been written by various researchers. Reference [14] outlined the notion and application in many sectors of federated learning, defined the forms of federated learning, and forecasted federated learning prospects. Federated learning is influenced by wireless channel uncertainty, and an optimisation approach is provided in [15]. Reference [16] explores the way to improve the effectiveness of federated learning communication and to reduce communications costs. In [17], a new model aggregation approach is presented based on the superpositional features of wireless multiple access channels. This demonstrates the enormous potential and development area for federated learning. However, current privacy protection technologies can provide the privacy protection of federated learning. Common technologies such as anonymity, anonymity, 1-diversity, and  $t$ -closeness cannot withstand background know-how assaults and offer security. Differential privacy is a common and efficient data security technology, which can quantify the degree of data privacy protection. The establishment of an adequate budget for privacy helps create a fair balance between data access and privacy protection. In the privacy protection of training AI model data that can offer significant protection of privacy for federal education, differential privacy is commonly employed.

In 2006, Dwork et al. first suggested differential privacy [18]. This approach of privacy security can prevent leaking of private. This technique is used primarily to tackle two privacy protection concerns. The first difficulty is how data sharing should be carefully configured for confidentiality, and the second is how to guarantee the availability of data protection. A mathematical model for the preservation of privacy is constructed on the basis of the two concerns. Benefiting from the prior knowledge of attackers, the concept of differential privacy is regarded as an efficient privacy security method and is widely utilized in data mining, machine learning, and other disciplines. Many literatures employ differing privacy techniques to secure and enhance the security of model training data.

Differential network privacy (DP-GAN) was proposed by Xie et al. [19] and others in 2018, which might safeguard GAN's privacy by introducing a gradient in the training process and could produce high-quality "fake" ages. The authors offer strict mathematical evidence that DP-GAN meets the confidential privacy criteria. During the same year, Lee and Kifer [20] presented a method for adaptive downward

gradient descent that could change the noise in line with the gradient. Yan et al. [21] suggested a multiposition data publication adaptive sampling mechanism and privacy protection technique and constructed a proportional integral differential (PID) controller-based adaptive sample mechanism. They also developed a quadtree distribution method and the corresponding approach for the allocation of the privacy budget to secure the privacy of the released data. In 2020, they also provided a forecast for centralized posting of large-scale location data on the basis of the potential profound learning paradigm [22].

The RML2016.10a dataset is used in our study to recognize signal modulation based on federal learning and examines training performance. This article is based on our previous work [23], and the contribution is summarized as follows:

- (1) In order to solve the problem of data privacy security in modulation recognition, we apply federated learning to modulation recognition. Different situations are considered to verify the performance of signal modulation recognition based on federated learning
- (2) In view of the lack of defense methods against data attacks on clients in a federated learning framework, this paper introduces differential privacy technology, proposes a differential privacy-driven federated learning method, which is applied to the field of signal modulation recognition, and verifies the performance of signal modulation recognition under different privacy budgets

The rest of this paper is organized as follows. In Section 2, we introduced in detail the principles of federated learning, differential privacy, and convolutional neural networks. After we describe the proposed system model and its implementation process in Section 3, the simulation results are presented in Section 4. This paper is summarized in Section 5, and finally we pointed out the shortcomings of this article and future work.

## 2. Preliminary

We will explain the federated learning framework, differential privacy, and locally trained models in this part.

**2.1. Federated Learning Framework.** As shown in Figure 1, the federated learning system consists of a central server and a large number of remote devices. The central server picks first of all a set of devices  $S_t \subseteq \{1, 2, \dots, T\}$  that fulfills the demands of all devices as training devices. We suppose that there are  $T$  devices in the federated learning system. For example, these criteria include connectivity capabilities of the device, computational power, and whether or not federated learning may take advantage of local data acquired by the device. A global model is then sent to each training device from the central server. Every trainers train a local network using the raw data gathered. After the local model has been trained by each device, updated model parameters are delivered encrypted to the central server and raw data

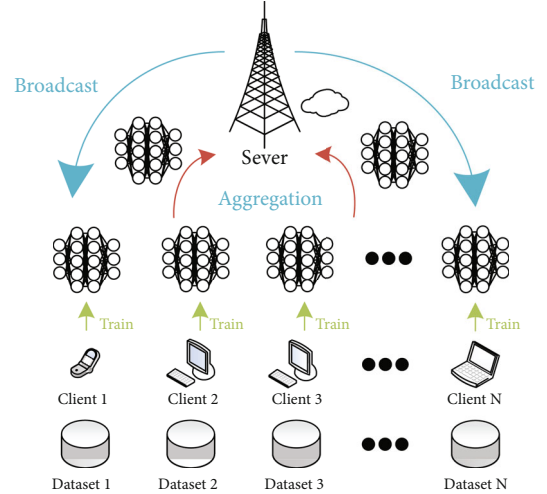


FIGURE 1: Federated learning system.

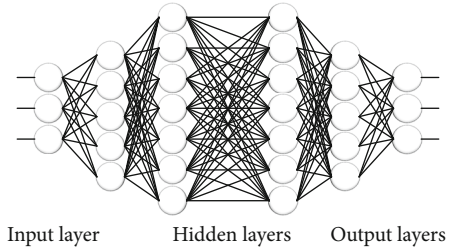


FIGURE 2: Typical convolutional neural network.

from the trainer does not fluctuate. We take the loss function of the local model, and the updated parameters sent by device  $i$  are

$$f_i = l_i(m; a_i, b_i), \quad (1)$$

where  $l_i$  is the input and output data loss function which refers to  $(a_i, b_i)$  the local models and  $m$  is the global model.

Once all the devices are updated on the central server, the aggregation operation is performed. In [24], which is termed secure aggregation, an aggregation approach is proposed. This aggregation procedure ensures the privacy and security of original data, and the attacker is not going to reverse the original data via the training device model update, preventing private data from leaking.

Next, the central server updates a global model by computing the average of local model updates and then sends the global updated model for the next training cycle to each device. The following may be stated in this connected learning model:

$$\min_{m \in \mathbb{R}^d} f(m) = \frac{1}{N} \sum_{i=1}^N f_i(m), \quad (2)$$

where  $N$  is the number of trainers.

Although the most important advantage of federated learning is the privacy protection of raw data, the size of the transmitted update parameters is significantly smaller

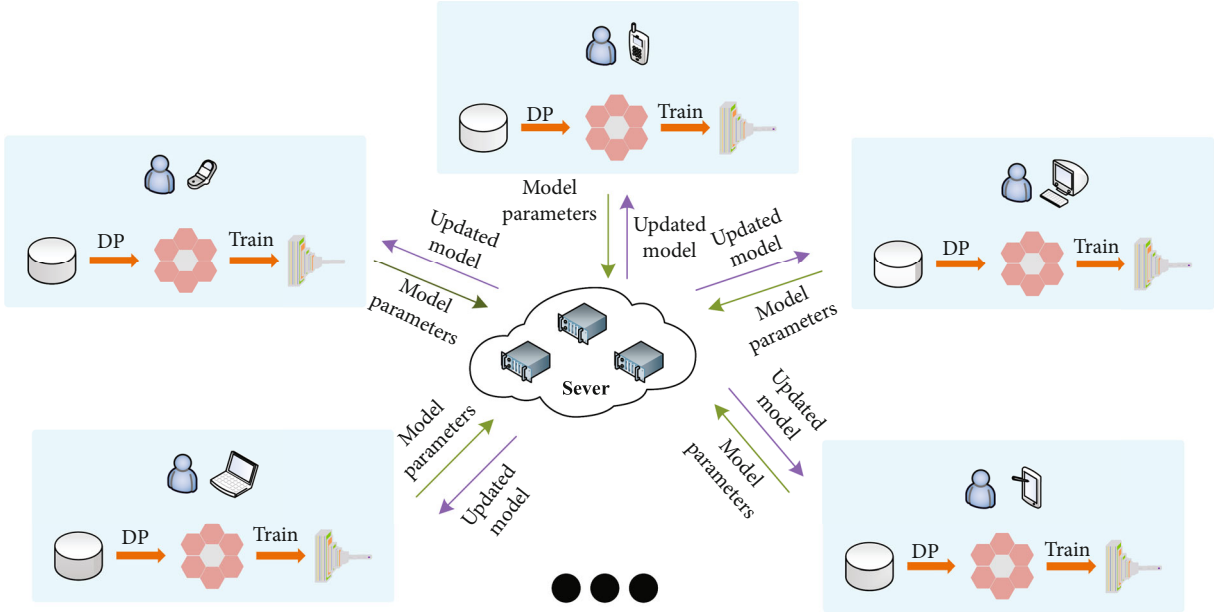


FIGURE 3: System model based on federated learning with differential privacy.

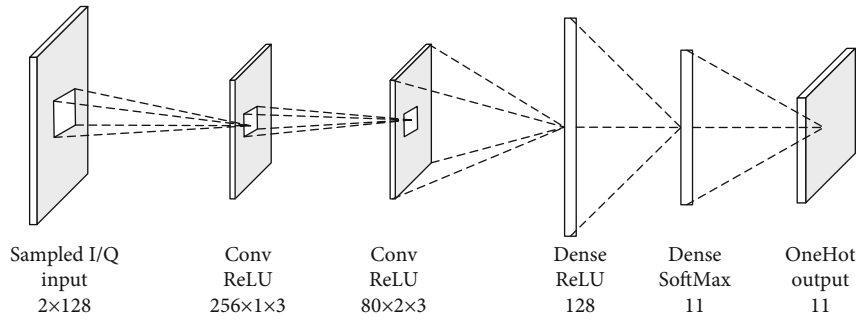


FIGURE 4: Local convolutional neural network.

```

1: initialization.
2: for each round  $t = 1, 2, \dots$  do
3:    $S_t \leftarrow$  Select a subset of  $N$  devices
4:   Broadcast global model  $m^{t-1}$  to each device in  $S_t$ 
5:   Do differential privacy processing to get  $\{D_{1^t}, D_{2^t}, \dots, D_{K^t}\}$ .
6:   for each device  $k \in S_t$  in parallel
7:      $M_k^t \leftarrow LocalUpdate(k, M^{t-1})$ 
8:   end for
9:   Transmit  $\{P_t^1, P_t^2, \dots, P_t^K\}$ 
10:  Aggregation  $P_{t+1}^{Global} = 1/K(\sum_{k=1}^K P_t^k)$ 
11: end for

```

ALGORITHM 1: Federated averaging algorithm.

than that of the original data during the process of federated learning and training, so communication costs and delays between devices and central servers are greatly reduced.

The convergence of the algorithm is an important characteristic of federated learning like virtually all algorithms which are distributed. However, federated learning's loss function does not ensure that it converges always. Reference

[25] studies the convergence of the loss function in the federated learning model on a theoretical basis. We discover that the loss function may be reduced and the precise accuracy can be greater if the local model is CNN with random gradient descent (SGD). We thus pick the locale training model CNN (SGD) and present the local training model in the next section.

**2.2. Differential Privacy.** In 2006, Dwork et al. first proposed the concept of differential privacy (DP). Differential privacy mainly protects personal information [26]. In other words, after differential privacy processing, if a personal record is not in a dataset, the attacker can obtain almost the same information. The concept of differential privacy proposed by Dwork et al. is powerful enough to protect data privacy. Moreover, differential privacy is in line with people's understanding of the protection of personal privacy information. No matter whether a record exists in a dataset, the attacker cannot get more information about the record, even if the attacker has other external information. Differential privacy can resist background knowledge attack. After differential privacy processing, each personal information of the dataset is independent of the output of the dataset query. What can be guaranteed is that personal privacy information will not be infringed.

The implementation process of differential privacy is to add noise and introduce randomness into the data. The purpose of introducing randomness is to reduce the risk of privacy leakage to the greatest extent when querying data, while ensuring certain query accuracy. This can bring a benefit, which can add noise quantitatively and achieve a good balance between data availability and privacy protection.

If the two probability output results of a given random function  $K$  on an adjacent dataset  $D_1$  and  $D_2$  satisfy the following inequality, then the random function  $K$  satisfies the differential privacy:

$$\Pr [K(D_1) \in S] \leq \exp(\epsilon) \Pr [K(D_2) \in S] + \delta. \quad (3)$$

Adjacent datasets refer to two datasets with one record difference at most; that is, one dataset is generated by adding or deleting one record from another dataset. In equation (3),  $\Pr [K(D_1) \in S]$  represents the probability of the output of function  $K$  on  $D_1$  in the range  $S$ , and the ratio of the two probability values is less than or equal to  $e^\epsilon$ .  $\epsilon$  is called privacy budget or privacy parameter, which is used to balance the degree of privacy protection and data utility. It can be seen from equation (3) that the smaller  $\epsilon$ , the more consistent the two probability values tend to be; that is, the existence of a single record does not affect the output result; the higher the degree of privacy protection; and correspondingly, the lower the data utility. Similarly, the larger  $\epsilon$ , the lower the degree of privacy protection and the higher the data utility. When  $\epsilon = 0$ , the adjacent datasets are output with the same probability distribution, which, of course, completely loses the data availability.

**2.3. Convolution Neural Network.** The CNN is a classic and frequently used deep learning structure that tackles some of the issues that were difficult to overcome in prior artificial intelligence [27]. Great breakthroughs were made in image processing, video recognition, and other domains, which might contribute to the present deep learning boom exactly because of these successes. Figure 2 illustrates the CNN structure.

CNN has a convolution structure and a deep neural network. In order to ease the model problem, convolution

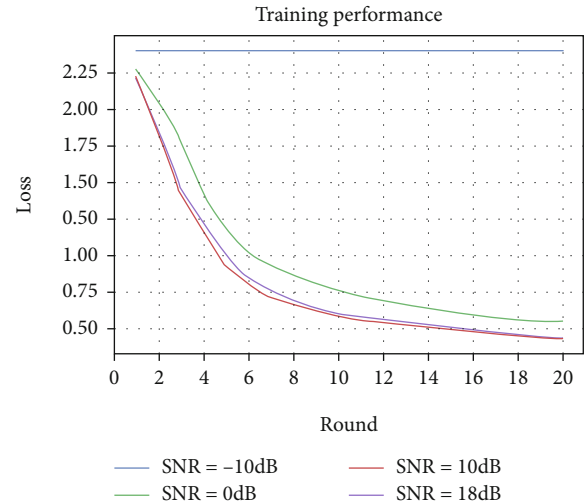


FIGURE 5: Training performance (different training signal SNRs).

structure can reduce the number of network parameters. Hidden layers are an important element of convolution neural networks. Normally, common CNN architectures contain input, convolution, fully connected, pooling, and output layers. CNN generally consists of many layers of convolution and pooling.

The preceding layer's feature mapping employs the learning convolution kernel to finish the convolution process. It is highly necessary to convert the kernel to a convolution layer. Functional extractor is the heart of the convolution kernel. Its major task is to automatically extract the deep data from the input signal. With the activation function, the output of the convolution result produces the neurons of this layer and therefore generates the functional map of this layer known as the functional extraction layer. In order to extract the local area properties, the local receptive zone of the former layer is linked to the input of every neuron.

The fully connected layer is the last layer of the network. The preceding layer-by-layer transformation and map extraction features conduct redynamic categorization and other processing. Usually, the ReLU function is the activation function of each neuron in the all linked layers. In order to achieve the classification function, the final output layer might employ SoftMax activation.

Current networks can learn a great deal of input-output mapping, understanding the exact mathematical connection between input and output. There are far more unlabeled data than labeled data in real applications. Simultaneously, manual data labelling also needs considerable effort. However, a number of labeled training data are necessary, which restrict the practical use of CNN to a certain degree in order to completely train the supervised CNN and to have higher generalizing capacities.

### 3. System Model

Considering the privacy security problems in the field of modulation recognition, we propose a modulation

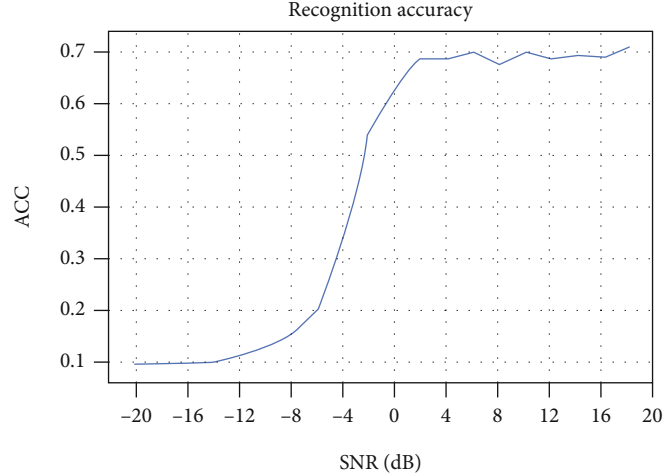


FIGURE 6: Recognition accuracy (different test signal SNRs).

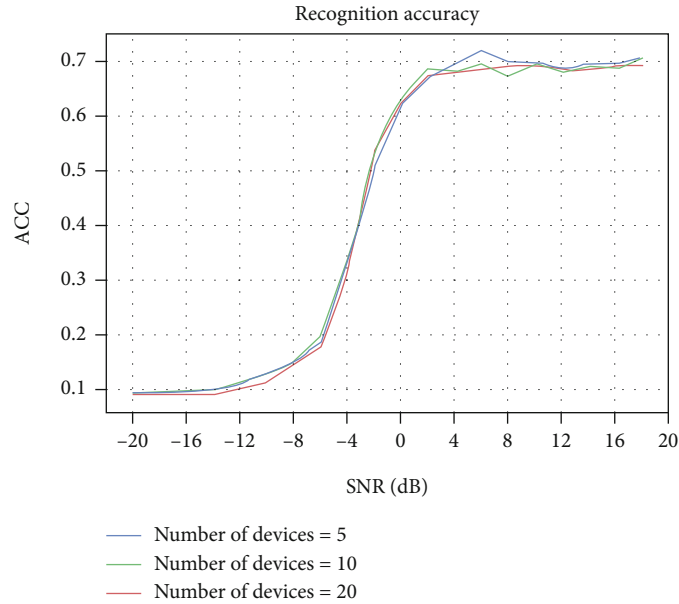


FIGURE 7: Recognition accuracy (different numbers of training devices).

recognition method based on federated learning under differential privacy protection.

Our differential privacy federated learning framework is shown in Figure 3. Each user has its own dataset; after differential privacy processing, each user updates the model locally. The cloud platform server collects the model parameters of users, updates and integrates the global model, and then sends the updated global model to users. Due to the equal amount of data for each user, the local model training process is basically consistent. The specific process is as follows:

- (1) Suppose there are  $K$  users, and each user has its own collected dataset  $\{D_1, D_2, \dots, D_K\}$ . Then, each user performs differential privacy processing on their own dataset to get  $\{D_1', D_2', \dots, D_K'\}$ . Finally, each user conducts the training of the deep learning

model locally. The trained deep learning models are denoted as  $\{M_1, M_2, \dots, M_k\}$

- (2) After each user trains locally, they can upload the model parameter  $\{P_t^1, P_t^2, \dots, P_t^K\}$  to the cloud platform server, where  $t$  represents the  $t$  round of interaction between the user and the cloud platform server and  $K$  represents the  $K$ -th user
- (3) The cloud platform server aggregates and integrates the parameter updates from each user to obtain

$$P_{t+1}^{\text{Global}} = \frac{1}{K} \left( \sum_{k=1}^K P_t^k \right). \quad (4)$$

The significance of aggregation is to integrate the



parameters of each user, which is more helpful for model optimization.

- (4) The cloud platform server transmits the aggregated and integrated global model parameters  $P_{t+1}^{\text{Global}}$  back to each user, and each user loads the updated model parameters into the deep learning model for the next round of training.

**3.1. Differential Privacy Implementation.** The implementation of differential privacy requires the introduction of a noise disturbance dataset. How much noise is added is related to the antinoise ability of the dataset. This antinoise ability is called global sensitivity (GS).

For any given query function, the sensitivity of function  $f$  is

$$\Delta f = \max \|f(D_1) - f(D_2)\|_1, \quad (5)$$

where  $f : D \rightarrow R^d$  represents mapping  $D$  to the  $d$ -dimensional real number domain space and  $D$  is the dataset. In equation (5),  $D_1$  and  $D_2$  are the adjacent datasets mentioned above or called sibling datasets, and  $\|\cdot\|_1$  is the 1-norm. To put it vividly, the global sensitivity represents the biggest difference obtained after adding or deleting a certain dataset; that is, it measures the sensitivity of the dataset to its modification. At present, the common mechanisms to realize differential privacy include the Laplace mechanism, Gaussian mechanism, and exponential mechanism. The Gaussian mechanism and Laplace mechanism are mainly aimed at numerical data, and the latter is mainly aimed at protecting labelled classified data. This paper uses the Laplace mechanism to make differential privacy dataset. The specific principles are as follows:

Laplace mechanism: for any given query function  $f : D \rightarrow R^d$ , if  $M(d)$  satisfies the output result of the following equation, the Laplace mechanism meets differential privacy:

$$M(D) = f(D) + \left( \text{Laplace} \left( \frac{\Delta f}{\epsilon} \right) \right)^d, \quad (6)$$

where  $\text{Laplace}(\cdot)^d$  is the  $d$ -dimensional Laplace distribution. It can be seen that the added noise level is proportional to  $\Delta f$  and inversely proportional to the privacy budget.

**3.2. Local Deep Learning Model.** The federated learning system's local training model structure is illustrated in Figure 4. The CNN network we employ is a four-layer network with two convolution layers. Other layers also employ the ReLU activation algorithm in addition to the final output layer with SoftMax. The data dimensions of the network are  $2 \times 128$ , as illustrated in Figure 4. The size of the kernel utilized in the first convolution layer is  $1 \times 13$  with 256 kernels. The second layer of the convolution layer utilizes a bigger  $2 \times 13$  kernel with 80 kernels.

After two layers of convolution, the complete connection layer, which includes 256 neurons, extracts additional global characteristics. Finally, for categorization, the final

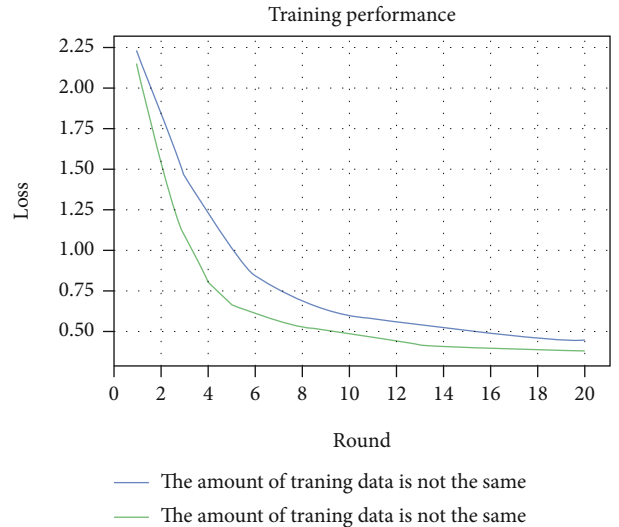


FIGURE 8: Training performance (different amounts of training data).

completely linked layer is employed. Since we utilize 11 modulation types in our dataset, we have 11 classification neurons in the output layer. The last neuron output is the probability in the current category of this input. The output with the highest probability is the outcome of categorization of current data after calculation of SoftMax activation function.

Algorithm 1 shows the above-mentioned federated training framework.

## 4. Simulation Results

We conducted several simulated tests in this section utilizing Google's federal learning framework (TFF) to assess the availability of signal modulation recognition based on federated learning. TensorFlow Federated (TFF) is an open-source framework for machine learning and other computations on decentralized data. In terms of hardware support, the CPU we use is Intel (R) core (TM) i7-9700 CPU @ 3.00 GHz, the memory is 32.0 GB, and the graphics card is NVIDIA geforce RTX 2080. We utilize the dataset suggested in [1] including 11 modulated  $[-20, +18]$  dB radio signals (each data sample has two raw I/Q channels data with a size  $2 \times 128$ ). Under the federated learning framework, 20 local trainings are conducted in each round, and such a round takes about 100 s. In numerous situations, simulations have been evaluated for signal modulation identification.

**4.1. Performance of Training Signals with Different SNRs.** We initially evaluated when there are 10 training equipment and all modulation kinds are contained in the data taught by each device. There are 900 signals in the same number of training data for each device. After 20 training rounds, federated learning loss curves on data with the SNRs -10 dB, 0 dB, +10 dB, and +18 dB are obtained, as illustrated in Figure 5.

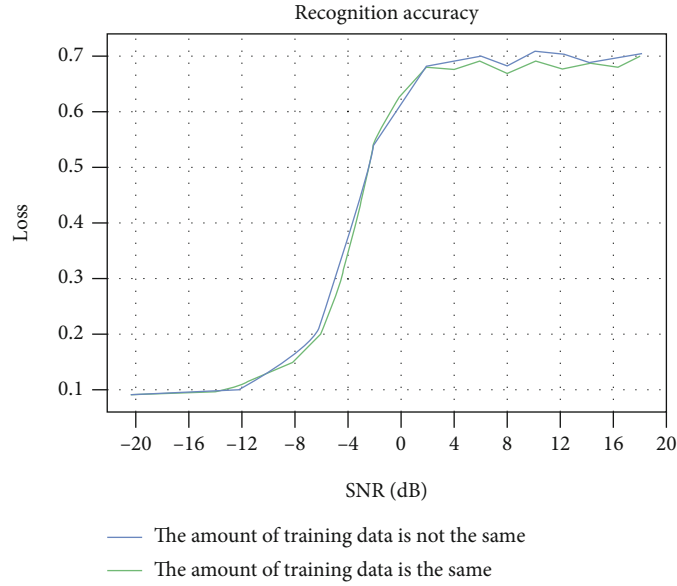


FIGURE 9: Recognition accuracy (different amounts of training data).

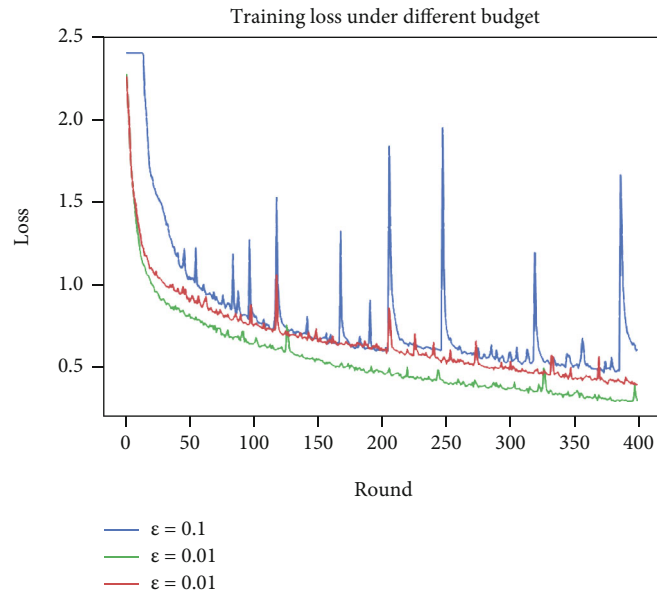


FIGURE 10: Training loss under different privacy budget.

We demonstrate that the loss of training is significantly greater and constant at 2.39 under the conditions of low SNR. The training loss is comparable, at least 0.44, in high SNR circumstances. The training loss can reach at least 0.54 in the 0 dB SNR situation. This shows that in the low SNR setting, the training performance is poor. Noise and interference can be caused by disguising the signal properties in a low SNR situation. But our system training performance has increased significantly in the 0 dB SNR or greater SNR environment.

**4.2. Recognition Accuracy of Test Data with Different SNRs.** Consider that 10 training devices are available, and the training data of each device contains all modulation types. Each

device has 900 signals and is adjusted to +18 dB for the quantity of training data. Following 20 training rounds, the accuracy of the test data accuracy of various SNRs is displayed in Figure 6.

Experimental results indicate that the accuracy of the modulation recognition is 62.96% at 0 dB SNR, and the accuracy of the identification drops quickly at the SNR of 0 dB. The accuracy of identification may reach 70.61% in the high-SNR situation. This is equivalent to the results of the proposed approach [1].

**4.3. The Impact of Different Numbers of the Training Device on Federated Learning Performance.** We assume next that each device's training data includes all modulation modules

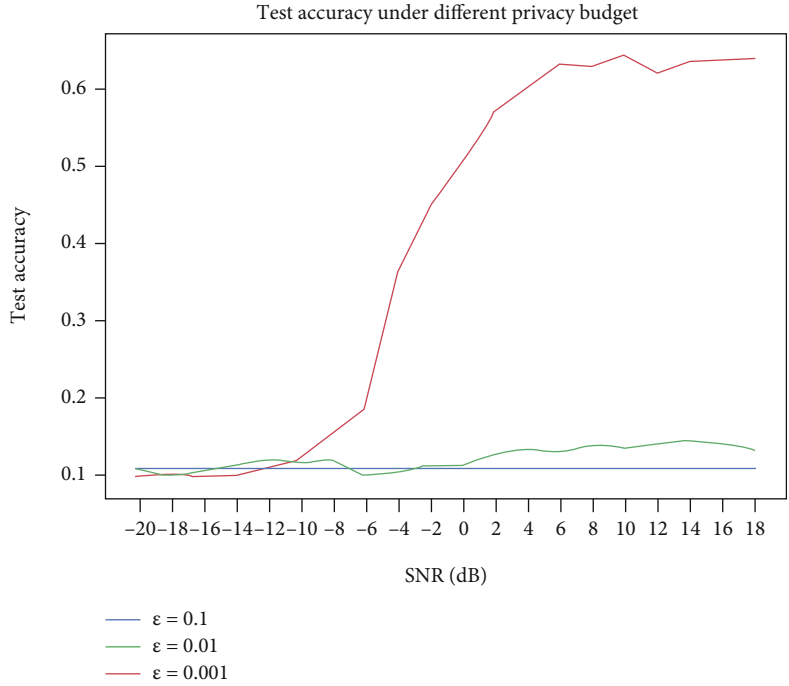


FIGURE 11: Test accuracy under different privacy budgets.

while each device has the same quantity of training data and the SNR is set at +18 dB. The number of trainers is 5, 10, and 20, and the accuracy of the detection curve is given in Figure 7 after 20 training rounds. The accuracy curve of recognition shows that the training performance of a federated learning system with diverse training equipment ranges from -20 dB to +12 dB of SNR which is practically equal. The federated learning system with five training devices is subject to SNR = +6 dB and has the best accuracy in accuracy which is 71.98%.

With SNR +18 dB, the 20 training devices in the federated learning system had the highest accuracy (69.19%). This demonstrates that the higher the training devices, the better the training performance; the varied number of equipment may be employed for training in various settings. For example, five equipment can be selected for training in an environment where the SNR is +6 dB, to minimize the training costs while obtaining greater precision. If greater precision is needed at higher SNRs, 20 training devices will be selected. If the criteria for precision are not severe, 5 equipment can be taught to attain the necessary precision.

*4.4. The Impact of Different Amounts of Training Data on Performance.* Next, we investigate a case in which each device’s training data covers all forms of modulation and sets the SNR at +18 dB. However, there is no difference in the amount of data on the devices. The loss curves and the detection precision curves under different SNRs are, respectively, presented in Figures 8 and 9 after 20 rounds on 10 devices.

The results of the simulation show that if the quantity of training data from the device is identical, the formation loss is smaller, because the local model update is comparable in

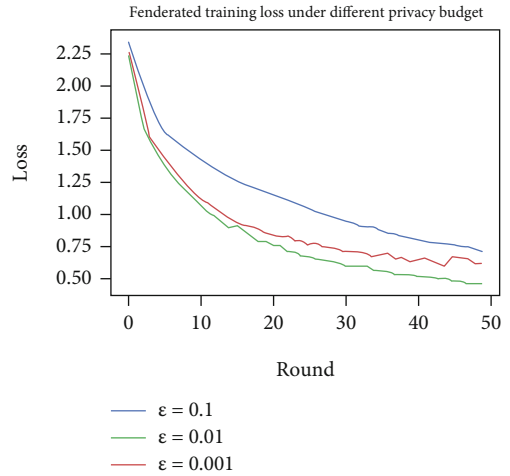


FIGURE 12: Federated training loss under different privacy budgets.

this scenario. The accuracy of recognition obtained by federated learning systems with diverse training data, however, is considerably higher than that of the former low SNR circumstances. The discrepancies between the training data can be caused. Some data, for example, have fewer characteristics while some have more features. Thus, it can increase the performance of federated learning to choose how much training data is.

*4.5. Performance Comparison of Differential Privacy Centralized CNN and Differential Privacy Federated CNN*

*4.5.1. Performance of Differential Privacy Centralized CNN.* According to the global sensitivity definition mentioned above, the global sensitivity of the original dataset used in

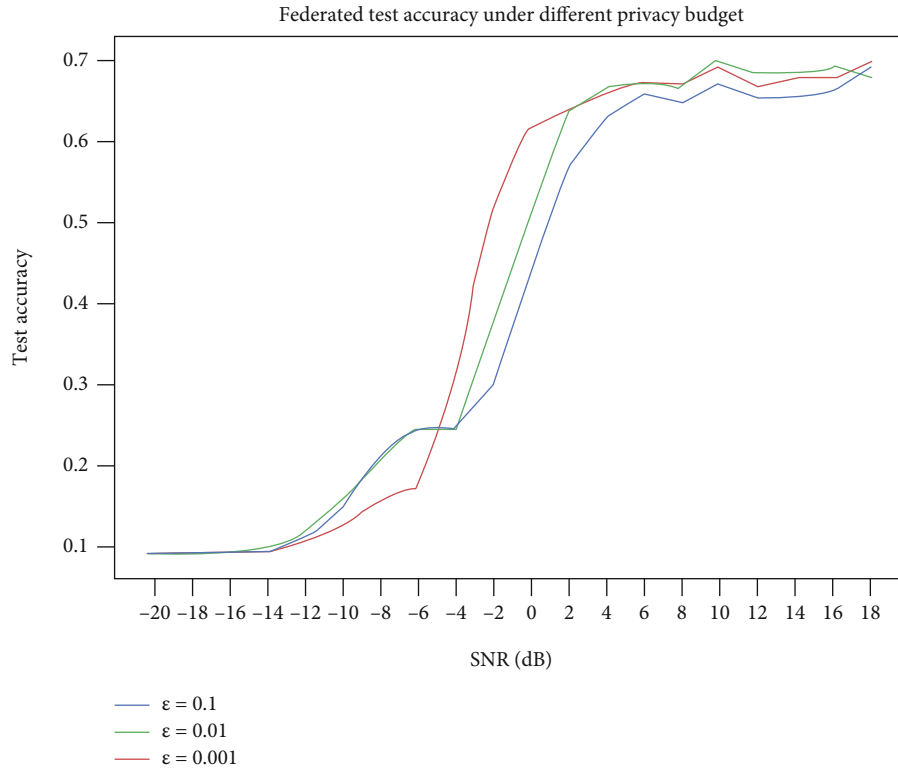
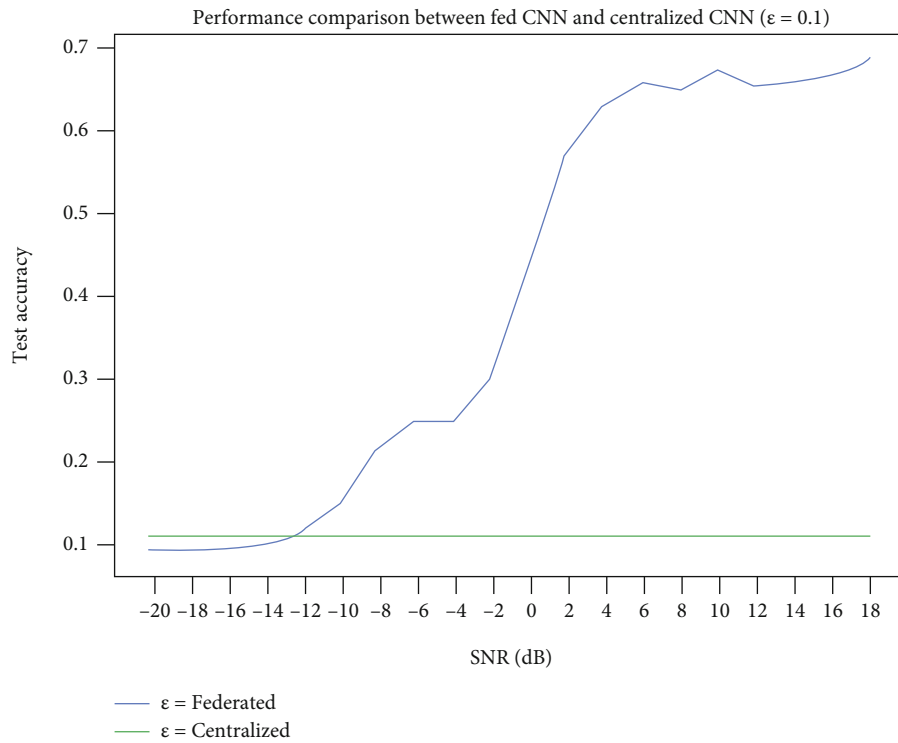


FIGURE 13: Federated test accuracy under different privacy budgets.

FIGURE 14: Performance comparison between federated CNN and centralized CNN ( $\epsilon = 0.1$ ).

this article is 0.417. For the convenience of calculation, we take the global sensitivity as 0.5. When  $\epsilon$  is 0.1, 0.01, and 0.001, the differential privacy budgets are 5, 50, and 500,

respectively. We first conducted experiments on the differential privacy dataset on a centralized CNN. The training loss curve and test accuracy curve under different privacy

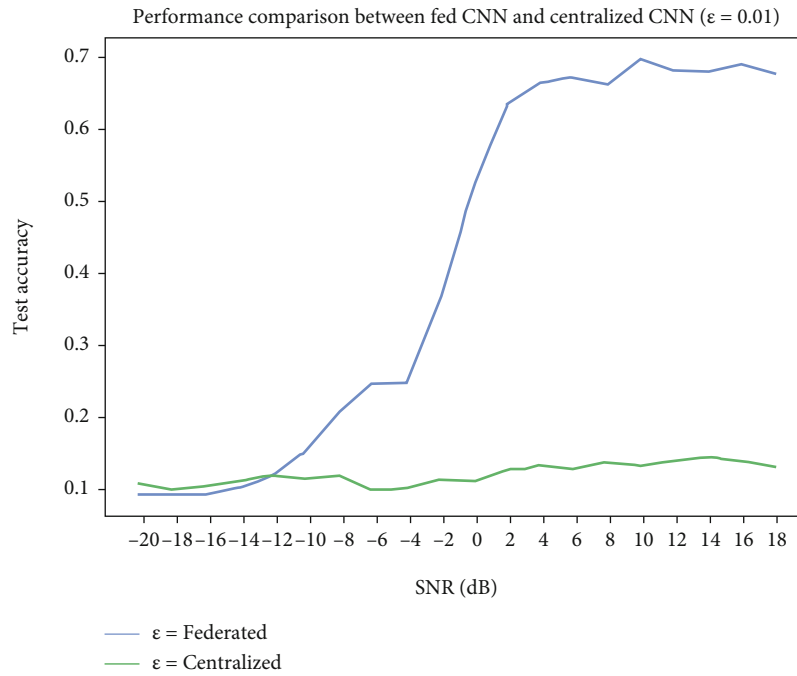


FIGURE 15: Performance comparison between federated CNN and centralized CNN (ε = 0.01).

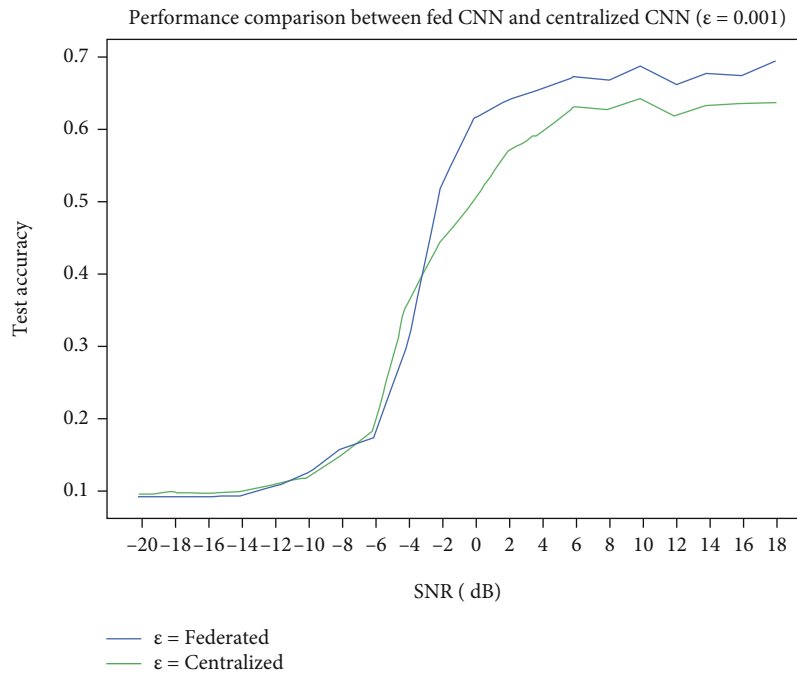


FIGURE 16: Performance comparison between federated CNN and centralized CNN (ε = 0.001).

budgets after 400-round training are shown in Figures 10 and 11, respectively.

It can be seen from the experimental results that as the privacy budget increases, the performance of the centralized CNN improves, but when the privacy budget is 5 and 50, the performance of the centralized CNN is very poor; that is, the utility of the dataset is very low. When the privacy budget is

500, the performance of the centralized CNN is acceptable, with the highest recognition accuracy rate of 63.9%.

*4.5.2. Performance of Differential Privacy Federated CNN.* Next, we conducted a differential privacy dataset experiment on federated CNN. The same as above, when epsilon is 0.1, 0.01, and 0.001, the differential privacy budget is 5, 50, and

500, respectively. Based on the above experimental results, we fixed the number of devices in the federated learning framework to 5, and each device dataset has the same size. After 50 rounds of federated training, the differential privacy federated training loss curve and test accuracy curve are shown in Figures 12 and 13, respectively.

The experimental results show that when the privacy budget is 50, the performance of the differential privacy federated CNN is the best, with the highest test accuracy rate of 69.9%. The performance when the privacy budget is 5 and 500 is close to the performance when the privacy budget is 50. Among them, the performance of the two at a low signal-to-noise ratio is better than the performance of privacy budget of 50, and at a high signal-to-noise ratio, the performance of the former two is slightly worse than the performance of privacy budget of 50. It proves that the differential privacy dataset has high data utility in the federated learning framework.

*4.5.3. Performance of Differential Privacy Federated CNN.* Finally, we compare the performance of federated CNN and centralized CNN under different privacy budgets. When the privacy budget is 5, 50, and 500, the test accuracy curves of federated CNN and centralized CNN are shown in Figures 14–16, respectively.

From the experimental results, it can be found that the differential privacy federated learning framework proposed in this paper has better performance than centralized differential privacy learning. When the privacy budget is 5 and 50, the performance of differential privacy federated learning is significantly higher than that of centralized differential privacy learning. Compared with nondifferential privacy federated learning, differential privacy federated learning can effectively protect data privacy and security, while also achieving a comparable recognition accuracy rate, ensuring high data utility.

## 5. Conclusion

In this article, we examined the federated learning and the feasibility of federated learning-based signal modulation recognition. On this basis, the signal modulation recognition based on differential privacy federated learning is studied. Federated learning performance has been assessed via simulation in five situations. It obtained a recognition rate of more than 70% under the premise of safeguarding privacy and security. This demonstrates the enormous potential of federated learning for signal processing. Federated learning is being implemented in more areas with more and more emphasis devoted to the security of data privacy worldwide.

## Data Availability

The RML2016.10a data used to support the findings of this study may be released upon application to the DEEPSIG, who can be contacted at info@deepsig.io.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (61771154) and the Fundamental Research Funds for the Central Universities (3072021CF0815). This work is also supported by the Key Laboratory of Advanced Marine Communication and Information Technology, Ministry of Industry and Information Technology, Harbin Engineering University, Harbin, China. A preprint has previously been published [23]. We added discussion and experimental verification on differential privacy content.

## References

- [1] T. J. O'Shea, J. Corgan, and T. Charles Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*, pp. 213–226, Springer, Cham, 2016.
- [2] Y. Tu, Y. Lin, J. Wang, and J. Kim, "Semi-supervised learning with generative adversarial networks on digital signal modulation classification," *CMC-Computers Materials & Continua*, vol. 55, no. 2, pp. 243–254, 2019.
- [3] Y. Lin, X. Zhu, Z. Zheng, Z. Dou, and R. Zhou, "The individual identification method of wireless device based on dimensionality reduction and machine learning," *The Journal of Supercomputing*, vol. 75, no. 6, pp. 3010–3027, 2019.
- [4] Y. Wang, M. Liu, J. Yang, and G. Gui, "Data-driven deep learning for automatic modulation recognition in cognitive radios," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4074–4077, 2019.
- [5] Z. Zhang, X. Guo, and Y. Lin, "Trust management method of D2D communication based on RF fingerprint identification," *IEEE Access*, vol. 6, pp. 66082–66087, 2018.
- [6] Y. Tu, Y. Lin, C. Hou, and S. Mao, "Complex-valued networks for automatic modulation classification," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10085–10089, 2020.
- [7] Y. Lin, Y. Tu, Z. Dou, L. Chen, and S. Mao, "Contour Stella image and deep learning for signal recognition in the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 34–46, 2021.
- [8] Y. Lin, Y. Tu, and Z. Dou, "An improved neural network pruning technology for automatic modulation classification in edge devices," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5703–5706, 2020.
- [9] Y. Wang, G. Gui, H. Gacanin, T. Ohtsuki, H. Sari, and F. Adachi, "Transfer learning for semi-supervised automatic modulation classification in ZF-MIMO systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 2, pp. 231–239, 2020.
- [10] M. Liu, K. Yang, N. Zhao, Y. Chen, H. Song, and F. Gong, "Intelligent signal classification in industrial distributed wireless sensor networks based industrial Internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4946–4956, 2021.

- [11] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: recent advances and future challenges," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 790–807, 2021.
- [12] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6G: opening new horizons for integration of comfort, security, and intelligence," *IEEE Wireless Communications*, vol. 27, no. 5, pp. 126–132, 2020.
- [13] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, "Adversarial attacks in modulation recognition with convolutional neural networks," *IEEE Transactions on Reliability*, vol. 70, no. 1, pp. 389–401, 2021.
- [14] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
- [15] N. H. Tran, W. Bao, A. Zomaya, M. N. H. Nguyen, and C. S. Hong, "Federated learning over wireless networks: optimization model design and analysis," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1387–1395, Paris, France, 2019.
- [16] J. Konečný, H. McMahan, F. Yu, P. Richtárik, A. Suresh, and D. Bacon, "Federated learning: strategies for improving communication efficiency," <https://arxiv.org/abs/1610.05492>.
- [17] K. Yang, T. Jiang, Y. Shi, and Z. Ding, "Federated learning based on over-the-air computation," in *ICC 2019-2019 IEEE international conference on communications (ICC)*, pp. 1–6, Shanghai, China, 2019.
- [18] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography*, Springer, Berlin, Heidelberg, 2006.
- [19] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," 2018, <https://arxiv.org/abs/1802.06739>.
- [20] J. Lee and D. Kifer, "Concentrated differentially private gradient descent with adaptive periteration privacy budget," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1656–1665, London, United Kingdom, 2018.
- [21] Y. Yan, L. Zhang, Q. Z. Sheng, B. Wang, X. Gao, and Y. Cong, "Dynamic release of big location data based on adaptive sampling and differential privacy," *IEEE Access*, vol. 7, pp. 164962–164974, 2019.
- [22] Y. Yan, B. Wang, Q. Z. Sheng, A. Mahmood, T. Feng, and P. Xie, "Modelling the publishing process of big location data using deep learning prediction methods," *Electronics*, vol. 9, no. 3, p. 420, 2020.
- [23] J. Shi, H. Zhao, M. Wang, and Q. Tian, "Signal recognition based on federated learning," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1105–1110, Toronto, ON, Canada, 2020.
- [24] K. Bonawitz, V. Ivanov, B. Kreuter et al., "Practical secure aggregation for federated learning on user-held data," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, Dallas Texas, USA, 2017.
- [25] S. Wang, T. Tuor, T. Salonidis et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.
- [26] G. Gong, C. Lessoy, C. Lu, and K. Lv, "Differential privacy spatial decomposition via flattening Kd-tree," *International Journal of Performability Engineering*, vol. 16, no. 7, pp. 1058–1066, 2020.
- [27] D. Bhavana, K. Kishore Kumar, M. Bipin Chandra, P. V. S. K. Bhargav, D. J. Sanjana, and G. M. Gopi, "Hand sign recognition using CNN," *International Journal of Performability Engineering*, vol. 17, no. 3, pp. 314–321, 2021.

## Research Article

# LstFcFedLear: A LSTM-FC with Vertical Federated Learning Network for Fault Prediction

Xiangquan Zhang<sup>1</sup>, Zhili Ma,<sup>1</sup> Anmin Wang,<sup>2</sup> Haifeng Mi,<sup>2</sup> and Junjun Hang<sup>3</sup>

<sup>1</sup>State Grid Gansu Electric Power Company, Lanzhou 730030, China

<sup>2</sup>State Grid Baiyin Power Supply Company, Baiyin 730900, China

<sup>3</sup>Huainan Normal University, Huainan 232001, China

Correspondence should be addressed to Xiangquan Zhang; [nhuang1111@163.com](mailto:nhuang1111@163.com)

Received 18 May 2021; Revised 5 July 2021; Accepted 5 August 2021; Published 28 August 2021

Academic Editor: Jinbo Xiong

Copyright © 2021 Xiangquan Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The firefighting IoT platform links multiple firefighting subsystems. The data of each subsystem belongs to the sensitive data of the profession. Failure prediction is a crucial topic for firefighting IoT platforms, because failures may cause equipment injuries. Currently, in the maintenance of fire IoT terminal equipment, fault prediction based on equipment time series has not been included. The use of intelligent technology to continuously predict the failure of firefighting IoT equipment can not only eliminate the intervention of regular maintenance but also provide early warning of upcoming failures. In order to solve this problem, we propose a vertical federated learning framework based on LSTM fault classification network (LstFcFedLear). The advantage of this framework is that it can encrypt and integrate the data on the entire firefighting IoT platform to form a new dataset. After the synthesized data is trained through each model, the optimal model parameters can be finally updated. At the same time, it can ensure that the data of each business system is not leaked. The framework can predict when IoT equipment will fail in the future and then provide what measures should be used. The experimental results show that the LstFcFedLear model provides an effective method for fault prediction, and its results are comparable to the baseline.

## 1. Introduction

The firefighting IoT platform is one of the key safeguards for enterprise fire safety. However, the current fire Internet of things platform has low accuracy in identifying various types of alarm information. How to effectively identify the false alarm information of the fire Internet of things platform is very important. The current firefighting IoT platform is linked to multiple firefighting subsystems, such as smoke and sprinkler sensors in the office area and power environment monitoring in the substation. Since the data belongs to different business departments, the data of each department is expected to run on their own independent systems, which requires that each data cannot interact with other data. However, in order to improve the accuracy of the false alarm prediction of the fire Internet of things platform, it is necessary to use all the business data to train the network model. Based on this, we introduced a federated learning framework

and proposed the LstFcFedLear network. The accuracy of fault prediction is one of the keys to ensure the normal operation of the fire IoT. Predictive science is a discipline that analyzes a large amount of data and discovers some potential relevance among them. This provides an important basis for industry equipment failure prediction [1, 2]. In order to further improve the accuracy of various failure predictions, many new technologies (for example, artificial intelligence, big data, and blockchain) have been gradually applied to factories [3, 4].

In essence, failure prediction is to correlate the occurrence of failure events with the failures that may occur in the future. Failure prediction has made important developments in 1979. The corresponding mathematics of fault prediction is to use input to predict output. Due to the widespread existence of nonlinearity and uncertainty, it is more difficult to establish an efficient system model in mathematics, which is also one of the objective reasons for missed



detection and false alarms. Later, Box et al. proposed the application of time series to forecasts, which greatly improved the accuracy. The characteristic of the neural network is that it can perform nonlinear mapping, so it is widely used in the field of prediction. Unfortunately, neural networks need to be data-driven, and the biggest thing is that they need to manually set the network model parameters.

From the current enterprise monitoring system, it is relatively easy to obtain a large amount of historical equipment data and operating data. Therefore, it is feasible to use historical data to predict failures. With the rapid development of new technologies, it is also feasible to use artificial intelligence, big data, and other technologies to assist in forecasting. At present, in the use of artificial intelligence for fault prediction research, a supervised or unsupervised method is one of the two most common methods. For the design of the network structure, the designer can only rely on experience to subjectively design the depth of the network, the number of neurons, and other parameters. Designers with different experience design different networks. This leads to a problem, and the same problem may have different solutions. Among the many algorithm models, the support vector machine algorithm is more widely used.

If the condition of all equipment in the factory can be monitored and the failure can be alerted in advance, the reliability and stability of the entire factory can be greatly increased. In recent years, in the field of PHM, a lot of research on these fault topics has been carried out, which greatly reduces the cost of fault maintenance of factory equipment and also improves the efficiency of the factory [5]. The key function of PHM is to diagnose equipment failures and discover the causes of equipment failures [6]. Equipment failure prediction is very challenging, and the main reason is the need to consider both the maintenance plan and the type of failure. Over the years, the forecasting model has been continuously developed and improved. But so far, the complex algorithm model [7, 8] still has many limitations. In order to overcome these shortcomings, some studies have adopted machine learning algorithms, such as neural networks [4] and support vector machines (SVM) [5] to predict failure types. These studies have promoted the development of probabilistic models to a certain extent [9, 10], but probabilistic models lack clear physical meaning in fault prediction.

Compared with traditional machine learning algorithms that cannot process time series data, the advantage of the LstFcFedLear method is that it can predict the sequence of future data through learning from historical experience. The main contributions of this article to our work are summarized as follows:

- (1) We propose a vertical federated learning framework based on LSTM fault classification network (LstFcFedLear). The advantage of this framework is that it can encrypt and integrate the data on the entire firefighting IoT platform to form a new dataset
- (2) The LstFcFedLear model can ensure that the data of each business system is not leaked. This framework can predict the probability of future failure of the fire

IoT and can provide corresponding measures to solve the failure

The structure of this article is as follows: Section 2 introduces related research. Section 3 introduces the new framework method. Section 4 shows the experimental verification results. Section 5 is the conclusion and future work.

## 2. Related Works

VSC and MMC lack the ability to regulate DC short-circuit current during DC faults. For multiterminal DC system fault detection, the calculation of short-circuit current during the discharge phase of the DC fault capacitor is crucial. Li et al. proposed a transient equivalent model suitable for fault analysis of multiterminal DC systems. This model only retained the high-frequency components in the original fault network, which greatly simplified the circuit analysis at the initial stage of the fault [11]. Since the current waveform when the arc fault occurs is very similar to the current waveform of some loads, it is difficult to detect arc faults through simple current characteristics. Aiming at this problem, Lin et al. proposed an arc fault detection method combining a self-organizing feature mapping network and a sliding window method [12]. On the basis of autonomously mining the inherent characteristics of current data, the current signal is continuously detected by using the correlation and continuity between adjacent periodic current samples. The proposed method can effectively realize arc fault detection, and the accuracy of arc fault detection can reach 99%.

The existing bearing fault alarm system is mainly based on the rule diagnosis of a single shaft temperature variable, and the alarm is not timely. In response to the above problems, Liu et al. combined the correlation of the multiaxis axle temperature of the same car and proposed a data-driven method for detecting and positioning train bearing faults [13]. The proposed DiCCA modeling method is verified by using the axle temperature data of a train in actual operation, and the results show the effectiveness of the proposed method. Based on a data-driven approach, Xiong et al. proposed an edge-assisted privacy protection original data sharing framework, which ensures that the data connected to autonomous vehicles will not be destroyed [14].

Using the sensor data of the traction system, Chen et al. proposed an optimal data-driven fault detection method to solve the fault problem of the dynamic traction system [15]. And based on the improved SVM, the optimal data-driven fault diagnosis problem is studied. Finally, through the actual high-speed train experimental platform, the rationality and effectiveness of the proposed method are verified. Yang et al. proposed a data-driven soft closed-loop fault-tolerant control strategy for the voltage sensor failure of the DC side capacitor in the H-bridge structure of STATCOM [16]. This method selected capacitor voltage and system output current as original signals and established the MLS-SVM prediction model based on historical operating data [17]. The predictive output of the MLS-SVM model and the residual signal output by the actual sensor are used to establish a sensor fault detection and judgment mechanism.

The test results showed that the method has good accuracy and real-time performance [18].

Condition monitoring and fault diagnosis are necessary means to ensure the safe and stable operation of mechanical equipment. Wang et al. proposed a deep learning framework based on ABiLSTM for intelligent fault diagnosis of mechanical equipment [19]. The framework first preprocesses the raw data collected by the sensor and divides it into a training sample set and a test sample set. Secondly, Oramas and Tuytelaars extracted features of the original time-domain signal by training multiple bidirectional LSTM networks of different scales and obtained multiscale features of equipment failures [20]. The experimental results show that the ABiLSTM model can achieve multiscale feature extraction of the original signal. By comparing with methods such as CNN, DAE, and SVM, the fault recognition performance of the ABiLSTM model is better than that of various common models [21, 22]. The results of generalization performance experiments on the ABiLSTM model show that the fault recognition accuracy of samples under off-changing conditions can still reach more than 95%; the LSTM network architecture is shown in Figure 1.

### 3. Materials and Methods

**3.1. LSTM Network.** A Recurrent Neural Network (RNN) is a type of neural network specially used to process time series data samples. Each layer of RNN not only outputs to the next layer but also outputs a hidden state. RNN's convolutional neural network can be easily extended to images with a large width and height, and some convolutional neural networks can also handle images of different sizes. RNN can be extended to longer sequence data, and most RNNs can handle data with different sequence lengths. It can be seen as a fully connected neural network with self-loop feedback. In forward propagation, the hyperbolic tangent activation function is generally used from the input layer to the hidden layer [23]. The hidden layer to the output layer uses softmax to map the output to a probability distribution of (0, 1). We will see that the output value of the hidden layer at the current moment is affected not only by the input at the current moment but also by the input at all times in the past. In this way, the output value of the hidden layer can be regarded as the memory of the network, which makes it very suitable for processing data samples that have dependencies before and after [24, 25]. An important feature of RNN is that the parameters of the model are shared at different times. This allows us to share the statistical strength of different locations over time. When some parts of the sequence data appear in multiple locations, this parameter sharing mechanism becomes particularly important [26, 27]. LSTM is a type of RNN. The timing backpropagation algorithm transmits the error information step by step in the reverse order of time. When the length of each time series training data is large or the time is small, the gradient of the loss function with respect to the hidden layer variable at a certain time is more likely to disappear or explode.

The input vector of a standard RNN network is  $x = (x_1, \dots, x_T)$ . The RNN network uses equations (1) and (2) to solve

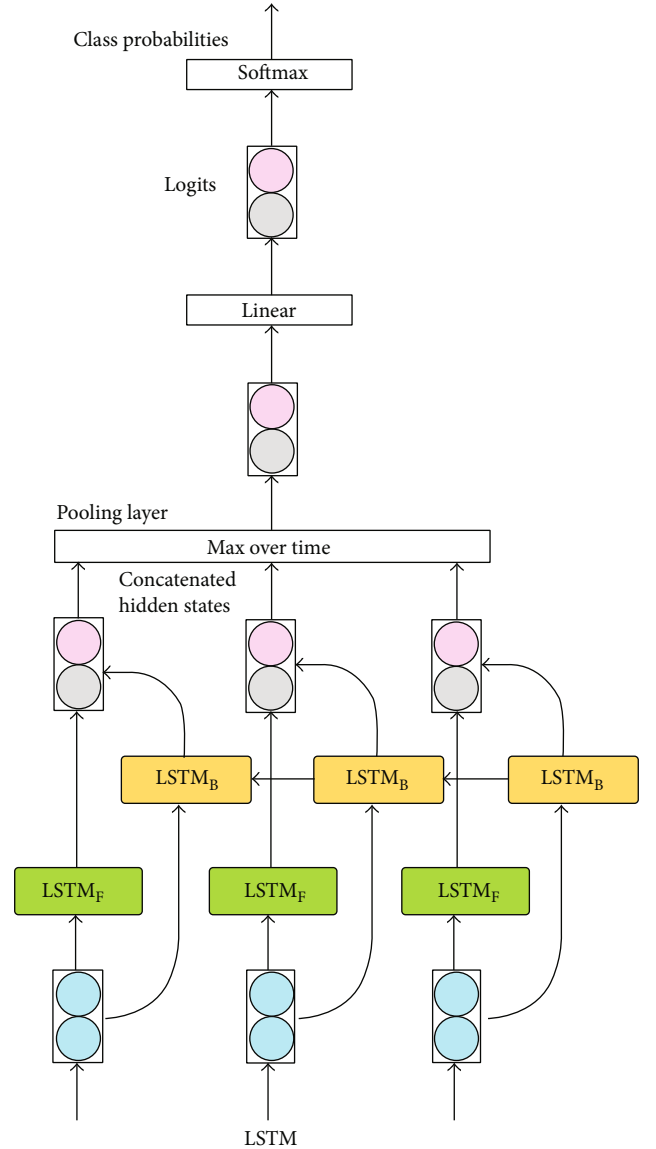


FIGURE 1: The LSTM network architecture.

the hidden vector  $h = (h_1, \dots, h_T)$  and the output vector  $y = (y_1, \dots, y_T)$ .

$$h_t = \sigma(W_{ih}x_t + W_{hh}h_{t-1} + b_h), \quad (1)$$

$$y_t = W_{ho}h_t + b_o. \quad (2)$$

Among them,  $W_{ih}$  refers to the input weight matrix.  $W_{hh}$  refers to the weight of the hidden layer.  $W_{ho}$  refers to the calculated output matrix of the hidden layer.  $b_h$  and  $b_o$  refer to all bias vectors, and  $\sigma$  is usually set as a sigmoid function  $\sigma(x) = 1/(1 + \exp(-x))$ .

The biggest problem encountered by RNN is the gradient problem of gradient disappearance and gradient explosion. LSTM is one of the RNN architectures, essentially using memory cells and gate cells to solve the problem of gradient disappearance and gradient explosion. The memory cell function of LSTM focuses on the input gate unit, which can

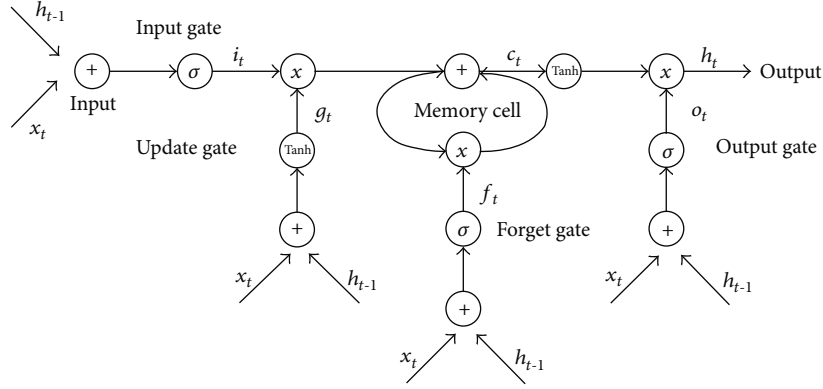


FIGURE 2: The process of the LstFcFedLear network.

make the state of each memory cell free from external interference. Each multiplication forget gate allows the memory to filter out irrelevant storage contents. The activation function of each multiplication forget gate is essentially an application program, which is mainly used for the calculation of the state of the internal memory unit. This article uses the LSTM structure proposed by Gers et al. The results of the memory unit and gate unit are shown in

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \quad (3)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \quad (4)$$

$$c_t = f_t c_{t-1} + i_t \tan h(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \quad (5)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o), \quad (6)$$

$$h_t = o_t \tanh(c_t). \quad (7)$$

Among them, the input vectors are  $f_t$ ,  $o_t$ , and  $c_t$ , which correspond to the vectors of the input gate, forget gate, and output gate at time  $t$ . It is worth noting that we regard the vector with the same size as the hidden vector  $h_t$ . The weight matrix  $W$  refers to the connection coefficient matrix between two different bodies.

The LSTM network is mainly composed of 32 bidirectional units, followed by the 50% discard layer and the sigmoid activation function. This uses L2 regularization to prevent network overfitting. In the model training, the cross-entropy loss function and Adam optimizer are used to train and solidify each LSTM network. When a fault is evaluated, LSTM divides the output fault into 5 levels, from small to large (0-4).

The performance of the pure LSTM model is better than that of the hybrid model [28]. Figure 2 shows the structure of the proposed model. By extracting the characteristics of the sequence data and using them as the input of the convolutional neural network model, the spatial characteristics of the data can be obtained. In order to prevent overfitting, Figure 3 adds a layer to prevent overfitting.

**3.2. Data Preprocessing.** In this article, the premise is that we predict daily failures. To this end, we focus on the cluster location and date of occurrence. This turns the research ques-

tion into determining when a failure occurs. Since we are solving two different prediction problems here, namely, binary classification and regression, we consider the production process of each type of prediction input dataset.

To address the problem of fault type classification for fire facility, we used the Fault Type of Fire Facility (FTFF) dataset from the Firefighting Internet of Things platform database of China State Grid Gansu Electric Power Company. This dataset contains two subdatasets, namely, FTFF1 and FTFF2.

We study the real dataset of 15084 alarm records of power firefighting equipment recorded between 2019 and 2020 from fault in power fire facility maintenance. All the FTFF datasets contain Alarm Time (AT), Fault Type (FT), Failure Equipment (FE), Fault Location (FL), Municipal Units (MU), and Level 2 Units (L2U). Considering the large number of subjects in each of these two datasets, we used the FTFF1 dataset for training and the FTFF2 dataset for testing.

The problem of fault type classification was approached as a five-class classification problem, with the following classes: 0 for the Overdue Fault (OF), 1 for the Offline Fault (OLF), 2 for the Power Failure (PF), 3 for the Equipment Damage (ED), and 4 for the False Alarm (FA). For each input data from one discrete time point, if the classification model identified this input as being of class 0-4, then the fault type of this discrete time point would be set to 0-4 (i.e., the fault type was predicted every time point).

The original fault dataset is first transformed into a fault vector,  $v_R = (v_{R1}, \dots, v_{RT})$ . In the regression algorithm model, for the fault at a certain time  $t$ , the text is expressed as  $v_{Rt} = (v_{t1}, \dots, v_{ts})$ . Among them,  $v_{ts}$  refers to the fault in the  $s$ -th space at a certain time  $t$ . In the simplified neural network, the sigmoid and hyperbolic tangent function are usually integrated in the gate and used as the activation function. The purpose is to convert the input value to between 0 and 1, where 1 means it is worth paying attention to and 0 means not needing attention. On the other hand, the role of the tanh function is to adjust the network performance by compressing the value to between -1 and 1. LSTM-FC is very sensitive to causality.

**3.3. Prediction Models.** In the design of this article, we design two types of fault diagnosis models: binary classification and prediction. In this research, we develop two types of failure

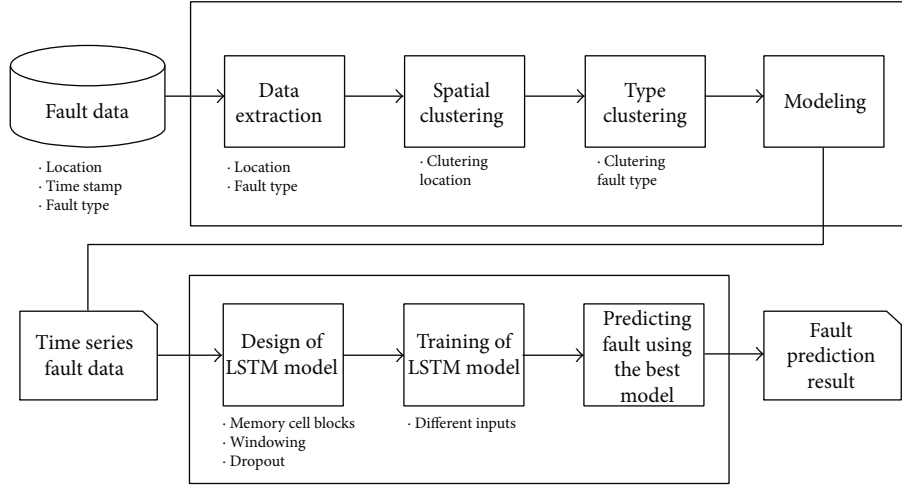


FIGURE 3: The proposed method for data preprocessing.

TABLE 1: The results of the forecasting model.

Model	Results	RMSE	SDE
Specificity = $\frac{TN}{(TN + FP)}$	MAE		
KNN	0.323	0.823	0.763
SVM	0.301	0.743	0.692
CNN	0.287	0.665	0.662
LstFcFedLear	0.226	0.619	0.634

prediction models, namely, binary classification and prediction. In addition, we also focus on evaluating the relationship between spatial clusters to judge the impact on the prediction results. In this article, we have designed four types of failure prediction models, as shown in Table 1. In the second section, we introduced that the RNN model can receive delay information and has the ability to judge whether this information has an impact on the storage unit. The proposed LSTM-FC model is shown in Figures 1 and 4. It can be seen that these two models use different types of input data, and the activation functions in the output layer are also different. It is worth noting that it expresses faults through weighting factors, and these weights are determined by the following equation:

$$e_t = h_t w_a, \quad (8)$$

$$a_t = \frac{\exp(e_t)}{\sum_{i=1}^T \exp(e_i)}, \quad (9)$$

$$v = \sum_{i=1}^T a_i h_i. \quad (10)$$

In these equations,  $h_t$  refers to the fault that occurs at time  $t$ .  $w_a$  is the weight matrix set by the attention layer.  $a_t$  refers to the probability of possible failure at time  $t$ .  $v$  refers to the weighted summation of the probabilities at all times  $t$ .

By converting the input into a fault sequence,  $X = \{x_1, x_2, \dots, x_N\}$ .  $x_t$  refers to the fault that occurs at time  $t$  calculated by the LSTM model. At each time step, we first use

the forward LSTM to predict the probability of the next failure. The overall goal is to minimize the following objective functions:

$$L_f = -\frac{1}{N} \sum_{t=1}^N \log \Pr(X_{t+1} | X_1, \dots, X_t). \quad (11)$$

Among them,  $L_f$  refers to all the parameters of the model in forward prediction. The  $\Pr(\cdot)$  function in the LSTM model is calculated as  $x_{t+1}$ , which mainly depends on the previous probability.

After getting a set of fault history data, the probability of the next fault can also be predicted through the reverse sequence. Therefore, we have also established a backward LSTM, the purpose of which is to predict the previous failure probability based on the later occurrence probability.

$$L_A = -\frac{1}{N} \sum_{t=N-1}^0 \log \Pr(a_t | a_N, \dots, a_{t+1}). \quad (12)$$

Finally, we analyze and classify the fault types and incorporate the key information into the whole process of LSTM training.

In summary, by using the optimized model, that is, equation (13), the parameters of the algorithm model can be obtained. After that, the corresponding topological quantities can be calculated.

**3.4. LstFcFedLear Model.** LSTM-FC can calculate the topological value of the genetic network. Using this advantage of LSTM-FC, the LSTM-FC network algorithm can be iterated repeatedly to obtain the most reasonable parameter matrix. In Algorithm 1, we show the algorithms of the LSTM-FC method one by one.

In order to be able to encrypt and integrate the data on the entire fire IoT platform to form a new dataset and to ensure that the data of each business system is not leaked, we have designed the following framework, as shown in Figure 5. Then, the new dataset is fed into the LstFcFedLear

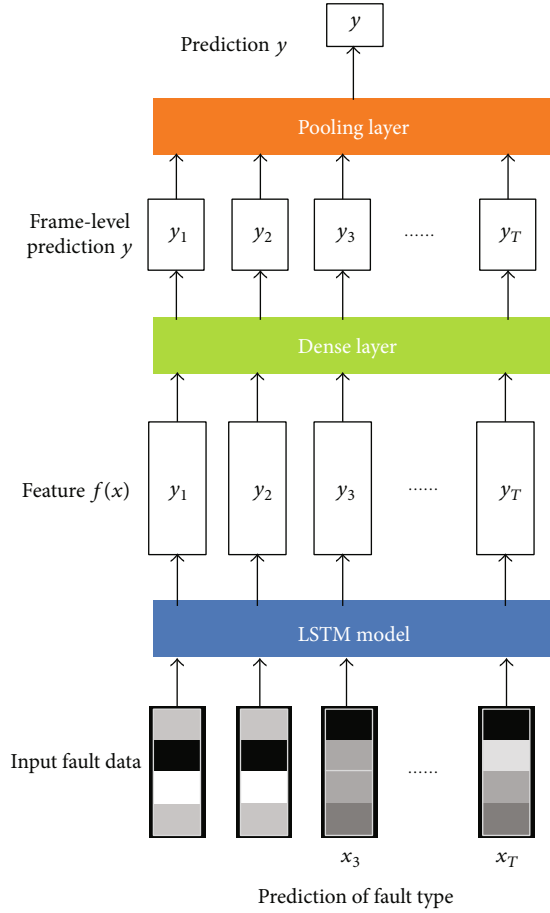


FIGURE 4: The prediction with LstFcFedLear from the dataset.

```

INPUT PARAMETERS: Reasonable labeling  $T_+, T_-$ 
Label labeling is not reasonable  $\sim T_+, \sim T_-$ 
Maximum number of pre-training
Maximum number of training
OUTPUT VALUE: Train well-performing models
1:  $d, c \leftarrow 0$ 
2: get  $x$  on  $T$ 
3: while  $d < \text{preEpoch}$  do
4:   for each fault  $i$  in  $T$  do
5:     get  $\tilde{r}+i, \tilde{r}-i$  in  $L$ 
6:     get  $L$  by (9) on  $i$ ;
7:     update  $\theta$ ;
8:     compute  $L_f$ 
9:     update  $\theta$ ;
10:  end for  $11 \leftarrow j + 1$ ; 12 end while
13: while  $k < \text{trainEpoch}$  do
15:   for each fault  $x_i$  in  $T$  do
16:     compute  $x_i$ ;
17:     compute  $y_i$ ;
18:     compute  $L_A$ ;
19:     update  $\theta$ ;
20:  end for
21:   $k \leftarrow k + 1$ ;
22: end while

```

ALGORITHM 1: LSTM-FC network algorithm.

model for training as a training set. The LstFcFedLear submodel corresponds to each fire subdata. The LstFcFedLear submodel is responsible for training each subdata to obtain the corresponding training parameters. Finally, all LstFcFedLear submodels update their parameters to a unified model. Specific steps are as follows:

Step 1. The central server sends the public key to the LstFcFedLear1, LstFcFedLear2, LstFcFedLear3, ..., LstFcFedLear models and uses the Paillier partial homomorphic encryption algorithm to align the encrypted samples. The Paillier encryption algorithm is mainly divided into three steps. The first is to generate a key according to the Paillier encryption algorithm. Then, use the generated key to encrypt each part of the data. Finally, after the model training is completed, the model is decrypted [29].

Step 2. The encrypted samples are fed to the LstFcFedLear1, LstFcFedLear2, LstFcFedLear3, ..., LstFcFedLear models for iterative training, and the local parameter gradients of the models are calculated, respectively.

Step 3. LstFcFedLear1, LstFcFedLear2, LstFcFedLear3, ..., LstFcFedLear models push the gradient and loss calculated by each to the central server. The central server uses the private key to decrypt.

Step 4. The central server sends the decrypted gradient and loss back to the LstFcFedLear1, LstFcFedLear2, LstFcFedLear3, ..., LstFcFedLear models.

Step 5. LstFcFedLear1, LstFcFedLear2, LstFcFedLear3, ..., LstFcFedLear models update the model parameters.

Step 6. LstFcFedLear1, LstFcFedLear2, LstFcFedLear3, ..., LstFcFedLear models are iteratively trained to generate a joint model.

## 4. Experiment Results

4.1. Dataset. To address the problem of fault type classification for fire facility, we used the Fault Type of Fire Facility (FTFF) dataset from the Firefighting Internet of Things platform database of China State Grid Gansu Electric Power Company. This dataset contains two subdatasets, namely, FTFF1 and FTFF2.

We study the real dataset of 15084 alarm records of power firefighting equipment recorded between 2019 and 2020 from fault in power fire facility maintenance. All the FTFF datasets contain Alarm Time (AT), Fault Type (FT), Failure Equipment (FE), Fault Location (FL), Municipal Units (MU), and Level 2 Units (L2U). Considering the large number of subjects in each of these two datasets, we used the FTFF1 dataset for training and the FTFF2 dataset for testing.

The problem of fault type classification was approached as a five-class classification problem, with the following classes: 0 for the Overdue Fault (OF), 1 for the Offline Fault (OLF), 2 for the Power Failure (PF), 3 for the Equipment Damage (ED), and 4 for the False Alarm (FA). For each input data from one discrete time point, if the classification model identified this input as being of class 0–4, then the fault type of this discrete time point would be set to 0–4 (i.e., the fault type was predicted every time point) [30–32].

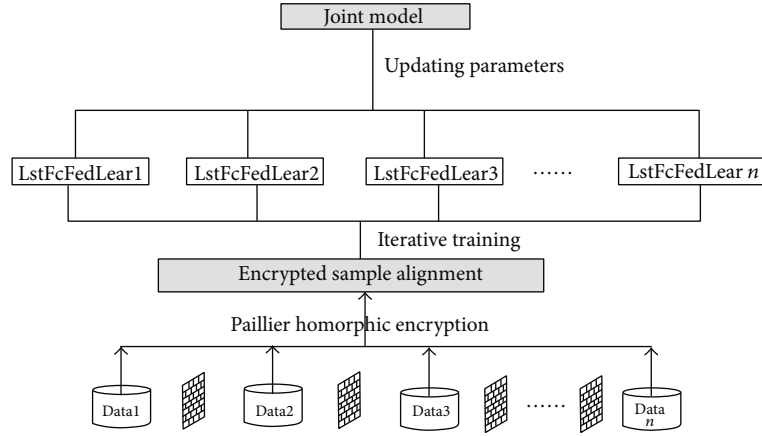


FIGURE 5: LstFcFedLear model framework.

**4.2. Prediction by LstFcFedLear.** It is easy to see that displaying a certain vector in a sequence or a certain sequence in a sequence is basically the same as the training process of the LstFcFedLear algorithm. The following figure illustrates the training progress curve of the LstFcFedLear and SVM algorithm in detail. It can be clearly seen from the figure that the loss and root mean square error performance of the two algorithm models in the training process are very similar. The loss curve can explain that the learning speed of SVM is relatively slow at the beginning of training. But it is worth noting that as time goes by, the learning curve of SVM is close to a certain value, and it has gradually stabilized. It can be inferred from these data that LstFcFedLear did not learn enough knowledge at the beginning, but over time, this problem was solved. In general, the performance of LSTM is slightly better than that of LstFcFedLear.

In order to further demonstrate the accuracy and generalization ability of LstFcFedLear, we compare its accuracy with the other three methods in [7–9]. In order to show the awareness of the results of the experiment, we use the method in [10]. The simulated test data was obtained using SynT-ReN, an environment frequently used in the industry. Figure 1 illustrates the operational characteristics (ROC) of the new data generated by LstFcFedLear and CNN. Obviously, on the synthesized test data, the accuracy of LstFcFedLear is better than that of the CNN method. As shown in Figure 6(b), compared with the FDR performance of SVM, KNN, and CNN methods, the error rate of LstFcFedLear is the lowest. This result clearly shows that for the genetic disease dataset, LstFcFedLear is better than SVM, KNN, and CNN.

As shown in Figure 6(c), the comparison between the positive prediction curve of LstFcFedLear and the PPV of SVM, KNN, and CNN shows that the LstFcFedLear method is optimal. This also further shows that LstFcFedLear is also superior to SVM, KNN, and CNN in terms of synthesizing genetic test data.

In this experiment, the LstFcFedLear model has 16 to 100 storage units. The training time interval of the model is [16, 100], and the unit is *s*. Throughout the experiment, the mean square error is the only indicator that measures the performance of binary classification and regression

models in the learning phase. In order to reduce the loss and increase the learning rate, the Adam optimizer is used in the LstFcFedLear model with the parameters  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . In order to prevent overfitting in the learning phase, a total of 3 times of cross-validation were used in this experiment. The pros and cons of the model's hyperparameters are the key to whether a model can achieve the best performance. In this experiment, we repeatedly test the hyperparameters of the LstFcFedLear model, such as the model's backtracking situation, the number of storage units, and the loss rate. The backtracking rate indicates how large the time interval to consider [33]. Table 2 shows the different dropout probability values in the experiment. What we want to explain here is that the loss function in the article represents the mean square error between the training value and the predicted value. In addition, it can be clearly seen from Table 2 that the accuracy of LstFcFedLear is as high as 94.6, which is 8.03% higher than the average of KNN, SVM, and CNN. The sensitivity of LstFcFedLear is as high as 93.4, which is 7.77% higher than the average of KNN, SVM, and CNN. The specificity of LstFcFedLear is as high as 95.1, which is 8.37% higher than the average of KNN, SVM, and CNN. These three indicators also show that LstFcFedLear is the best.

**4.3. Performance Comparison.** Here, we compare the performance of SVM, KNN, and CNN in detail with the performance of the LstFcFedLear model proposed in this paper. The experiment uses the scikit-learn package in Python for testing. In order to be able to make a thorough comparison with the performance of the LstFcFedLear model, a 3-fold cross-validation was specifically used in the experiment. For the best training parameters, grid search technology is also used in the experiment. It can be seen from the experimental results that for SVM, the value of the penalty parameter is set to  $c = 0.1$ . In Tables 1–3, we, respectively, compared the fault classification performance of KNN, SVM, CNN, and LstFcFedLear in detail. The comparison results show that the LstFcFedLear model has the best performance. In terms of accuracy and recall, LstFcFedLear is the best among these three. The second place is the CNN model. Table 2 mainly

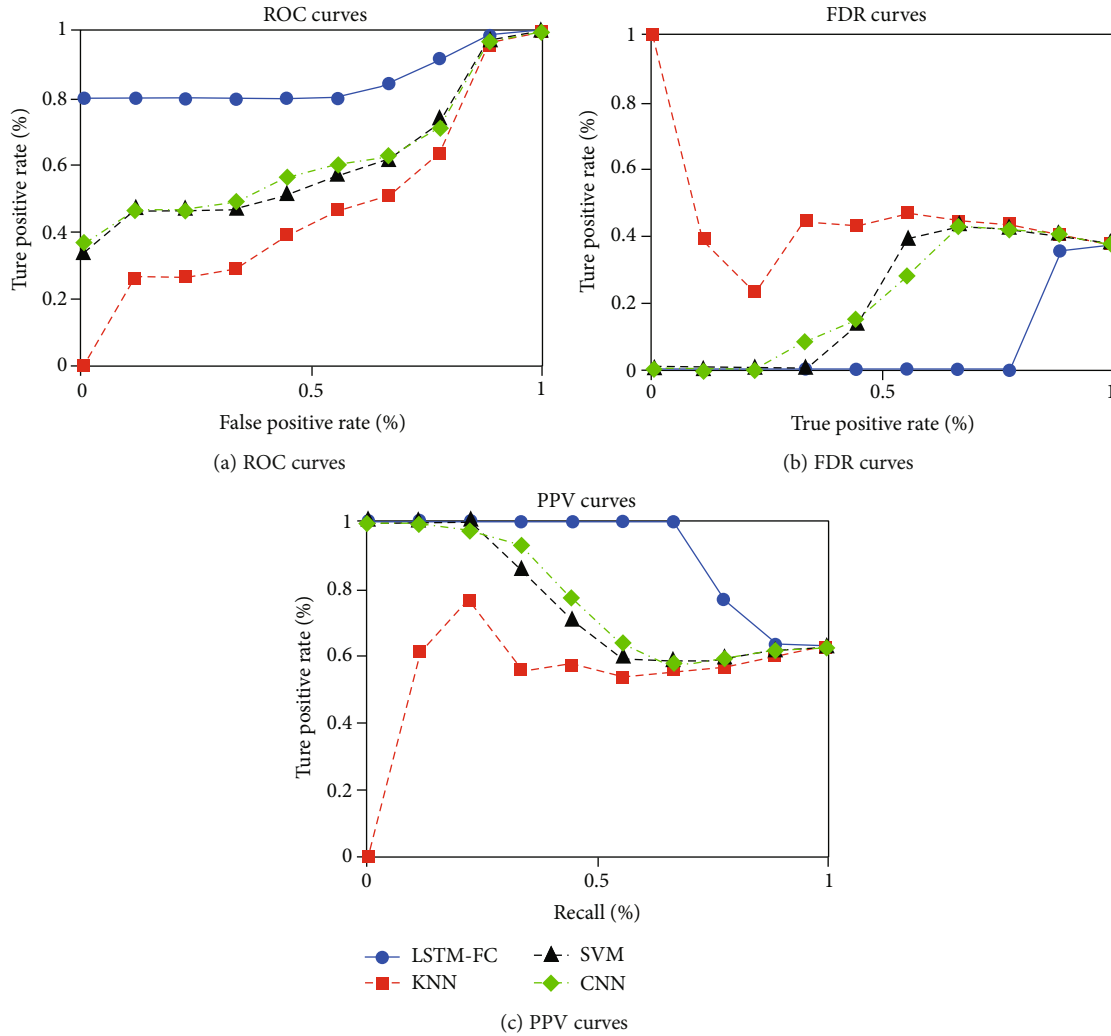


FIGURE 6: Accuracy and precision for prediction/classification among LstFcFedLear and different competitors based on fault datasets: (a) ROC diagram, (b) FDR chart, and (c) PPV graph.

TABLE 2: The comparison results between the traditional fault prediction method and the method proposed.

Method	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN	84.2	81.3	83.3
SVM	86.3	87.3	87.7
CNN	89.2	88.3	89.2
LstFcFedLear	94.6	93.4	95.1

TABLE 3: The running time of the different models in this paper between LstFcFedLear from SVM, KNN, and CNN.

Method	KNN	SVM	CNN	LstFcFedLear
Training time	4.209	8.703	787.342	983.306
Running time	2.217	4.332	9.243	7.276

compares the comparison results between the traditional fault prediction method and the method proposed in the article.

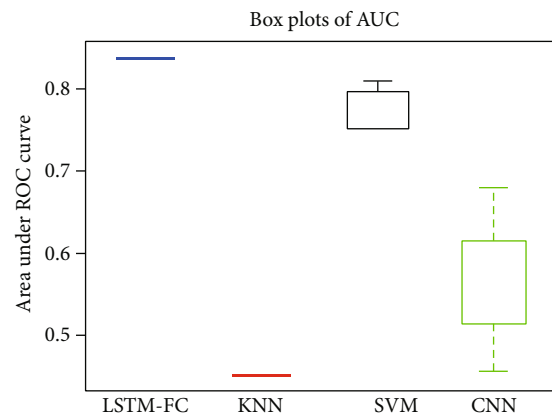


FIGURE 7: Graph of AUC box plot for LstFcFedLear.

The measurement indicators used in the experiment are sensitivity, accuracy, and Youden index scale, that is, true positive (TP), true negative (TN), false positive (FP), and false negative (FN) [28–31]. TP represents the number of

samples classified as correct. TN represents the number of samples judged to be false. FP represents the number of samples classified as incorrect. FN represents the number of samples classified as correct. The specific judgment formula is as follows:

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}, \quad (13)$$

$$\text{Sensitivity} = \frac{TP}{(TP + FN)}, \quad (14)$$

$$\text{Specificity} = \frac{TN}{(TN + FP)}. \quad (15)$$

As shown in Figure 6, the area enclosed by the curve has shown that LstFcFedLear is larger than the other three types. That can show that in terms of accuracy, LstFcFedLear is definitely better than the other algorithms. It can be seen from Figure 6 that the value of LstFcFedLear actually reaches the maximum average number AUC, about 0.84. But the AUC values of SVM, KNN, and CNN are 0.47, 0.78, and 0.59, respectively. The rankings are LstFcFedLear, KNN, CNN, and SVM. In addition, we calculated and visualized the area enclosed under the curve in Figure 6 in order to highlight the accuracy of all query methods. The AUC value obtained by the LstFcFedLear model is about 0.84, which can also indicate that the model is the best. More importantly, the average AUC owned by GlobalMIT is about 0.47, which is obviously much lower than that of LstFcFedLear.

As shown in Figure 7, the LstFcFedLear model has the best performance in classifying all faults into positive probability because the area under the ROC curve corresponding to LstFcFedLear is the largest. According to the area ranking under the ROC curve, it can be seen that the ranking of SVM is only lower than that of LstFcFedLear but is better than that of CNN and KNN in turn. It is worth noting that the ROC area of the LstFcFedLear model is 10 times that of KNN, 6 times that of SVM, and 3 times that of CNN. The huge area difference once again illustrates the excellent accuracy of the LstFcFedLear model.

Table 1 characterizes the accuracy of these algorithm models from another level. The MAE value of LstFcFedLear is 0.226, which is 0.075 lower than the average of KNN, SVM, and CNN. The test results show that the MAE value of KNN is the largest, indicating that the effect of the model is the worst. The ranking of other models from good to bad is CNN, SVM, and KNN. In terms of RMSE, the RMAE value of LstFcFedLear is 0.619, which is 0.123 lower than the average of KNN, SVM, and CNN. The test results show that the RMAE value of KNN is the largest, indicating that the effect of the model is the worst, which is consistent with the performance on the MAE value. The ranking of other models from good to bad is CNN, SVM, and KNN. In terms of SDE, the SDE value of LstFcFedLear is 0.634, which is 0.072 lower than the average of KNN, SVM, and CNN. The test results show that the SDE value of KNN is the largest, 0.763, indicating that the effect of the model is the worst, which is consistent with the performance on the MAE and RMSE values. The

ranking of other models from good to bad is CNN, SVM, and KNN.

In Table 3, we compare the running time of the LstFcFedLear, KNN, SVM, and CNN models. It can be seen that although LstFcFedLear is indeed superior in various performances, it pays relatively high in training time and running time costs. As can be seen in Table 3, in terms of training time cost, the time-consuming order is KNN, SVM, CNN, and LstFcFedLear from least to more. KNN is indeed very time-consuming, but its performance is too poor to be suitable for promotion and application in the industry. LstFcFedLear may take a little longer time, but it is relatively stable in terms of performance.

## 5. Conclusion

In short, we propose a vertical federated learning framework based on LSTM fault classification network to predict the failure of the fire IoT platform. The advantage of this framework is that it can encrypt and integrate the data on the entire firefighting IoT platform to form a new dataset. After the synthesized data is trained through each model, the optimal model parameters can be finally updated. At the same time, it can ensure that the data of each business system is not leaked. The experimental results showed that the LstFcFedLear model provides an effective method for fault prediction, and its results are comparable to the baseline. And the results among LstFcFedLear and SVM, KNN, and CNN methods showed that LstFcFedLear performs better than all methods in RMSE prediction, with the improvement being 9.8% and 24.3%, respectively. In the future, we plan to apply the LstFcFedLear model to power production application scenarios and then further optimize the robustness and other performance of the model.

## Data Availability

We used the Fault Type of Fire Facility (FTFF) dataset from the Firefighting Internet of Things platform database of China State Grid Gansu Electric Power Company. This dataset contains two subdatasets, namely, FTFF1 and FTFF2.

## Conflicts of Interest

All authors declare no conflict of interest over this article.

## Acknowledgments

This work was supported by the State Grid Gansu Electric Power Company Science and Technology Xing'an Project (No. 52272221001Z).

## References

- [1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks & Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.
- [2] Y. Li, R. Gault, and T. M. McGinnity, "Probabilistic, recurrent, fuzzy neural network for processing noisy time-series data,"



- IEEE transactions on neural networks and learning systems*, vol. 2, no. 1, pp. 1–10, 2021.
- [3] X. Liu, Y. Zhou, and Z. Wang, “Deep neural network-based recognition of entities in Chinese online medical inquiry texts,” *Future Generation Computer Systems*, vol. 114, pp. 581–604, 2020.
  - [4] S. N. Bhattu, S. K. Nunna, D. V. L. N. Somayajulu, and B. Pradhan, “Improving code-mixed POS tagging using code-mixed embeddings,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 19, no. 4, pp. 1–31, 2020.
  - [5] H. Yuan, “Combined networks with multi-level attention for distantly-supervised relation extraction,” *Journal of Physics: Conference Series*, vol. 1550, article 032065, 2020.
  - [6] S. Chen and M. Wu, “Attention collaborative autoencoder for explicit recommender systems,” *Electronics*, vol. 9, no. 10, p. 1716, 2020.
  - [7] R. Kiros, R. Salakhutdinov, and R. S. Zemel, *Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models*, TACL, 2015.
  - [8] J. Xiong, J. Ren, L. Chen et al., “Enhancing privacy and availability for data clustering in intelligent electrical service of IoT,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2019.
  - [9] X. Liang, L. Lin, W. Yang, P. Luo, J. Huang, and S. Yan, “Clothes co-parsing via joint image segmentation and labeling with application to clothing retrieval,” *IEEE Transactions on Multimedia*, vol. 18, no. 6, pp. 1175–1186, 2016.
  - [10] Y. Zuobin, S. I. Yuanping, M. A. Jianfeng, J. Wenjie, and X. U. Shengmin, “P2HBT: partially policy hidden E-healthcare system with black-box trace ability,” *Chinese Journal of Electronics*, vol. 30, no. 2, pp. 219–231, 2021.
  - [11] J. Li, Y. Li, N. Yuan, K. Jia, and G. Song, “DC fault analysis and detection for offshore wind farms integration via MTDC,” *Electric Power Automation Equipment*, vol. 12, pp. 119–128, 2020.
  - [12] J. Lin, Y. Wang, K. Li, and M. Tian, “Arc fault detection method based on self-organizing feature mapping network,” *Electric Power Automation Equipment*, vol. 8, pp. 210–219, 2020.
  - [13] Q. Liu, X. Fang, Y. Dong, and S. Qin, “Dynamic modeling and reconstruction based fault detection and location of train bearings,” *Chinese Association of Automation*, vol. 12, pp. 2233–2241, 2019.
  - [14] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, “Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles,” *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
  - [15] H. Chen, B. Jiang, H. Yi, and N. Lu, “Data-driven fault diagnosis for dynamic traction systems in high-speed trains,” *SCIENCE CHINA Information Sciences*, vol. 50, no. 4, pp. 496–510, 2020.
  - [16] X. Yang, W. Duan, W. Chen, and J. Wang, “Study on fault tolerant control strategy of sensor fault in STATCOM,” *Journal of power capacitor and the reactive power compensation*, vol. 5, pp. 36–41, 2018.
  - [17] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Eleventh annual conference of the international speech communication association*, pp. 1–15, Beijing, 2010.
  - [18] M. Sadrzadeh, D. Kartsaklis, and E. Balkır, “Sentence entailment in compositional distributional semantics,” *Annals of Mathematics and Artificial Intelligence*, vol. 82, no. 4, article 9570, pp. 189–218, 2018.
  - [19] T. Wang, T. Wang, P. Wang, H. Qiao, and M. Xu, “An intelligent fault diagnosis method based on attention-based bidirectional LSTM network,” *Journal of Tianjin University Science and Technology*, vol. 6, pp. 601–608, 2021.
  - [20] J. Oramas and T. Tuytelaars, “Modeling visual compatibility through hierarchical mid-level elements,” 2016, <https://arxiv.org/abs/1604.00036/>.
  - [21] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui, “Jointly modeling embedding and translation to bridge video and language,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, New York, 2016.
  - [22] Y. Tian, Z. Wang, J. Xiong, and J. Ma, “A blockchain-based secure key management scheme with trustworthiness in DWSNs,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6193–6202, 2020.
  - [23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” vol. 1512, 2015, <https://arxiv.org/abs/1512.00567/>.
  - [24] J. Xiong, X. Chen, Q. Yang, L. Chen, and Z. Yao, “A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2347–2360, 2020.
  - [25] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, and S. Belongie, “Learning visual clothing style with heterogeneous dyadic cooccurrences,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4642–4650, Chengdu, 2015.
  - [26] J. Xiong, R. Ma, L. Chen et al., “A personalized privacy protection framework for mobile crowdsensing in IIoT,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.
  - [27] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg, “Retrieving similar styles to parse clothing,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 5, pp. 1028–1040, 2015.
  - [28] W. Yang, P. Luo, and L. Lin, “Clothing co-parsing by joint image segmentation and labeling,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3182–3189, Shanghai, 2014.
  - [29] W. Fang, M. Zamani, and Z. Chen, “Secure and privacy preserving consensus for second-order systems based on Paillier encryption,” *Systems & Control Letters*, vol. 148, pp. 104869–104882, 2021.
  - [30] T. Yao, T. Mei, and C.-W. Ngo, “Learning query and image similarities with ranking canonical correlation analysis,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 28–36, Xiamen, 2015.
  - [31] L. Yu, E. Park, A. C. Berg, and T. L. Berg, “Visual Madlibs: fill in the blank description generation and question answering,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2461–2469, Guangzhou, 2015.
  - [32] J. Xiong, M. Zhao, M. Bhuiyan, L. Chen, and Y. Tian, “An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 922–933, 2021.
  - [33] J. Zhao, X. Zhang, F. di et al., “Exploring the optimum proactive defense strategy for the power systems from an attack perspective,” *Security and Communication Networks*, vol. 2021, Article ID 6699108, 14 pages, 2021.

## Research Article

# A Novel Way to Generate Adversarial Network Traffic Samples against Network Traffic Classification

Yongjin Hu , Jin Tian , and Jun Ma 

Information Engineering University, Zhengzhou 450001, China

Correspondence should be addressed to Jun Ma; [sijunhan@163.com](mailto:sijunhan@163.com)

Received 29 April 2021; Revised 9 July 2021; Accepted 12 August 2021; Published 26 August 2021

Academic Editor: James Ying

Copyright © 2021 Yongjin Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Network traffic classification technologies could be used by attackers to implement network monitoring and then launch traffic analysis attacks or website fingerprint attacks. In order to prevent such attacks, a novel way to generate adversarial samples of network traffic from the perspective of the defender is proposed. By adding perturbation to the normal network traffic, a kind of adversarial network traffic is formed, which will cause misclassification when the attackers are implementing network traffic classification with deep convolutional neural networks (CNN) as a classification model. The paper uses the concept of adversarial samples in image recognition for reference to the field of network traffic classification and chooses several different methods to generate adversarial samples of network traffic. The experiment, in which the LeNet-5 CNN is selected as a classification model used by attackers and Vgg16 CNN is selected as the model to test the transferability of the adversarial network traffic generated, shows the effect of the adversarial network traffic samples.

## 1. Introduction

As a basic technology for enhancing network controllability, network traffic classification technology helps researchers understand traffic distribution, optimize network transmission, and improve network service quality; however, it is often leveraged by attackers for monitoring network traffic against the network targets and classifying the application types (such as mail, multimedia, and websites) the network traffic belong to. Based on the classification results, network traffic interception is implemented and a possible website fingerprint attack may be followed [1]. In particular, the network traffic classification, in which area machine learning and deep learning are applied, provides attackers easier conditions that result in extremely high classification accuracy. A typical scenario for a network traffic classification method based on deep learning that is used by attackers is shown in Figure 1.

Although the application of deep learning in network traffic classification can improve the accuracy of classification and has demonstrated huge potential in areas such as image recognition and natural language processing, adversaries against the

deep learning models including the convolutional neural networks (CNN) have raised the interest of scholars on the concept of “Adversarial Sample” that was introduced to the area of computer vision by Szegedy et al. [2].

In the study of image recognition, Szegedy has found that CNN tends to give an error output with high confidence degrees when intentionally adding some undetectable and tiny perturbations to the input samples of the learning models. For deep learning models, these are called “Adversarial Samples” that are crafted by these tiny perturbations to the original dataset. From the perspective of attack, the most direct application of adversarial samples is in the area of computer vision, including face identification and automatic driving. By adding perturbation undetected by eyes to the image, failures in face identification and traffic signs [3] are triggered and damages from misclassification are then caused. In the area of information security, it can also lead to detection avoidance [4] by deceiving the malware detection models based on neural network. However, on the contrary, the adversarial samples, from the perspective of defense, are also of high value. First, it can improve the robustness of deep learning models in responding to possible

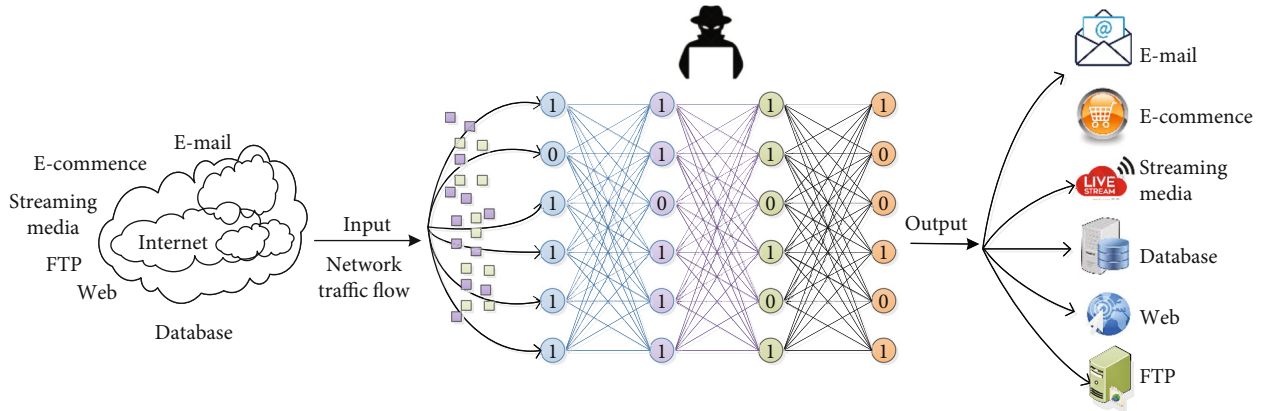


FIGURE 1: A typical scenario for network traffic classification method based on deep learning.

TABLE 1: Five types of flow in network traffic classification.

Flow granularity	Points of interest
TCP connections	Heuristics based on the observation of some TCP flags (i.e., SYN, FIN, and RST) or TCP state machines are used to identify the start and the end of each connection.
Flow	A typical flow definition uses the 5-tuple {source (IP), source (port), destination (IP), destination (port), and transport-level protocol}.
Bidirectional flows	Same as above, but includes both directions of traffic, assuming both directions of flows can be observed (especially challenging on backbones where internet routing is often asymmetric).
Services	Typically defined as all traffic generated by an IP-port pair.
Hosts	Some approaches classify a host by the predominant traffic it generates, assuming both directions of traffic (to and from the host) can be observed.

adversarial sample attack by being trained with adversarial samples generated in advance [5]. Second, the adversarial samples can be leveraged to deceive the classification models by attackers using the deep learning network, which results in misclassification and increase of attack cost, thus canceling the attacks. From the second view above, this paper is designed for defenders to trigger errors in attackers' network classification by crafting adversarial samples for network traffic with the addition of perturbation and thus forming deceptive network traffic against attackers' network traffic classification attacks.

In this paper, the concept of adversarial samples is introduced to defend the network traffic classification attacks initiated by attackers. Adversarial samples of network traffic are generated to deceive network traffic classification models based on deep learning network used by the attackers, resulting in misclassification and attack failure. The contributions of the paper are as follows: firstly, the concept of adversarial samples is introduced into network traffic as a view of active defense, and deceptive effects initiated by different adversarial samples are compared. Secondly, contrary to the fact that attackers initiate attacks with adversarial samples in other areas, the adversarial samples of a network are considered as a defensive way to confuse the attackers' classification models, that can be regarded as "attacks in active defense." Finally, the LeNet-5 CNN is selected as a network traffic classification model used by attackers to be deceived, and

Vgg16 CNN is chosen as the model to test the transferability of the adversarial network traffic generated.

## 2. Related Work

**2.1. Network Traffic Classification.** Based on the granularity of network traffic, the study in network traffic classification is mainly for the following three levels [6]: packet, flow, and stream. In the three levels mentioned above, the flow level includes five types of flow network traffic according to different granularities [7] as shown in Table 1, which are the most widely used.

In this paper, flow network traffic is used as the original data. By crafting adversarial samples of network traffic, the defenders deceive the attackers who use deep learning methods as their classification models. The classification methods based on deep learning assume that the statistical characteristics (such as flow duration distribution) of the network layers for some types of applications are unique. These methods, including Decision Tree, Naive Bayes, Support Vector Machine, Association Rules Learning, Neural Network, and Genetic Algorithm, are applied in the classification model's construction to classify, with such characteristics as broad scenarios, high classification accuracy, and ability in encrypted data traffic classification.

For studies of traffic classification based on machine learning, the main idea is to construct united statistical

TABLE 2: Notations of attack model.

Notation	Description	Remark
$TF$	Network traffic observed by attackers	
$X$	Feature set of traffic $TF$	$X = \{x_1, x_2, x_3 \dots x_m\}$
$C$	Application type set corresponding to network traffic $TF$	$C = \{c_1, c_2, c_3 \dots c_i, \dots c_n\}$
$F(x)$	Classification function of the classification model	Input value is traffic $TF$ , output value is the probability of the $i$ th application type in application set $C$

TABLE 3: Notations of defense model.

Notation	Description	Remarks
$P$	Perturbation	
$TA$	The network traffic adversarial samples that are created by defenders with addition of perturbation $P$ into traffic $TF$	
$X'$	Feature set of traffic $TA$	$X' = \{x'_1, x'_2, x'_3 \dots x'_m\}$
$F(x')$	Classification function of the classification model	Input value is traffic $TA$ , output value is the probability of type $i'$ th of $C$ traffic where $TA$ belongs to

*Input:*Normal Network Traffic  $TF$

*Output:*Adversarial Samples of Network Traffic  $TA$

*BEGIN.*

1.*Preprocess* ( $TF$ ); //Pre-process  $TF$  and Extract characteristic  $X$ ;

2.*TranspPcapToIDX* ( $TF$ ); //Transform  $TF$  from pcap format to IDX format;

3.*Normalized* (); //Delaminate each characteristic dimension and normalize into section [0,255];

4.*Reshape* ( $TF$ ); //Reshape each characteristic value of multiple types of characteristic as a grey value;

5.*Visualization* ( $TF$ ); //Form a  $28 \times 28$  matrix and visualize the network traffic;

6.*Training* ( $TF$ , *mode*); //Train CNN models

7.*Test* ( $TF$ ); //Test the accuracy of normal network traffic;

8.*CraftingPerturbation* (*method*); //use different methods of perturbation crafting to craft perturbation;

9. $TA = \text{GenerateAdvSample}()$ ; //  $TA = TF + P$ , overlay the perturbation and original traffic to craft adversarial samples of network traffic  $TA$

10.*Visualization* ( $TA$ ); //Compare  $TA$  and results from Step 5

11.*Evaluate* ( $TA$ ); //evaluate adversarial samples of traffic network  $TA$  being crafted

12.*Return*  $TA$ ; //output adversarial samples of traffic network.

*END*

ALGORITHM 1: Adversarial samples of network traffic crafting algorithm.

attributes of traffic as the fingerprint to classify. Ref. [8] applies for the first time machine learning into network traffic and assumes the fact that the bytes in flow can be regarded as pixels in images, and the deep learning method with excellent performance in image recognition can be used for network classification. Ref. [9] integrates feature extraction, feature selection, and classification into an end-to-end framework and calculates the load bytes of different behaviours by first-order CNN to construct fingerprints. Ref. [10] leverages characteristics of anonymized TOR (The Onion Router) network and applies the direction of the length sequence as the input for deep learning networks including SAE (Stacked Auto Encode), CNN, and LSTM (Long Short-Term Memory), to classify the webpage access. Ref. [11] applies for the first time the method of representation learning into the area of malicious network traffic clas-

sification, which regards the original traffic data as images, then it conducts classification with CNN that does well in image classification tasks, and finally, it achieves the purpose of classifying the malicious network traffic. These studies have proven the feasibility of deep learning in traffic classification and at the same time, provided targets for adversarial samples of network traffic classification based on deep learning.

For studies in adversarial network traffic classification based on deep learning, Ref. [12] proposes a defense method loading background network traffic and validates the Tor and JAP (Java Anon Proxy) anonymized network. Ref. [13] has validated the effects of encrypted network traffic classification adversary filled by encrypted protocol bytes. Ref. [14] applies different real traffic as noise during website access. Ref. [15] proposes that Walkie-talkie loads a website in

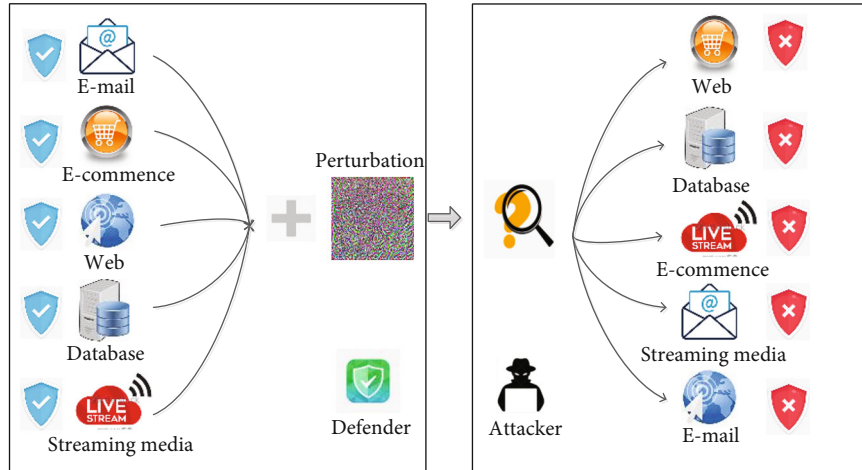


FIGURE 2: Attack and defense scenario.

TABLE 4: Experiment’s environment and parameters.

Environment	Parameters
OS	Win10,64bit
Processor	CPU: Intel Core i5-7200 U; GPU: NVIDIA GeForce 940MX
Memory	8 GB, LPDDR3, 2133 MHz
Pycharm	Community Edition 2019.3.1
Tensorflow	Ver. 2.1.0
Library support	Cleverhans 3.0.1 [32]

simplex mode to confuse the burst feature. The abovementioned studies have mainly achieved the goal of modifying the communication characteristics of the traffic and have proposed methods that mostly focus on how to avoid being detected, which are of limited ability to disguise and deceive, and with an insufficient adversary. At present, the studies close to our work are those on network traffic disguise and confusion in the area of privacy protection, in which TOR releases obfsproxy, an obfuscated proxy software [16] that makes the encrypted traffic of SSL (Secure Socket Layer) or TLS (Transport Layer Security) look like unencrypted HTTP or instance communication traffic. Ref. [17] releases TOR’s transmission layer plug-in, SkypeMorph, to fill the communication traffic between a TOR client and a network bridge to Skype video communication traffic for statistical analysis of adversarial traffic. Ref. [18] proposes the method of analysis of traffic classification rules in a black box, which can infer traffic analysis identification rules through tests and thus modify the communication packet to avoid being detected. However, all these studies neither apply the concept of adversarial samples into those on adversarial traffic analysis nor discuss it as a method of defense for defenders, which, however are the focus of this paper.

**2.2. Adversarial Samples.** The key of adversarial samples is to craft adversarial perturbation. In the area of computer vision, it is essential for perturbation to meet the requirement of being

invisible to human eyes after addition of original images and be able to confuse original classification models. In this paper, the deception for traffic classification models still have to meet certain requirements (e.g., bandwidth), though it is not necessary for the perturbation being crafted to meet the requirement of “being invisible to human eyes.”

Now, the majority of studies are focused on crafting the adversarial perturbation to misclassify an image. Szegedy et al. [2] discovered the weakness of the deep neural network in the area of image classification, proposed the concept of adversarial samples, and described the perturbation crafting as an optimized issue for the first time. Goodfellow et al. [19] proposed an optimal method of max norm constrained perturbation, which is called the Fast Gradient Sign Method (FGSM), to improve the computational efficiency and proved that high dimension linearity is the primary reason to make adversarial samples better. Kurakin et al. [20] proposed a basic iteration method that leverages FGSM in iteratively crafting perturbation. Moosavi-Dezfooli et al. [21] discovered adversarial perturbation irrelevant to particular images in image classification models, that is, the existence of universal perturbations, which can lead the classification models to misclassify any image with the addition of this perturbation. Athalye et al. [22] have discovered that the deep network classifier could also be deceived by objects in the real world printed by 3D printers. DeepFool [23] further improved the effectiveness of adversarial perturbation. Metzen et al. [24] introduced Universal Adversarial Perturbation (UAP) for semantic segmentation tasks and extended the iterative FGSM attack of [21] and changed the labels for prediction of each pixel. Mopuri et al. sought data-free universal perturbation without any sample data distribution. They proposed a new algorithm without target data to craft universal adversarial perturbation called FFF [25]. Their later work, GDUAP [26], has improved the attack effect to cause misclassification for different structures and parameter classification models and validated the validity of the method in tasks across computer visions. Furthermore, attacks in other areas are studied besides those on classification and recognition tasks in computer visions, and there is presently no research on attacks against network traffic classification.

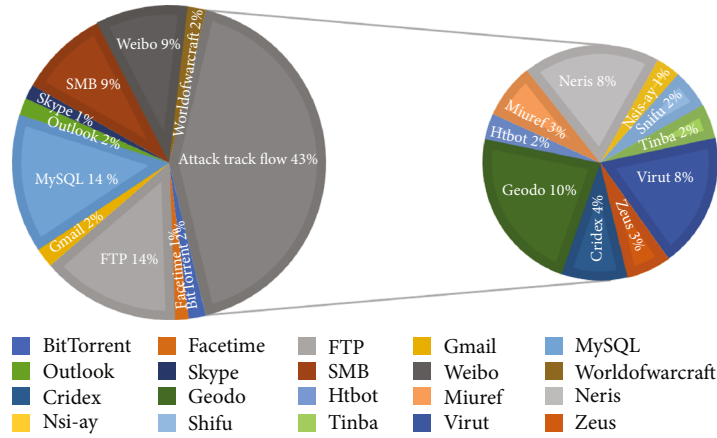


FIGURE 3: Statistical chart of USTC-TFC2016 dataset distribution.

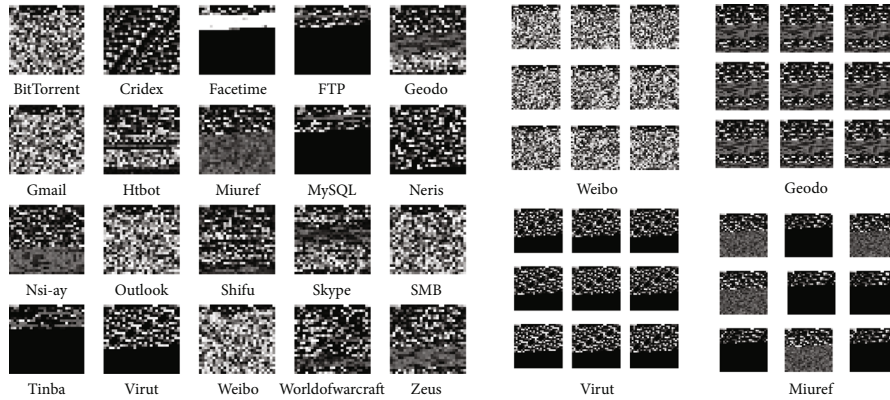


FIGURE 4: Visualization of 20 types of network traffic and consistency in the same type.

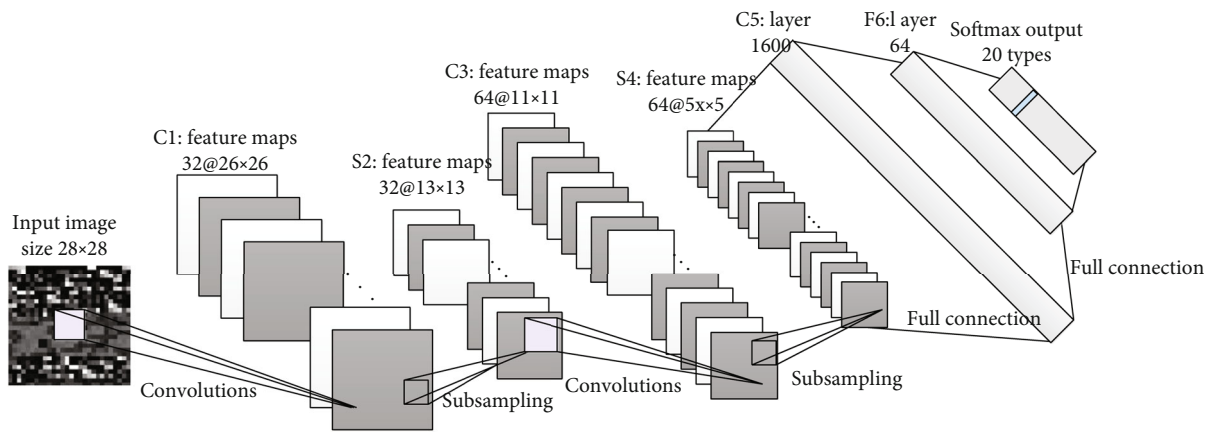


FIGURE 5: Network structure and parameters of LeNet-5 CNN.

### 3. Security Models

3.1. *Attack Model.* This paper assumes that attackers can observe the flow-level network traffic between host nodes and extract features such as packet size and internal packet arrive time. By training on classification models with these network traffic, attackers could infer the application types to thus conduct classification. From [27, 28], we build an

attack model, which is specifically described as follows: attackers attempt to classify the network traffic  $TF$  being observed into type  $i$  of application set  $C$ , in which:  $C = \{c_1, c_2, c_3 \dots c_p, \dots c_n\}$ . The feature set of network traffic  $TF$  is  $X$ ,  $X = \{x_1, x_2, x_3 \dots x_m\}$ .  $F(x)$  is the classification function of the model, and the output value is the probability of type  $i$  in  $C$ . Related notations are shown in Table 2.

TABLE 5: Network parameters of LeNet-5 CNN.

Name		Parameters
Input layer		$28 \times 28$
C1 convolution layer	Convolution core	$32 \times (3 \times 3)$
	Output	$32 \times (26 \times 26)$
S2 pooling layer	Sampling window	$2 \times 2$
	Output	$32 \times (13 \times 13)$
C3 convolution layer	Convolution core	$64 \times (3 \times 3)$
	Output	$64 \times (11 \times 11)$
S4 max pooling layer	Sampling window	$2 \times 2$
	Output	$64 \times (5 \times 5)$
Full connection layer		$1600 \times 1$
Full connection layer		$64 \times 1$
Output layer		$20 \times 1$

3.2. *Defense Model.* Defenders, according to network traffic  $TF$ , generate network traffic adversarial samples  $TA$  by adding perturbation  $P$ . This paper will generate different adversarial samples of network traffic  $TA$  by different methods of crafting perturbation, from which the feature set  $X' = \{x'_1, x'_2, x'_3 \dots x'_m\}$  is extracted, which will make the output of the attackers' classification function  $F(x')$  different from the original output  $F(x)$ . That is to say, the attackers will misclassify the traffic into the type  $i'$ th rather than type  $i$  th. Related notations are shown in Table 3.

3.3. *Methods of Generating Perturbation.* Ref. [29] summarizes the perturbation crafting into full-pixel perturbation and partial-pixel perturbation, on the basis of which there are three secondary types including target/nontarget, black box/white box, and visible/invisible. In collaboration with characteristics of network traffic classification, the methods of crafting perturbation introduced in this paper are just like those of the full-pixel perturbation in image classification; that is, adversarial samples are crafted under the context that the parameters and internal structure of the classifier used (such as LeNet-5) by attackers are known. These adversarial samples are required to lead the attackers' classifier to misclassify into not only target label but also nontarget label. Based on the abovementioned, the four perturbation crafting methods introduced in this paper are as follows:

(1) L-BFGS

L-BFGS is introduced by Szegedy [2] when he proposed the concept of adversarial samples. L-BFGS generates adversarial samples based on optimization, and is described as follows:

$$\min c \times \|x - x'\|_2 + \text{loss}_{F,t}(x'), s.t. x' \in [0, 1]^n. \quad (1)$$

(2) FGSM

TABLE 6: Network structure and parameters of Vgg-16 CNN.

Name		Parameters
Input layer		$28 \times 28$
Convolution layer	Convolution core	$32 \times (3 \times 3)$
	Output	$32 \times (26 \times 26)$
Batch normalization layer	Output	$32 \times (26 \times 26)$
Convolution layer	Convolution core	$32 \times (3 \times 3)$
	Output	$32 \times (24 \times 24)$
Batch normalization layer	Output	$32 \times (24 \times 24)$
Max pooling layer	Sampling window	$2 \times 2$
	Output	$32 \times (12 \times 12)$
Convolution layer	Convolution core	$64 \times (3 \times 3)$
	Output	$64 \times (10 \times 10)$
Batch normalization layer	Output	$64 \times (10 \times 10)$
Convolution layer	Convolution core	$64 \times (3 \times 3)$
	Output	$64 \times (8 \times 8)$
Batch normalization layer	Output	$64 \times (8 \times 8)$
Max pooling layer	Sampling window	$2 \times 2$
	Output	$64 \times (5 \times 5)$
Convolution layer	Convolution core	$128 \times (3 \times 3)$
	Output	$128 \times (3 \times 3)$
Batch normalization layer	Output	$128 \times (3 \times 3)$
Flatten layer	Output	$1152 \times 1$
Full connection layer dense 1	Output	$64 \times 1$
Full connection layer dense 2	Output	$20 \times 1$

As one of the basic methods in crafting adversarial samples, FGSM, proposed by Goodfellow et al. [19], induces a network to misclassify the image generated by adding increments into the direction of a gradient based on the principle of gradient descent. FGSM calculates perturbation by using the following:

$$P = \varepsilon \text{sign}(\nabla \mathfrak{F}(\theta, x, y)). \quad (2)$$

(3) JSMA

JSMA is a typical white box and targeted attack algorithm constrained by  $l_0$  norm proposed by Papernot et al. in 2016, which is aimed at computing a direct mapping from the input to the output to achieve an explicit adversarial goal. JSMA algorithm mainly includes three processes: calculating forward derivative of a deep neural network, calculating adversarial saliency maps, and modifying samples by adding perturbation [30].

(4) C&W Method

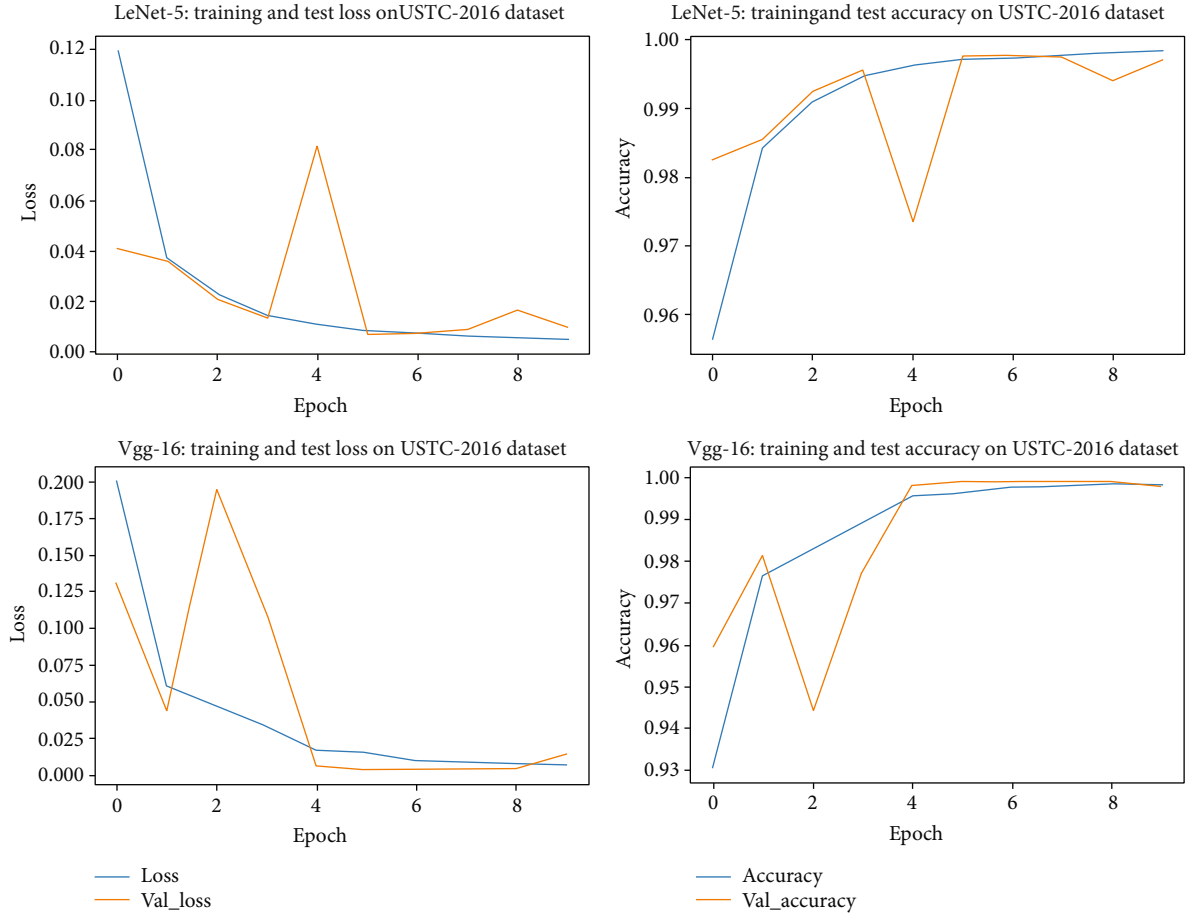


FIGURE 6: Classification train and test on the UST-TFC2016 dataset.

C&W is proposed by Carlini and Wagner [31] based on FGSM, L-BFGS, and JSMA, which improves greatly in norm  $l_0$ ,  $l_2$ ,  $l_\infty$ . The method with norm  $l_2$  as an example is shown in equation (3). C&W can produce strong adversarial samples, enhance its adversarial transferability, and achieve the ability of black box attacks.

where

$$\min \left\| \frac{1}{2} \left( (\tanh(w) + 1) - x \right) \right\|_2 + c \cdot f \left( \frac{1}{2} (\tanh(w) + 1) \right),$$

$$f(x') = \max \left( \max \left\{ Z(x')_i : i \neq t \right\} - Z(x')_i - k \right). \quad (3)$$

**3.4. Adversarial Samples of Network Traffic Crafting Algorithm.** Based on the abovementioned analysis and perturbation crafting algorithm, this paper has designed the adversarial samples of a network traffic crafting algorithm. The details are as follows:

In Algorithm 1, the real traffic needs to be preprocessed and normalized first. And then, each characteristic value of multiple types of characteristics is reshaped as a grey value in 0-255 and the network traffic is visualized. Next, the CNN model selected by attackers is constructed and trained.

The function *Training* ( $TF, mode$ ) enables it to classify the traffic data visualized and test the accuracy of classification. At the same time, different methods of crafting perturbation are used to generate perturbation, which will be overlaid with original traffic to be adversarial samples of network traffic  $TA$ . Finally, by comparing  $TA$  and  $TF$ , adversarial samples of traffic network will be evaluated.

## 4. Experiments

This paper constructs an attack and defense scenario shown in Figure 2, in which attackers are assumed to be able to observe the flow-level network traffic of different applications between host nodes and use the classification models based on deep learning for further attacks. Defenders use the adversarial samples of network traffic crafting method proposed in this paper to add a different perturbation to lead attackers to misclassify during network traffic classification and thus achieve the purpose of defense.

Environment and parameters required by the experiment are shown in Table 4.

**4.1. Dataset.** The USTC-TFC2016 dataset [11] used in this paper is as the flow traffic observed by attackers which is commonly used by network traffic classification. This dataset includes ten types of malware traffic captured from the real



TABLE 7: Classification test on the UST-TFC2016 dataset.

Application type	Accuracy		Precision		F1_score	
	LeNet-5	Vgg-16	LeNet-5	Vgg-16	LeNet-5	Vgg-16
BitTorrent	1.00	1.00	1.00	1.00	1.00	1.00
Cridex	1.00	1.00	1.00	1.00	1.00	1.00
Facetime	1.00	1.00	1.00	1.00	1.00	1.00
FTP	1.00	1.00	1.00	1.00	1.00	1.00
Geodo	1.00	1.00	1.00	1.00	1.00	1.00
Gmail	0.99	0.99	0.99	1.00	0.99	0.99
Htbot	1.00	1.00	1.00	1.00	1.00	1.00
Miuref	1.00	1.00	1.00	1.00	1.00	1.00
MySQL	1.00	1.00	1.00	1.00	1.00	1.00
Neris	0.99	1.00	0.99	1.00	0.99	1.00
Nsis-ay	0.99	1.00	1.00	0.99	0.99	1.00
Outlook	0.98	1.00	1.00	0.99	0.99	0.99
Shifu	1.00	1.00	1.00	0.99	1.00	1.00
Skype	1.00	0.98	1.00	1.00	1.00	0.99
SMB	1.00	1.00	1.00	1.00	1.00	1.00
Tinba	1.00	1.00	1.00	1.00	1.00	1.00
Virut	0.99	1.00	0.99	1.00	0.99	1.00
Weibo	1.00	1.00	1.00	1.00	1.00	1.00
WOW	1.00	1.00	1.00	0.99	1.00	1.00
Zeus	1.00	1.00	1.00	1.00	1.00	1.00

network environment by CTU researchers from 2011 to 2015 and ten types of normal traffic data simulated by professional tools. To reflect more kinds of traffic as possible, ten kinds of traffic contain eight classes of common applications. The size of the USTC-TFC2016 dataset is 3.71 GB, and the format is pcap.

**4.2. Data Preprocessing.** In this part, with the toolkit USTC-TK2016, raw traffic data (pcap format) is converted to CNN’s input data (idx format). The whole process includes traffic split, traffic clear, image generation, and IDX conversion [11]. After preprocessing, 20 types of different applications of network traffic are formed, including 10 types of 243761 normal traffic flows and 10 types of 179252 malicious traffic flows, in which 90% (379812 flows) are used as training dataset and 10% (42201 flows) as test dataset. The statistical chart of dataset distribution is showed in Figure 3.

Each of the 20 types of network traffic can be visualized to grey image with 784 (28 \* 28) bytes. The visualization results are shown in Figure 4. In Figure 4, the left group shows the visualization result of all 20 types of traffic and the right group shows the consistency in the same traffic type. It is obvious that these images visualized from network traffic have obvious discrimination degree, and each type of traffic has high consistency.

**4.3. Attacker Classification Model.** It is assumed that the attacker uses LeNet-5 CNN as his classification model, which is widely used in classification of network traffic appli-

cations [33]. Ref. [34] has improved the LeNet-5 CNN model with its network structure, and the network structure and parameters of LeNet-5 CNN are shown in Figure 5 and parameters in Table 5.

To validate the transferability of the adversarial samples of network traffic generated, the Vgg-16 CNN model is selected as the classification model to test adversarial samples crafted for LeNet-5. The parameters of the network structure of Vgg-16 are shown in Table 6.

**4.4. Classification Test.** Without the defense of adversarial samples of network traffic, the effect of classification of LeNet-5 and Vgg-16 used by the attacker is perfect. The classification test is shown in Figure 6 and Table 7 with three evaluation metrics: accuracy, precision, and F1 score.

**4.5. Perturbation Crafting.** With Algorithm 1 and four methods of perturbation crafting, adversarial samples of network traffic  $TD$  of the defender model are crafted for LeNet-5 CNN. Taking with Geodo type as the example, perturbations crafted by different methods are shown in Figure 7. In Figure 7, the column “Perturbation” shows the difference of perturbations generated by four methods, in which the brightness of the perturbation pixel corresponds to the value of the heat map. The value “1” and value “-1” of the heat map mean the strongest “positive” and “negative” perturbations after standardization. For example, perturbations generated by JASM are stronger than the perturbations generated by C&W.

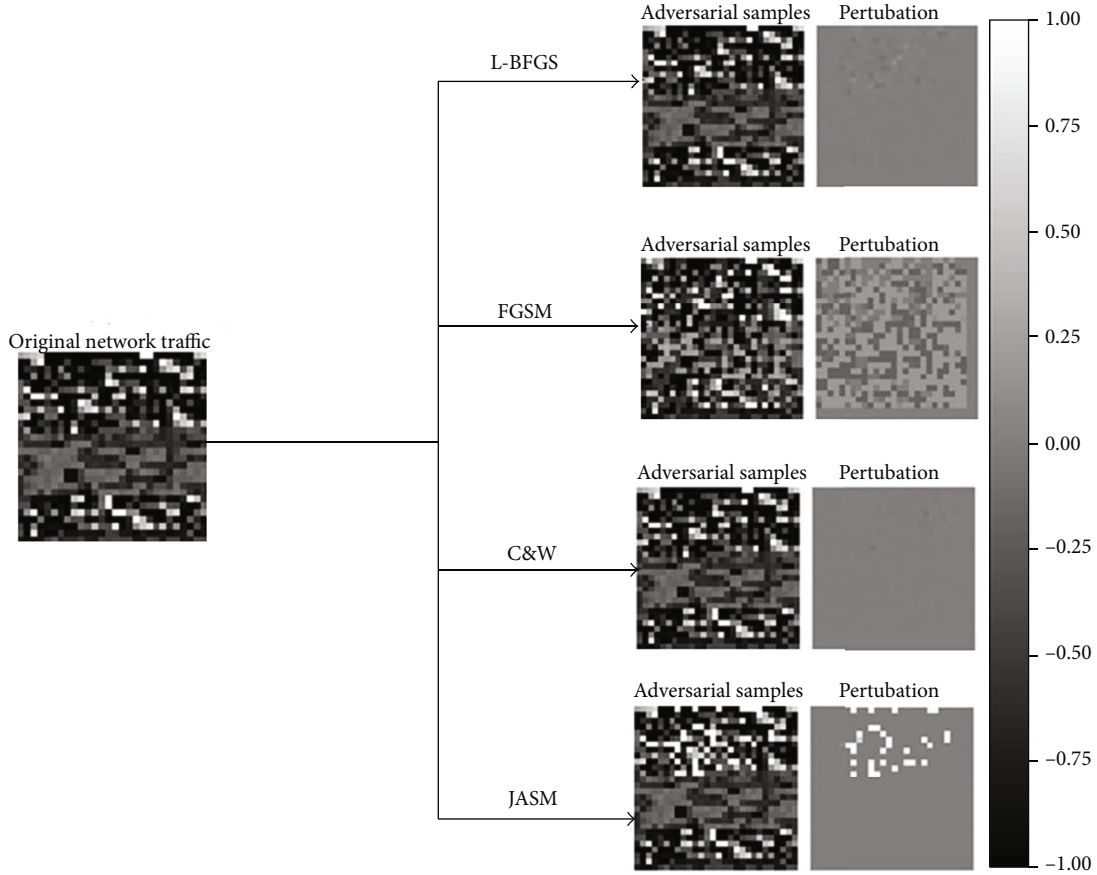


FIGURE 7: Comparison of adversarial samples of network traffic crafted by different methods.

**4.6. Comparison and Analysis.** The comparison of the experiment consists of two parts: (1) The defender carries out untargeted defense to the attacker, which means the purpose of defender is make the attacker misclassify the application class to another but no particular class. For example, the attacker misclassifies the Outlook network traffic to any other class such as Gmail and FTP. (2) The defender carries out targeted defense to the attacker, which means the purpose of the defender is to make the attacker misclassify the application class to a particular class. For example, the attacker misclassifies the Outlook network traffic to MySQL. In each part, after the test of the effect of the adversarial samples for LetNet-5 CNN, the transferability is validated, which means the defender uses adversarial samples generated for LeNet-5 CNN to deceive Vgg-16 CNN. To evaluate the effect of deceiving in untargeted defense, deceiving rate (DR) is used as shown in equation (4). And in targeted defense, matching rate (MR) [35] is defined as that which describes the percentage of the adversarial examples generated for the source model that is misclassified as the target label by the target model.

$$DR = 1 - \frac{TP_i}{TP_i + FN_i}. \quad (4)$$

To evaluate the quality of adversarial samples generated,  $l_0$  norm,  $l_2$  norm, and Structural Similarity Index (SSIM) are used. The comparison of the experiment is shown in Table 8 and Table 9:

From the comparison in Tables 8 and 9, we can validate the effect of adversarial samples of network traffic generated by four different methods. In the untargeted defense group, the adversarial samples crafted by C&W perform best on deceiving LeNet-5 CNN with low change to original but with disadvantages of slow to crafting perturbation and low transferability to other CNN models, which could be used in an application field that could provide high computation ability and demand for high deceiving rate. FGSM could craft perturbation quickly and transfer the deception to other CNNs. However, the change is to the original image of perturbation by FGSM which is much more than other methods. In the targeted defense group, C&W is also the best to perform the ability of deceiving LeNet-5 but has no effect of transferability to Vgg-16, and neither are other methods. Contrary to L-BFGS performing better in this part than in the untargeted part, FGSM performs worse in contrast to the performance in the untargeted part. About the transferability of the adversarial samples of network traffic, only JASM performs a little bit of transferability.

TABLE 8: Untargeted defense on LeNet-5 and transferability on Vgg-16.

Methods of crafting perturbation	Traffic application class	Deceiving rate on LeNet-5	L2 norm	L0 norm	SSIM	Time consuming (second)	Transferability deceiving rate on Vgg-16
L-BFGS	Geodo	6.40%	8.31%	73.28%	74.86%	13	31.25%
	Neris	53.60%	2.58%	65.20%	91.04%	13	62.31%
	Virut	46.00%	2.71%	68.96%	93.72%	13	60.87%
	Cridex	0.20%	5.00%	81.00%	86.53%	15	100.00%
	Average	26.55%	4.65%	72.11%	86.54%	14	63.61%
FGSM	Geodo	100.00%	10.83%	51.24%	38.28%	2	38.28%
	Neris	90.20%	8.79%	60.45%	54.64%	2	98.00%
	Virut	95.40%	7.74%	70.86%	77.97%	2	58.91%
	Cridex	90.80%	8.26%	76.53%	76.33%	2	44.05%
	Average	94.10%	8.91%	64.77%	61.81%	2	59.81%
C&W	Geodo	100.00%	1.00%	50.00%	99.59%	151	0.00%
	Neris	100.00%	1.00%	42.00%	99.99%	89	0.40%
	Virut	100.00%	1.00%	58.00%	99.98%	110	1.20%
	Cridex	100.00%	1.00%	75.00%	99.88%	134	0.20%
	Average	100.00%	1.00%	56.25%	99.86%	121	0.45%
JASM	Geodo	99.80%	11.38%	4.52%	63.30%	135	61.12%
	Neris	96.40%	10.12%	5.36%	65.20%	137	62.24%
	Virut	86.20%	8.91%	5.86%	71.04%	135	36.19%
	Cridex	99.80%	7.89%	5.26%	72.98%	136	31.66%
	Average	95.55%	9.56%	5.25%	68.10%	136	47.80%

TABLE 9: Targeted defense (MySQL as the targeted class) on LeNet-5 and transferability on Vgg-16.

Methods of crafting perturbation	Traffic application class	Matching rate on LeNet-5	L2 norm	L0 norm	SSIM	Time consuming (second)	Transferability deceiving rate on Vgg-16
L-BFGS	Geodo	100.00%	1.07%	66.19%	98.65%	68	0.00%
	Neris	100.00%	1.24%	72.76%	97.71%	69	0.00%
	Virut	94.60%	1.15%	75.26%	99.21%	68	0.00%
	Cridex	100.00%	1.11%	78.59%	99.66%	66	0.00%
	Average	98.65%	1.14%	73.20%	98.80%	68	0.00%
FGSM	Geodo	10.60%	10.54%	46.00%	36.26%	2	0.00%
	Neris	0.20%	8.00%	72.00%	78.37%	2	0.00%
	Virut	1.80%	8.33%	67.22%	76.67%	2	0.00%
	Cridex	2.20%	8.27%	76.27%	80.75%	2	0.00%
	Average	3.70%	8.79%	67.37%	68.01%	2	0.00%
C&W	Geodo	100.00%	1.00%	51.00%	99.71%	174	0.00%
	Neris	100.00%	1.00%	55.00%	99.86%	135	0.00%
	Virut	100.00%	1.00%	66.00%	99.89%	139	0.00%
	Cridex	100.00%	1.00%	76.00%	99.73%	167	0.00%
	Average	100.00%	1.00%	62.00%	99.80%	154	0.00%
JASM	Geodo	93.60%	7.43%	2.62%	75.24%	136	0.00%
	Neris	61.40%	11.10%	5.24%	59.74%	135	0.33%
	Virut	28.80%	10.27%	6.06%	64.65%	135	0.69%
	Cridex	69.80%	8.62%	5.65%	70.47%	135	0.00%
	Average	63.40%	9.36%	4.87%	67.53%	135	0.26%

## 5. Conclusion and Further Work

This paper first describes the current research status in the area of network traffic classification. Then, from the perspective of defenders and based on researches related, it introduces the concept of adversarial samples to network traffic and raises a novel way to generate adversarial samples of network traffic. After the models of attack and defense are described, experiments are conducted with four methods of crafting perturbation. In the experiments, LeNet-5 CNN is considered as the classification model used by the attacker to be deceived. By adding perturbation generated by different methods to grey images transformed from normal network traffic, the adversarial samples of network traffic are formed, respectively, to confuse the target model. The experiments not only compared the effect of adversarial samples generated on LeNet-5 CNN but also validated the transferability of adversarial samples of network traffic on Vgg-16 CNN.

There are three limitations and related future work about this work. First, the main goal of this paper is to show the effect of adversarial samples of network traffic, so only the basic methods of crafting perturbation are used and compared. The effect of other methods should also be considered. Secondly, it is assumed that the classification model used by the attacker in the experiment is LeNet-5. However, in the real attack and defense, other CNNs may be selected as the classification model too. So, the effect on other CNNs will be validated next. Lastly, our work in this paper only performs the transformation from the network traffic to grey images, but how to change the image to network traffic and how to keep the integrity of the original network traffic during transforming need to be studied carefully in further work.

### Data Availability

The data used to support the findings of this study are included within the article.

### Conflicts of Interest

The authors declare no conflicts of interest.

### Acknowledgments

This work was supported by the Foundation of Science and Technology on Information Assurance Laboratory (No. KJ-15-108).



### References

- [1] A. Back, U. Möller, and A. Stiglic, "Traffic analysis attacks and trade-offs in anonymity providing systems," in *Information Hiding*, I. S. Moskowitz, Ed., vol. 2137 of Lecture Notes in Computer Science, Springer, 2001.
- [2] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," 2013, <http://arxiv.org/abs/1312.6199>.
- [3] C. Sitawarin, A. N. Bhagoji, A. Mosenia, P. Mittal, and M. Chiang, "Rogue signs: deceiving traffic sign recognition with malicious ads and logos," 2018, <http://arxiv.org/1801.02780>.
- [4] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection system," in *MILCOM 2018—2018 IEEE Military Communications Conference (MILCOM)*, pp. 559–564, Los Angeles, CA, USA, 2018.
- [5] W. W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, <http://arxiv.org/1702.05983>.
- [6] G. Xiong, J. Meng, Z. Cao, Y. Wang, L. Guo, and B. X. Fang, "Research progress and prospects of network traffic classification," *Journal of Integration Technology*, vol. 1, no. 1, pp. 32–42, 2012.
- [7] A. Dainotti, A. Pescapé, and K. Claffy, "Issues and future directions in traffic classification," *IEEE Network: The Magazine of Computer Communications*, vol. 26, no. 1, pp. 35–40, 2012.
- [8] Z. Wang, "The applications of deep learning on traffic identification," <https://goo.gl/WouIM6>.
- [9] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "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, Beijing, China, 2017.
- [10] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," 2017, <http://arxiv.org/1708.06376>.
- [11] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, pp. 712–717, Da Nang, Vietnam, 2017.
- [12] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, pp. 103–114, New York: ACM Press, 2011.
- [13] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, I still see you: why efficient traffic analysis countermeasures fail," in *2012 IEEE Symposium on Security and Privacy*, pp. 332–346, San Francisco, CA, USA, 2012.
- [14] W. Cui, J. Yu, Y. Gong, and E. Chan-Tin, "Realistic cover traffic to mitigate website fingerprinting attacks," in *2018 IEEE 38th International Conference on Distributed Computing Systems*, pp. 1579–1584, Vienna, Austria, 2018.
- [15] T. Wang and I. Goldberg, "Walkie-talkie: an efficient defense against passive website fingerprinting attacks," in *Proceedings of the 26th USENIX Security Symposium*, pp. 1375–1390, Vancouver, BC, 2017.
- [16] R. Dingledine, *Obfsproxy: The Next Step in the Censorship Arms Race*, TOR Project official blog, 2012, <https://www.torproject.org/projects/ob-fsproxy>.
- [17] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "SkypeMorph: Protocol Obfuscation for TOR Bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security — CCS '12*, pp. 97–108, Raleigh, NC, USA, 2012.
- [18] F. Li, A. M. Kakhki, D. Choffnes, P. Gill, and A. Mislove, "Classifiers unclassified: an efficient approach to revealing IP traffic classification rules," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 239–245, New York, 2016.

- [19] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, <http://arxiv.org/abs/1412.6572>.
- [20] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, <http://arxiv.org/abs/1607.02533>.
- [21] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *The IEEE Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE Press*, pp. 1765–1773, Honolulu, Hawaii, USA, 2017.
- [22] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," 2017, <http://arxiv.org/abs/1707.07397>.
- [23] S. M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2574–2582, Las Vegas, Nevada, USA, 2016.
- [24] J. H. Metzen, M. C. Kumar, T. Brox, and V. Fischer, "Universal adversarial perturbations against semantic image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2755–2764, Venice, Italy, 2017.
- [25] K. R. Mopuri, U. Garg, and R. V. Bahu, "Fast feature fool: a data independent approach to universal adversarial perturbations," <http://arxiv.org/abs/1707.05572>, 2017.
- [26] K. R. Mopuri, A. Ganeshan, and R. V. Babu, "Generalizable data-free objective for crafting universal adversarial perturbations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2452–2465, 2019.
- [27] G. Verma, E. Ciftcioglu, R. Sheatsley, K. Chan, and L. Scott, "Network traffic obfuscation: an adversarial machine learning approach," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, pp. 1–6, Los Angeles, CA, USA, 2018.
- [28] J. B. Xiong, J. Ren, L. Chen et al., "Enhancing privacy and availability for data clustering in intelligent electrical service of IoT," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2019.
- [29] W. W. Pan, X. Y. Wang, M. L. Song, and C. Chen, "Survey on generating adversarial examples," *Journal of Software*, vol. 31, no. 1, pp. 67–81, 2020.
- [30] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387, Saarbruecken, Germany, 2015.
- [31] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, San Jose, CA, USA, 2017.
- [32] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel, "Cleverhans v1.0.0: an adversarial machine learning library," 2018, <http://arxiv.org/abs/1610.00768>.
- [33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [34] W. A. Yong, Z. Huiyi, F. E. Hao, Y. E. Miao, and K. E. Wenlong, "Network traffic classification method basing on CNN," *Journal on Communications*, vol. 39, no. 1, pp. 14–23, 2018.
- [35] D. Su, H. Zhang, H. Chen, J. Yi, P. Y. Chen, and Y. Gao, "Is robustness the cost of accuracy? — A comprehensive study on the robustness of 18 deep image classification models," 2018, <http://arxiv.org/abs/1808.01688>.

## Research Article

# Sentiment Classification Algorithm Based on the Cascade of BERT Model and Adaptive Sentiment Dictionary

Ruixue Duan <sup>1,2</sup> Zhuofan Huang <sup>1</sup> Yangsen Zhang <sup>2,3</sup> Xiulei Liu <sup>1,4</sup>  
and Yue Dang <sup>1</sup>

<sup>1</sup>Computer School, Beijing Information Science and Technology University, Beijing 100101, China

<sup>2</sup>Beijing Laboratory of National Economic Security Early-Warning Engineering, Beijing 100044, China

<sup>3</sup>School of Information Management, Beijing Information Science and Technology University, Beijing 100101, China

<sup>4</sup>Laboratory of Data Science and Information Studies, Beijing Information Science and Technology University, Beijing 100101, China

Correspondence should be addressed to Ruixue Duan; [duanruixue@bistu.edu.cn](mailto:duanruixue@bistu.edu.cn)

Received 29 May 2021; Accepted 22 July 2021; Published 15 August 2021

Academic Editor: Jinbo Xiong

Copyright © 2021 Ruixue Duan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The mobile social network contains a large amount of information in a form of commentary. Effective analysis of the sentiment in the comments would help improve the recommendations in the mobile network. With the development of well-performing pretrained language models, the performance of sentiment classification task based on deep learning has seen new breakthroughs in the past decade. However, deep learning models suffer from poor interpretability, making it difficult to integrate sentiment knowledge into the model. This paper proposes a sentiment classification model based on the cascade of the BERT model and the adaptive sentiment dictionary. First, the pretrained BERT model is used to fine-tune with the training corpus, and the probability of sentiment classification in different categories is obtained through the softmax layer. Next, to allow a more effective comparison between the probabilities for the two classes, a nonlinearity is introduced in a form of positive-negative probability ratio, using the rule method based on sentiment dictionary to deal with the probability ratio below the threshold. This method of cascading the pretrained model and the semantic rules of the sentiment dictionary allows to utilize the advantages of both models. Different sized Chnsenticorp data sets are used to train the proposed model. Experimental results show that the Dict-BERT model is better than the BERT-only model, especially when the training set is relatively small. The improvement is obvious with the accuracy increase of 0.8%.

## 1. Introduction

Mobile social networking has emerged and spread widely with the development of Internet applications. Its main goal is to provide an online platform for sharing interests, hobbies, comments, and other information for the general population, and it contains a large amount of information in a form of business reviews. Expressing users' feelings about products (such as hotels and movies) is becoming increasingly common through sentiment comments on various e-commerce websites, forums, WeChat, and other platforms. Extracting user preferences from big data [1] and completing the sentiment analysis do not only serve as a reference for

other users but also provide valuable directions for improvement for businesses. Sentiment analysis is one of the important research tasks in the field of natural language processing, and it helps to complete recommendations in mobile social networks and has become a current research hotspot.

Early sentiment analysis was mostly based on artificially constructed sentiment dictionary design rules for sentiment discrimination [2, 3]. This method had the benefits of inherent simplicity and strong interpretability. However, since human emotional expressions are rich and diverse, and it is immensely difficult to create a complete set of rules capable of judging all complex emotional expressions. Therefore, people gradually adopted data-driven statistical machine

learning methods. Traditional classifiers, such as naive Bayes and support vector machines (SVMs), were used for sentiment analysis [4, 5], but these methods had the disadvantage of relying on hand-crafted features.

In recent years, deep learning has been successfully applied to the fields of image and speech recognition and natural language processing with its powerful representation learning ability, and it has also greatly promoted the research progress of sentiment analysis. Models such as LSTM [6] and BERT [7] were used to construct sentiment analysis algorithms, demonstrating the potential of deep learning models to improve sentiment analysis.

However, the deep learning models suffer from poor interpretability. Integration of sentiment dictionary information with better interpretability into the BERT characterization model and further performance improvement of the BERT model's sentiment analysis requires further research.

To this end, this paper proposes a sentiment analysis algorithm Dict-BERT, which is a cascade of deep learning BERT model and a sentiment dictionary. The concept of positive-negative probability ratio is proposed in this work and used alongside a threshold for deciding whether the BERT model is confident about the prediction, or the sample needs to be cascaded to the rule algorithm of the adaptive sentiment dictionary, and yields superior performance on sentiment classification. The Dict-BERT model combines the advantages of the BERT model and the sentiment dictionary and yields superior performance on sentiment classification. The Dict-BERT algorithm based on BERT and sentiment dictionary cascade performance on sentiment analysis task is evaluated on the Chnsenticorp data set. With the training corpus size of 2000, the Dict-BERT model demonstrates improved performance on the Chnsenticorp data set than just using BERT. The performance improved by 0.8 percentage points, with the achieved correct rate and F1 value reaching 0.9517 and 0.9520, respectively.

## 2. Related Works

There are three main methods of text sentiment analysis, namely, based on sentiment dictionaries, traditional machine learning, and deep learning algorithms.

Sentiment analysis based on sentiment dictionaries [8] is the most direct method. Generally, a heuristic-discriminative sentiment analysis algorithm is designed using a manually labelled sentiment dictionary, combined with adverbs and negative words. However, due to the continuous emergence of new words on the Internet, it is difficult for sentiment dictionaries to include all words referring to emotion. Moreover, human natural language is highly flexible, and it is difficult to design a discriminative sentiment analysis algorithm to determine the sentiment category of the text. In addition, the domain adaptability of the sentiment analysis algorithm based on the sentiment dictionary is very poor, and it is necessary to design a proprietary discriminant function for different scenarios, meaning this method has significant limitations.

A sentiment analysis method based on machine learning, proposed by Pang et al. [9] in 2002, used two text features of N-gram and part-of-speech and compared the effects of three

machine learning algorithms, namely, Naive Bayes, Maximum Entropy, and SVM, on sentiment analysis tasks. Kim and Hoyy [10] proposed quoting location features and evaluating word features to achieve sentiment classification. Xie et al. [11] proposed a new type of multistrategy fusion sentiment feature extraction technology, by constructing three different sentiment analysis models based on three levels of sentiment dictionary, emoticons, and SVM, and studied the fusion effects of different methods. Z. M. Liu and L. Liu [12] used the SVM algorithm, information gain, TF-IDF, and other feature weight calculation methods to improve the performance of sentiment analysis algorithm.

Deep learning models typically are a multilayer neural network, where the representation of the language model is obtained from large-scale data. The deep learning model initially used Google's Word2vec [13] to learn the representation of words, and its features were put into machine learning models, such as SVM, to perform sentiment classification. In addition to improving word representation accuracy, in order to benefit from valuable context information, a deep learning long short-term memory (LSTM) model was used to learn long-distance dependent information and enhance the semantic representation ability. Hu et al. [14] proposed building a related word lexicon on the basis of LSTM, which further improved the accuracy of text sentiment analysis. In recent years, models such as pretrained BERT and ALBERT [15] have emerged. These models are based on a multilayer Transformer model with a multilayer attention mechanism to complete semantic coding and contributed to the important breakthroughs in multiple natural language processing tasks, including sentiment analysis.

Since deep learning has gradually become a research hotspot in the field of natural language processing, technologies related to privacy protection [16] and the approach to solving sentiment analysis problems using deep learning methods of sentiment dictionary matching have also developed rapidly. Many scholars have attempted optimization of the text sentiment analysis algorithm using a sentiment dictionary [17]. Combining it with emotion distribution learning, Zhang et al. [18] proposed an end-to-end framework based on a multitask convolutional neural network, which can learn the sentiment distribution and classification simultaneously. Zhang et al. [19] proposed a Chinese microblog sentiment analysis algorithm based on sentiment dictionary, in which the sentiment value of the microblog text is obtained by the method of weight calculation, to realize the sentiment classification.

Wu et al. [20] proposed a slang sentiment word dictionary that is easy to maintain and expand, which is constructed using network resources, which demonstrated the advantages of using slang sentiment dictionary for sentiment classification. However, the sentiment dictionary-based classification algorithm is heavily related to the content of the sentiment dictionary and the weight of the sentiment word. Using only the sentiment dictionary appears to yield noticeably poorer performance, achieving 10% lower than the sentiment classification model based on deep learning. The effect of combined two approaches, sentiment dictionary, and deep learning algorithm requires further research.

### 3. Dict-BERT Model

*3.1. Dict-BERT Model Framework.* Dict-BERT is a sentiment analysis model based on cascaded BERT algorithm and adaptive sentiment dictionary. The flowchart of the Dict-BERT framework can be seen in Figure 1. The part of the algorithm containing the BERT model is mainly composed of an embedding layer, an encoder layer, two fully connected layers, and a softmax layer. The positive and negative sentiment judgments are completed through the two fully connected layers and the softmax layer. The softmax layer returns the probability of a positive and a negative sentiment. If the probability of the positive sentiment and the negative sentiment of the sample is  $[0.9, 0.1]$ , then it can be determined that the sample belongs to the positive sentiment.

In order to better quantify the model's prediction ability, this paper proposes the concept of positive-negative probability ratio.

$$\text{Positive-negative probability ratio} = \begin{cases} \frac{P_{\text{pos}}}{P_{\text{neg}}}, & \text{if } (P_{\text{pos}} > P_{\text{neg}}), \\ \frac{P_{\text{neg}}}{P_{\text{pos}}}, & \text{if } (P_{\text{neg}} > P_{\text{pos}}), \end{cases} \quad (1)$$

where  $P_{\text{pos}}$  represents the probability of a positive sentiment,  $P_{\text{neg}}$  represents the probability of a negative sentiment, and the sum of  $P_{\text{pos}}$  and  $P_{\text{neg}}$  is always 1. According to the definition, it can be seen that the positive-negative probability ratio must be greater than 1. When  $P_{\text{pos}}$  is larger than  $P_{\text{neg}}$ , the model determines that the sample belongs to positive sentiment class, and vice versa. If the positive and negative sentiment probabilities of the two samples are  $[0.9, 0.1]$  and  $[0.55, 0.45]$ , then the probability of the positive sentiment of the two documents is higher than the probability of the negative sentiment, so the model judges that these two documents are positive sentiment. However, the positive-negative probability ratios of the two articles are significantly different. The positive-negative probability ratio of the first document is 9, and the second one is 1.22. The higher the positive-negative probability ratio is, the higher the model's confidence in sentiment classification is. If the value of the positive-negative probability ratio is relatively low, it means that the model is struggling to correctly distinguish the sentiment tendency of the sample. In such cases the sentiment dictionary with the discriminant function are used to determine the sentiment of the sample.

This paper proposes a sentiment analysis model that combines a BERT model and an adaptive sentiment dictionary. The greater the value of the positive-negative probability ratio is, the greater the difference between the probabilities of the two categories of sentiment classification is, and therefore, the higher the credibility of the sentiment classification prediction is. On the contrary, when the positive and negative probability ratio is lower than the threshold, it indicates that the pretrained model cannot distinguish the sentiment categories. If the positive-negative probability ratio is higher than the predefined threshold, the output of the BERT classi-

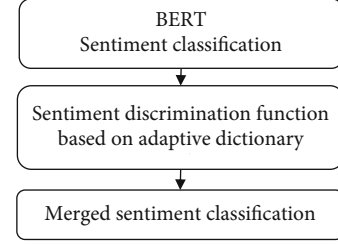


FIGURE 1: The framework of the Dict-BERT model.

fication model is directly used as the final sentiment classification. If the positive-negative probability ratio is lower than the set threshold, the discrimination function of the adaptive sentiment dictionary is used to complete the sentiment classification. In the next section, the effects of models with different thresholds will be introduced. The positive and negative probability ratio thresholds were selected as 1.2, 1.4, 1.6, 2.0, 3.0, and other values, respectively.

#### 3.2. BERT Sentiment Classification

*3.2.1. Overview of the BERT Model.* The BERT (Bidirectional Encoder Representation from Transformers) model is a pretrained model proposed by Google in 2018. The main goal of this model is to use large-scale unlabelled corpus for unsupervised training, so as to obtain a word embedding representation containing rich semantic information. The BERT model is internally composed of a two-way multilayer Transformer model. The Transformer model uses a multihead attention layer to encode contextual information. The input and output dimensions of each layer of the Transformer model remain unchanged, and the superposition of multiple layers of Transformer model can better realize the semantic encoding of the context.

*3.2.2. BERT Model for Sentiment Classification.* The flowchart of the BERT model for sentiment classification is shown in Figure 2. The embedding layer of the BERT model is composed of three parts, the word vector, the segment vector, and the position vector. Three different embedding functions are applied to transform word, segment, and position vectors. Each of the word, segment, and position vectors is in dimension 768 after the embedding processing. To represent the sentiment documents with all the word, segment, and position information, we add the values of the corresponding dimensions for all the three vectors. The word vector is obtained from the pretrained model. The segment vector is composed of 0 or 1. If the sentiment document length is shorter than max sentiment length, then the corresponding dimension is represented by 0; otherwise it is represented by 1. The position vector represents the word position in the sentiment document; it is indexed from 1 to max length. If the document is short, then 255 is filled in the position vector. The encoder layer of BERT is composed of a 12-layer Transformer model, which completes the semantic encoding of the context. Add two fully connected layers behind the encoder model to complete the conversion from the hidden layer to the hidden layer and the hidden



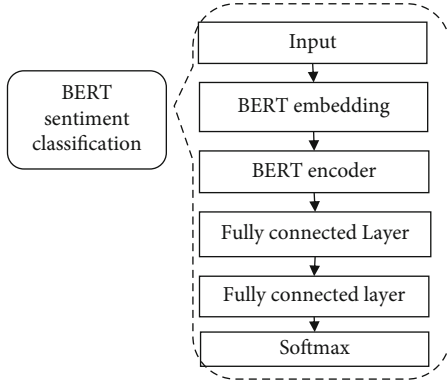


FIGURE 2: The flow chart of the BERT model for sentiment classification.

layer to the sentiment category. Finally, the softmax layer is used to calculate the probability of each sentiment category to complete the classification.

**3.3. Sentiment Discrimination Function Based on Adaptive Dictionary.** The sentiment analysis method based on sentiment dictionary generally adopts manually annotated sentiment dictionary (including positive and negative sentiment words), combined with adverbs and negative words, to design a heuristic discriminative sentiment analysis algorithm. Some sentiment dictionaries use numerical values to express the intensity of positive and negative emotions. However, the sentiment intensity of general sentiment dictionaries often cannot be consistent with the sentiment intensity of sentiment words for the test corpus. To solve this problem, this paper proposes a method of constructing a sentiment dictionary adaptively based on test corpus.

**3.3.1. Building an Adaptive Sentiment Dictionary.** In this paper, we construct a sentiment dictionary based on the HowNet sentiment dictionary and including the sentiment words frequency. The size of the sentiment dictionary is shown in Table 1. The frequency of different sentiment words in the corpus varies greatly. For example, the sentiment word “not bad” appears 211 times in the training set of the Chnsenticorp corpus, while the number of occurrences of “not occupying space” only 1 time. Obviously, the higher the frequency of sentiment words is, the stronger the emotional classification ability of sentiment words. In a corpus, the contribution of each sentiment word to the sentiment tendency is different. According to the number of occurrences of sentiment words, the sigmoid function is used to quantify the contribution of sentiment words to emotional tendency, where count represents the number of occurrences.

$$\text{Contribution} = \text{sigmoid}(\text{count}). \quad (2)$$

The calculation of contribution depends on the corpus, so the sentiment dictionary constructed in this paper is a kind of sentiment dictionary adaptive to the corpus.

TABLE 1: Number of Chinese sentiment words.

Vocabulary category	Number of word
Positive sentiment word	3638
Negative sentiment word	4401

**3.3.2. Sentiment Computing Based on Semantic Rules.** It is difficult to correctly judge the sentiment tendency of the text by relying solely on the sentiment dictionary. For example, when a negative word such as “not” or “no” accompany the sentiment word, the sentiment tendency will change. Adverbs of degree also have a great influence on the judgment of sentiment tendency. In text analysis degree, adverbs and negative words have a great influence on sentiment tendency, so this article mainly combines these two types of words and sentiment dictionary to design the discriminant function of sentiment analysis.

Next, we introduce how the judgment process of sentiment analysis is completed. First, the Peking University word segmentation tool PKUSEG is used to segment the classified text used for training and classification (in order to avoid splitting the sentiment words, you need to pass the sentiment word list to the segmentation tool) and then adjust the emotion according to the context of the emotional word, whether there are negative words or the degree adverbs. There is a contribution of words to the emotional tendency of the entire text. If the score of the positive sentiment of the text is higher than the score of the negative sentiment, it is judged that the text belongs to the positive sentiment; otherwise, it is judged to be the negative sentiment.

$$\text{Positive sentiment score} = \sum_{i=1}^N g_i \times f_i \times c_i. \quad (3)$$

$g_i$  represents the contribution of adverb words. If the adverb of degree appears in the context of positive sentiment words (the window is set to 4), then  $g_i$  is set to 2; otherwise, it is set to 1.  $f_i$  represents the contribution of negative words. If the negative words such as “no” or “not” appear in the context of positive sentiment words (the window is set to 3),  $f_i$  is set to 1; otherwise,  $f_i$  is set to -1.  $N$  indicates the number of positive sentiment words contained in the text.  $c_i$  represents the contribution of sentiment words which is defined in equation (2). Using the same method, the negative sentiment score of the text is calculated by using the context of negative sentiment words.

Using the obtained sentiment score, the emotional tendency of the text is finally determined.

## 4. Results and Discussion

**4.1. Experimental Environment and Parameter Selection.** For this study, The Pytorch was used for creating and training the classification models, using a GPU (Tesla P100) on Ubuntu16.04 system. In the experiment, the dimension of the word vector is set to 768, the maximum length of text is set to 256, and the number of Transformer layers in BERT

is set to 12. The following parameters were used to train the model: the dimension of hidden layer is 768, the size of mini-batch is set to 16, dropout is 0.1, warm up is 0.1, we use Adam optimization, epoch number is set to 5, and the learning rate is  $2 \times e - 5$ .

**4.2. Data Set.** Chnsenticorp data set [21] is a Chinese sentiment analysis data set. This corpus was compiled and published by Dr. Tan Songbo from Fudan University, which contains hotel reviews collected both from online and from review books, as well as labelled positive and negative sentiment polarities. Each sample is a short comment containing a few dozens of words. The data set consists of three parts, namely, the training set, verification set, and test set, in which the training set contains 9600 samples, the verification set contains 1200 samples, and test set contains 1200 samples. The data set details are shown in Table 2. In order to verify the effect, the sizes of the training sets are set to 2000, 4000, and 6000 samples, respectively. The results were evaluated on the test set.

#### 4.3. Evaluation Indicators

**4.3.1. Accuracy and F1.** Accuracy is used for the evaluation of the trained models, which is defined as a ratio of correctly classified samples to the total number of samples. Generally speaking, the higher the accuracy, the better. The formula is described as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (4)$$

where TN is true negative, FN is false negative, FP is true positive, and FN is false positive. In addition to accuracy, F1 value is used for the evaluation of classifier performance in this study.

**4.4. Baseline Model.** The baseline model is BERT. In the BERT model, the dimension of word vector is 768, the dimension of hidden layer is 768, the number of Transformer layers in BERT is 12, the size of minibatch is 16, the optimization function is Adam, and the drop out is 0.1. The output of the BERT model is input into two fully connected layers and a Softmax layer to complete sentiment classification. The parameters of the Dict-BERT model are the same as those of the BERT model.

**4.4.1. Performance of Sentiment Discrimination Function Based on Adaptive Dictionary.** The method of adaptive sentiment discrimination function was evaluated on the Chnsenticorp data set. The obtained results are shown in Table 3. In order to verify the effect of including adverbs and making the sentiment dictionary adaptive, the performance of the model was evaluated both with and without the adverbs and the adaptive capability of the sentiment dictionary.

From the experimental results in Table 3, it can be seen that the performance of the approach which uses only sentiment dictionary yield inferior performance with the accuracy under 80%. In the next experiment, we explain how to cascade the pretrained model with the low-accuracy method

TABLE 2: Chnsenticorp data set.

Data set	Sentiment category	Positive sentiment corpus	Negative sentiment corpus
Chnsenticorp	2	4798	4802
Chnsenticorp2000	2	996	1004
Chnsenticorp4000	2	2009	1991
Chnsenticorp6000	2	2986	3014
Chnsenticorp validation set	2	590	610
Chnsenticorp test set	2	602	598

TABLE 3: Performance analysis of sentiment discrimination module.

Model	Accuracy
Sentiment analysis model based on traditional sentiment dictionary	77.5%
Sentiment analysis model based on adaptive sentiment dictionary	81.5%
Adaptive affective analysis model-adverb of degree	81.33%

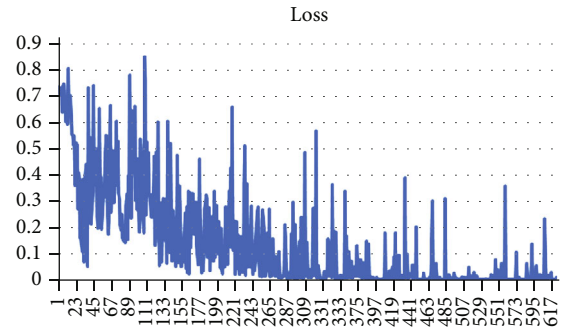


FIGURE 3: Verification of model convergence.

based on sentiment dictionary with the ultimate goal of improving the overall performance of the model.

**4.4.2. Verification of Model Convergence.** The semantic representation of each word can be obtained by pretraining the BERT model, and then, the model is fine-tuned by utilizing the training set of sentiment analysis. The cross entropy loss function is used to calculate the loss, and the parameters are updated using back propagation. Figure 3 demonstrates the convergence of the model on the Chnsenticorp data set (in 2000 samples, batch size is 16, so for each epoch, the training is done  $2000/16 = 125$  times, and epoch is 5). When the iteration times reach 625 ( $125 * 5$ ), the model converged.

**4.4.3. Comparison of Effects between the BERT Model and the Dict-BERT Model.** To verify the effectiveness of the model proposed in this paper, we conducted lots of experimental on Chnsenticorp data set.

Figure 4 and Table 4 show the accuracy of the classifier on the test set of Chnsenticorp data set with the training set size of 2000, when the positive and negative probability ratio thresholds of the BERT model are set to be 1.2, 1.4, 1.6, 1.8, 2, and 3, respectively. It can be seen from Figure 4 that when the

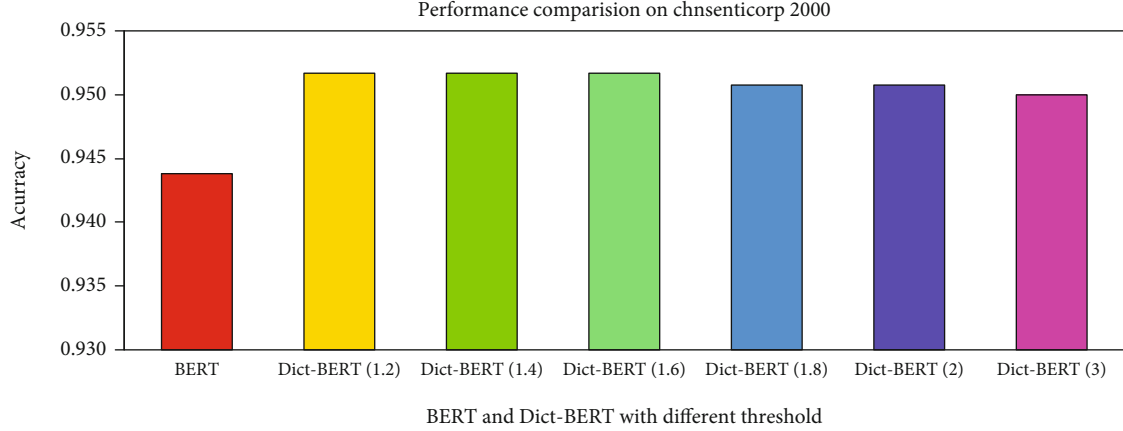


FIGURE 4: Dict-BERT and BERT models. Comparison of accuracy on Chnsenticorp2000 data set.

TABLE 4: Comparison of accuracy of each model in Chnsenticorp2000 data set and F1 score.

Model (threshold value of positive-negative probability ratio)	Accuracy	F1
BERT	0.9438	0.948
Dict-BERT(1.2)	0.9517	0.9520
Dict-BERT(1.4)	0.9517	0.9520
Dict-BERT(1.6)	0.9517	0.9520
Dict-BERT(1.8)	0.9508	0.9515
Dict-BERT(2.0)	0.9508	0.9515
Dict-BERT(3.0)	0.9500	0.9500

TABLE 5: Comparison of accuracy of each model in Chnsenticorp4000 data set and F1.

Model (threshold value of positive and negative probability ratio)	Accuracy	F1
BERT	0.9508	0.9515
Dict-BERT(1.2)	0.9558	0.956
Dict-BERT(1.4)	0.9558	0.956
Dict-BERT(1.6)	0.9542	0.954
Dict-BERT(1.8)	0.9533	0.9535
Dict-BERT(2.0)	0.9508	0.9525
Dict-BERT(3.0)	0.9500	0.956

training set size is only 2000, the accuracy of BERT model is only 0.9438. The Dict-BERT model combines the adaptive sentiment discrimination function with the BERT model. Although the accuracy rate of the model based on sentiment discrimination function is only 0.815, applying the positive and negative probability ratio method with the Dict-Bert model results in the accuracy improvement with the metric rising above 0.95. This approach makes full use of the advantages of pretraining the BERT model and the model based on emotion dictionary. When the threshold value is 1.2, 1.4, and 1.6, the accuracy rate increases to 0.9517, showing the total

accuracy increased by 0.8% and the F1 value increased by 0.4%.

Table 5 lists the accuracy and F1 value of the BERT model and Dict-BERT model with different positive and negative probability ratio thresholds with 4000 samples in the training set in Chnsenticorp data set.

Figure 5 shows the comparison between the accuracies obtained using two models. It can be seen that with the increase of training set, the accuracy of both models can be improved. The highest accuracy achieved by Dict-BERT model is 0.9558. The accuracy of Dict-BERT is better than that of the BERT model, improved by 0.5%.

In order to study the effect of training set size on the Dict-BERT model, Figure 6 compares the accuracies obtained using Dict-BERT and BERT models with varying positive and negative probability ratios with the training set size of 2000, 4000, and 9600. It can be seen that increasing the training set to 9600, the accuracy of Dict-BERT model slightly is higher than that of the BERT model, with a 0.08% difference. It is apparent that the accuracy of Dict-BERT and BERT is increasing with the increase of the training corpus. Under the condition of insufficient size of the training corpus, the Dict-BERT model has added semantic rules based on the emotion dictionary to make up for the insufficient training of the pretrained model. However, it appears that for the sufficient size of the training corpus the advantages of Dict-BERT model are reduced.

The higher the positive and negative probability ratio is, the higher the reliability of the pretrained model. When the positive-negative probability ratio is low, the credibility of the pretrained model is low, which means that the semantic information can be effectively used to improve the overall performance of the model. However, with the increase of the threshold, the amount of data used for classification using semantic rules based on sentiment dictionary increases. Since the accuracy of this method is low, the total accuracy of the model therefore decreases. It can be seen from Figure 6 that with the increase of the threshold of positive-negative probability ratio, the accuracy of the cascade model first increases, followed by a decrease. Enlarging the training set results in a steady improvement

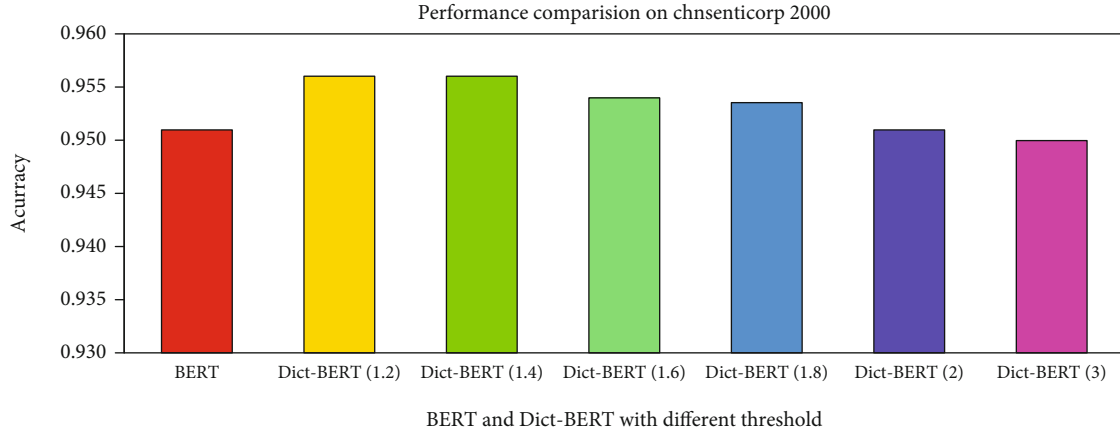


FIGURE 5: Dict-BERT and BERT model. Comparison of accuracy on Chnsenticorp4000 data set.

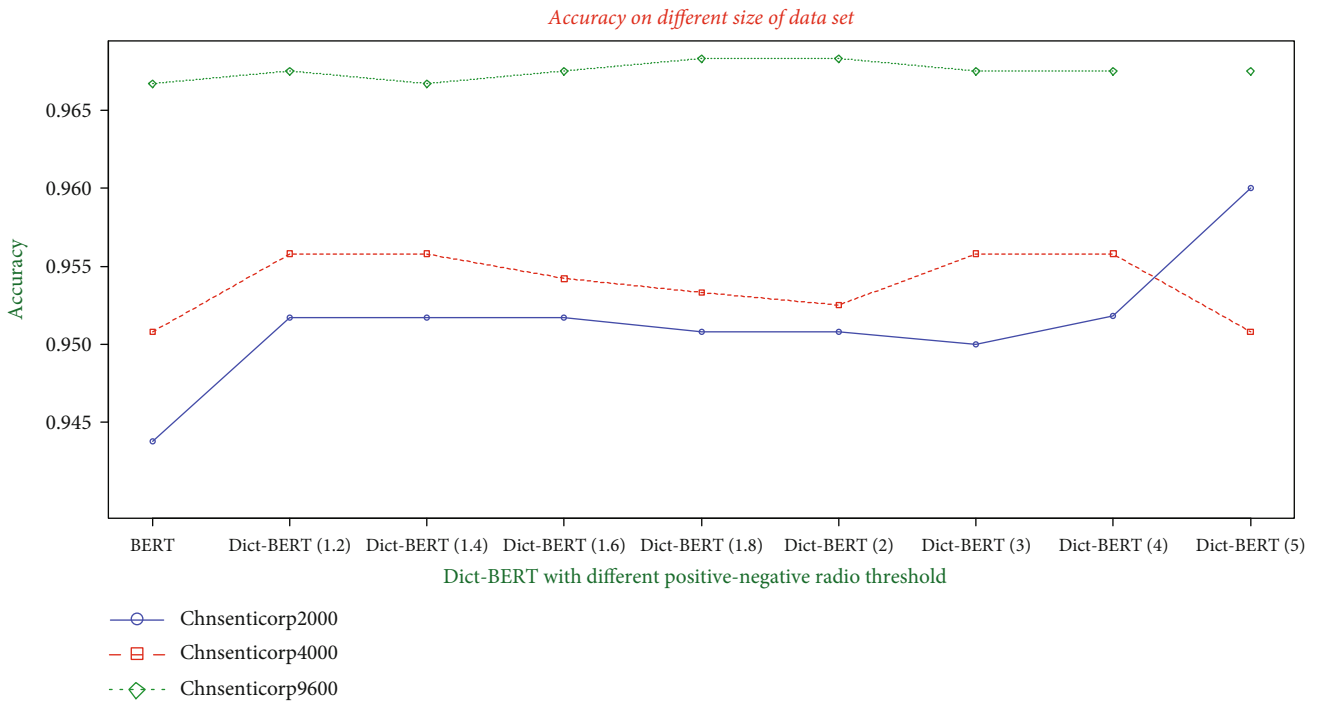


FIGURE 6: Comparison of model effects on different scales of Chnsenticorp data set.

of the accuracies of BERT and Dict-BERT models, with the accuracy of the Dict-BERT model higher than the accuracy of the BERT model. The smaller the training corpus is, the greater the improvement offered by the Dict-BERT model over the BERT model is.

### 5. Conclusion

In this paper, a sentiment analysis algorithm is proposed which is a cascade made of a pretrained deep learning BERT model and a semantic rule model. The accuracy rate of sentiment analysis based on the discriminant function of sentiment dictionary is only 81%, which is lower than the accuracy of the pretrained model. However, by cascading the pretrained model and the model based on sentiment dic-

tionary and introducing the concept of positive-negative probability ratio, the performance is further improved. The smaller the training corpus is, the more prominent the advantages of the proposed model are. If the training corpus of a task is insufficient, the cascade method proposed in this paper can be used to introduce more data knowledge to improve the performance of the system.

There are many discriminant models based on sentiment dictionary to solve the task of sentiment analysis. In the future, we would like to consider cascading these improved sentiment analysis models with improved pretrained models such as Roberta, BERT-wwm, XLNet, and ALBERT, to further improve the performance of sentiment analysis. Besides, privacy protection should also be considered in future sentiment analysis [22].

## Data Availability

The data that support the findings of this study are openly available in duanruixue/chnsenticorp at <https://github.com/duanruixue/chnsenticorp> (reference number [21]).

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This paper is supported by the funding of Beijing Natural Science Foundation (Grant No. 4204100), School Foundation of Beijing Information Science and Technology University (Grant No.1825023), Subject and Graduate Education Program of Beijing Information Science and Technology University (Grant No. 5112111004), and 2021 Promote Connotation Development of Universities-Scientific Research Training Program for College Students of BISTU (Grant No. 5102110805).

## References

- [1] J. Xiong, H. Liu, B. Jin, Q. Li, and Z. Yao, "A lightweight privacy protection scheme based on user preference in mobile crowdsensing," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 5, pp. 1–16, 2021.
- [2] J. S. Wu and K. Lu, "Chinese micro-blog sentiment analysis based on multiple sentiment dictionaries and semantic rule sets," *Computer Applications and Software*, vol. 36, no. 9, pp. 93–99, 2019.
- [3] L. Yang, F. Q. Zhou, and H. F. Lin, "Sentiment analysis based on emotion commonsense knowledge," *Journal of Chinese Information Processing*, vol. 33, no. 6, pp. 94–99, 2019.
- [4] L. Ma, X. Liu, and Y. L. Gong, "Research of micro-blog short text sentiment orientation analysis based on semantic," *Application Research of Computers*, vol. 33, no. 10, pp. 2914–2918, 2016.
- [5] H. Wei and Y. H. Wang, "Sentiment analysis of Chinese micro-blog based on topic," *Computer Engineering*, vol. 41, no. 9, pp. 238–244, 2015.
- [6] T. Hu, Y. Dan, J. Hu, X. Li, and S. Li, "News named entity recognition and sentiment classification based on attention-based bi-directional long short-term memory neural network and conditional random field," *Journal of Computer Applications*, vol. 40, no. 7, pp. 1879–1883, 2020.
- [7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," 2018, <https://arxiv.org/abs/1810.04805>.
- [8] F. Zhen, G. Yi, and Z. Zhenhao, "Sentiment analysis of movie reviews based on dictionary and weak tagging information," *Journal of Computer Applications*, vol. 38, no. 11, pp. 3084–3088, 2018.
- [9] B. Pang, L. L. Lee, and V. Shivakumar, "Thumbs up: sentiment classification using machine learning techniques," in *In the proceeding of EMNLP*, pp. 79–86, Philadelphia, USA, 2002.
- [10] S. M. Kim and E. Hooy, "Automatic identification of pro and con reasons in online reviews," in *Proceedings of the international conference on computational linguistics (COLING) and the Association for computational Linguistics (ACL): Posters*, pp. 483–490, Sydney, Australia, 2006.
- [11] L. Xie, M. Zhou, and M. Sun, "Hierarchical structure based hybrid approach to sentiment analysis of Chinese micro blog and its feature extraction," *Journal of Chinese Information Processing*, vol. 2012, no. 1, pp. 73–83, 2012.
- [12] Z. M. Liu and L. Liu, "Empirical study of sentiment classification for Chinese microblog based on machine learning," *Computer Engineering and Applications*, vol. 2012, no. 1, pp. 1–4, 2012.
- [13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *In proceedings of International Conference on Learning Representations*, Scottsdale, Arizona, USA, 2013.
- [14] F. Hu, L. Li, Z. L. Zhang, J. Y. Wang, and X. F. Xu, "Emphasizing essential words for sentiment classification based on recurrent neural networks," *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 785–795, 2017.
- [15] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: a lite BERT for self-supervised learning of language representations," 2019, <https://arxiv.org/abs/1909.11942>.
- [16] J. Xiong, R. Ma, L. Chen, Y. Tian, L. Lin, and B. Jin, "Achieving incentive, security, and scalable privacy protection in mobile crowdsensing services," *Wireless Communications and Mobile Computing*, vol. 2018, 12 pages, 2018.
- [17] H. X. Wan and Y. Peng, "Topic words extraction of social media based on semantic constrained and time associated LDA," *Journal of Chinese Computer Systems*, vol. 39, no. 4, pp. 742–747, 2018.
- [18] Y. Zhang, J. Fu, D. She, Y. Zhang, S. Wang, and J. Yang, "Text emotion distribution learning via multi-task convolutional neural network," in *International Joint Conference on Artificial Intelligence*, pp. 4595–4601, Stockholm, Sweden, 2018.
- [19] S. Zhang, Z. Wei, Y. Wang, and T. Liao, "Sentiment analysis of Chinese micro-blog text based on extended sentiment dictionary," *Future Generation Computer Systems*, vol. 81, no. 8, pp. 395–403, 2018.
- [20] L. Wu, F. Morstatter, and H. Liu, "Slang SD: building and using a sentiment dictionary of slang words for short-text sentiment classification," *Language Resources & Evaluation*, vol. 2016, no. 6, pp. 1–14, 2016.
- [21] Chnsenticorp data <https://github.com/duanruixue/Chnsenticorp>.
- [22] J. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2020.

## Research Article

# UserRBPM: User Retweet Behavior Prediction with Graph Representation Learning

Huihui Guo <sup>1</sup>, Li Yang <sup>2,3</sup> and Zeyu Liu <sup>2</sup>

<sup>1</sup>School of Cyber Engineering, Xidian University, 710126, China

<sup>2</sup>School of Computer Science and Technology, Xidian University, 710071, China

<sup>3</sup>Shaanxi Key Laboratory of Blockchain and Secure Computing, Xi'an, 710071, China

Correspondence should be addressed to Li Yang; yangli@xidian.edu.cn

Received 22 April 2021; Accepted 3 June 2021; Published 9 July 2021

Academic Editor: Lei Chen

Copyright © 2021 Huihui Guo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Social and information networks such as Facebook, Twitter, and Weibo have become the main social platforms for the public to share and exchange information, where we can easily access friends' activities and in turn be influenced by them. Consequently, the analysis and modeling of user retweet behavior prediction have an important application value, such as information dissemination, public opinion monitoring, and product recommendation. Most of the existing solutions for user retweeting behavior prediction are usually based on network topology maps of information dissemination or designing various handcrafted rules to extract user-specific and network-specific features. However, these methods are very complex or heavily dependent on the knowledge of domain experts. Inspired by the successful use of neural networks in representation learning, we design a framework, UserRBPM, to explore potential driving factors and predictable signals in user retweet behavior. We use the graph embedding technology to extract the structural attributes of the ego network, consider the drivers of social influence from the spatial and temporal levels, and use graph convolutional networks and the graph attention mechanism to learn its potential social representation and predictive signals. Experimental results show that our proposed UserRBPM framework can significantly improve prediction performance and express social influence better than traditional feature engineering-based approaches.

## 1. Introduction

Due to their convenient capability to share real-time information, social media sites (e.g., Weibo, Facebook, and Twitter) have grown rapidly in recent years. They have become the main platforms for the public to share and exchange information and to a great extent meet the social needs of users. Under normal circumstances, online social networks will record a large amount of information generated by people through interactive activities, including various user behavior data. User behaviors (also called actions) in online social networks consist of posting messages, purchasing products, retweeting information, and establishing friendships. By analyzing the distribution and causality of these behaviors, we can evaluate the influence of the initiator and the communicator of the behavior, we can predict people's behaviors on social networks, and we can deepen our under-

standing of human social behavior [1, 2]. Till now, there is little doubt that the large amount of data generated by user interaction provides an opportunity to study user behavior patterns, and the analysis and modeling of retweet behavior prediction have become a research hotspot. In addition to analyzing the retweeting behavior itself, retweeting can also help with a variety of tasks such as information spreading prediction [3, 4], popularity prediction [5, 6], and precision marketing [7, 8].

Owing to the enormous usefulness of prediction, a variety of studies have been conducted on the task of automatic prediction in social networks. Previous researches investigated the problem of user retweet behavior prediction from different points of view. On the first approach, some researchers build retweet behavior prediction models through network topology maps of information dissemination. Matsubara et al. [9] studied the dynamics of information diffusion in

social media by extending an analysis model for information dissemination from the classic ‘‘Susceptible-Infected’’ (SI) model. Wang and Wang [10] proposed an improved SIR model, which used the mean field theory to study the dynamic behavior in uniform and heterogeneous network models. Their experiment showed that the existence of the network would influence information communication. This kind of research method studied retweeting behavior by modeling the propagation path of the message from a global perspective. The other approach is the machine learning method based on feature engineering. Liu et al. [11] proposed a retweeting behavior prediction model based on fuzzy theory and neural network algorithm, which can effectively predict the user retweeting behavior and dynamically perceive the changes in hotspot topics. This research method relies on the knowledge of domain experts, and the process of feature selection may take a long time. The methods above build user behavior prediction models from different perspectives. The main purpose is to collect user behavior data in social networks, cluster and label user behavior data, and then exploit machine learning models to predict the retweeting behavior. However, in many online applications, such as personalized recommendation [12, 13] and advertising [8], or personalized services [14], it is critical to effectively analyze the social influence of each individual and further predict the retweeting behavior of users.

In this paper, we focus on user-level social influence. We aim to predict the action statuses of the target user according to the action statuses of her near neighbors and her local structural information. For example, in social networks, a person’s behavior will be affected by her neighbors. As shown in Figure 1, for the central user  $u$ , if some friends (red node) around her posted a microblog and other friends (white node) did not post it, whether the action statuses of user  $u$  will be affected by the surrounding friends and forward this tweet can be regarded as a user retweeting behavior prediction problem. The social influence hidden behind the retweeting behavior not only depends on the number of active users, but may also be related to the local network structure formed by ‘‘active’’ users. The problem mentioned above are common in practical applications, such as presidential elections [15], innovation adoption [16], and e-commerce [17]. Therefore, it has inspired many research work on user-level influence models, most of which [18–20] consider complicated handcrafted features, which require extensive knowledge of specific domains.

In recent years, graph convolution networks (GCN) [21, 22] are the best choice for graph data learning tasks. Inspired by the successful application of neural networks [23] in representation learning [24, 25], we designed an end-to-end framework UserRBPM to explore potential driving factors and predictive signals in user retweeting behaviors. We expect deep learning frameworks to have better expressive capability and prediction performance. The designed solution is to represent both influence driving factors and network structures into a latent space, and then use graph neural networks to effectively extract spatial features for learning, and further construct a user retweet behavior prediction model. To predict the status of a target user  $u$ , we first

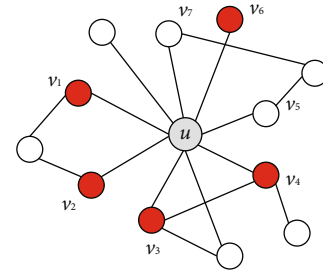


FIGURE 1: A motivating example of user retweet behavior prediction.

sample her  $k$ -order local neighbors through random walks with restart. After obtaining the  $r$ -ego network as shown in Figure 1, we leverage both graph convolution and attention mechanism to learn latent predictive signals. We demonstrate the effectiveness and efficiency of our proposed framework on Weibo social networks. We compare UserRBPM with several conventional methods, and experiment results show that the UserRBPM framework can significantly improve the prediction performance. The main contributions of this work can be summarized as follows:

- (i) We designed an end-to-end learning framework UserRBPM to explore potential driving factors and predictive signals in user retweeting behaviors
- (ii) We convert the retweeting behavior prediction into a binary graph classification, which is more operable and comprehensible
- (iii) Experiment results demonstrate that the UserRBPM framework can achieve better prediction performance than existing methods

The rest of this paper is organized as follows. Section 2 summarizes related work. Section 3 formulates the user retweet behavior prediction problem. We detail the proposed framework in Section 4. In Section 5 and Section 6, we conduct extensive experiments and analyze the results. Finally, we conclude our work in Section 7.

## 2. Related Work

**2.1. User Retweet Behavior Prediction.** Many studies on user retweet behavior in social networks are based on the analysis and modeling of the dynamics in the process of information dissemination. Currently, researches on user behavior prediction in social networks take primarily two approaches. On the first approach, Ota et al. [26] constructed the user topology network based on the user’s following relationship and discovered users who retweeted many tweets by overlapping propagation paths of retweeting. Yuan et al. [27] investigated the dynamics of friend relationships through online social interaction and proposed a model to predict repliers or retweeters according to a particular tweet posted at a certain time in online social networks. Tang et al. [28] studied the conformity phenomenon of user behavior in social networks and proposed a probabilistic model called Confluence

to predict user behavior. This model can distinguish and quantify the effects of the different types of conformity. Zhang et al. [29] proposed three metrics, namely, user enthusiasm, user engine, and user duration, to describe the user retweet behavior in the message spreading process and studied the relationship between these three metrics and the influence obtained by the user retweet behavior.

The other approach is the machine learning method based on feature engineering, which solved the problem of user behavior analysis and prediction by manually formulating rules to extract the basic features of users and network structural features. Luo et al. [30] explored features such as followers' status, retweet history, followers' interests, and followers' active time with a learning-to-rank framework to discover who would retweet a tweet poster on Twitter. Zhang et al. [18] analyzed the influence of the number of active neighbors of a user on retweeting behavior, proposed two instantiation functions based on structural diversity and pairwise influence, and applied a classifier based on logistic regression to predict users' retweet behaviors. Jiang et al. [19] pointed out that the retweeting prediction is a sing-type setting problem. By analyzing the basic influence factors of retweet behavior in Weibo, the sing-type collaborative filtering method is used to measure users' personal preferences and social influence to predict retweet behavior.

Recently, there have been efforts to detect those global patterns using deep learning. Li et al. [31] proposed an end-to-end predictor that incorporated recurrent neural network (RNN) and representation learning to infer the cascade size. This method significantly improved the performance of cascade prediction. Zhang et al. [32] proposed a novel attention-based deep neural network to obtain the user's attention interests from an attention-based neural network. Wang et al. [33] transformed the social influence prediction problem into a neural network multilabel classification problem and proposed the NNMLInf social influence prediction model. The experimental results showed that the node2vec method is more effective than the traditional manual feature extraction method in obtaining representative features of the network structure.

**2.2. Graph Representation Learning.** Graph representation learning has emerged as a powerful technique for solving real-world problems. Various downstream graph learning tasks have benefited from its recent developments, such as node classification [34], similarity search [35], and graph classification [36, 37]. Network embedding is a bridge connecting the original data of the network and network application tasks. It is aimed at representing the nodes in the network as low-dimensional, real-valued, and dense vectors. The resulting vectors can be represented and reasoned in the vector space. Therefore, the primary challenge in this field is to find a way to represent or encode the structure of graphs so that they can be easily exploited by machine learning models.

Traditional machine learning approaches relied on user-defined heuristics to extract features encoding structural information about a graph (e.g., degree statistics or kernel functions). However, recent years have seen a surge in

approaches for automatically learning to encode a graph structure into low-dimensional embedding using techniques based on deep learning and nonlinear dimension reduction. Chen et al. [38] exploited graph attention networks (GAT) to learn user node representation by spreading information in heterogeneous graphs and then leveraged limited labels of users to build end-to-end semisupervised user profiling predictor. Zhang et al. [25] introduced the problem of heterogeneous graph representation learning and proposed a heterogeneous graph neural networks model HetGNN. Extensive experiments on various graph mining tasks, i.e., link prediction, recommendation, and node classification, demonstrated that HetGNN can outperform state-of-the-art methods. Fan et al. [39] first provided a principled approach to jointly capture interaction and opinions in the user-item graph, and then presented a Graph Network model (GraphRec) to model social recommendation for rating prediction.

### 3. Problem Formulation

In this section, we introduce necessary definitions and then formulate the problem of user retweet behavior prediction.

*Definition 1 (ego network).* The ego network model is one of the important tools for studying human social behavior and social networks. Compared with the global network version, the research version of the ego network pays more attention to individual users, and it is in line with the need for personalized services in actual application systems. The research version of this paper can also be extended to other scenarios that include network relationships.

(*r-neighbors*) Let  $G = (V, E)$  denote a social network, where  $V$  is a set of users' nodes and  $E \subseteq V \times V$  is a set of relationships between users. We use  $v_i \in V$  to represent a user and  $e_{ij} \in E$  to represent a relationship between  $v_i$  and  $v_j$ . In this work, we consider undirected relationships. For a user  $u$ , its  $r$ -neighbors' nodes are defined as  $\Gamma_u^r = \{v : d(u, v) \leq r\}$ , where  $d(u, v)$  is the shortest path distance (in terms of the number of hops) between  $u$  and  $v$  in the network  $G$ , and  $r \geq 1$  is a tunable integer parameter to control the scale of the ego network.

(*r-ego network*) The  $r$ -ego network of user  $u$  is the sub-network induced by  $\Gamma_u^r$ , denoted by  $G_u^r$ .

*Definition 2 (social action).* In sociology, social action is an act that takes into account the actions and reactions of individuals. Users in social networks perform social actions, such as retweeting behaviors and citation behaviors. At each time stamp  $t$ , we observe a binary action status of user  $u$ ,  $s_u^t \in \{0, 1\}$ , where  $s_u^t = 1$  indicates that user  $u$  has performed this action before or on the timestamp  $t$ , and  $s_u^t = 0$  indicates that the user has not performed this action yet.

*Definition 3 (historical behavior).* Let  $L$  denote a stream of action logs, where each log entry  $l \in L$  is a triple  $(v, a, t)$ , representing user  $u \in V$  performed action  $a \in A$  before or on the timestamp  $t$ . Here,  $A$  is a set of action types,  $A = \{(v_i, a,$



$t \mid t \in \psi, vi \in V\}$ , and  $\psi$  denotes the time scope of historical behavior. For example, a retweeting behavior is an action in Twitter, and a citation is an action in academic social networks.

In this paper, our research motivation of the user retweeting behavior prediction problem can be vividly illustrated through the example shown in Figure 1. For a user  $u$  in her 2-ego network (i.e.,  $r = 2$ ), if some users retweet the message  $m$  before or on the timestamp  $t$ , they are considered to be active. We can observe the action statuses of  $u$ 's neighbors, such as  $s_{v_1}^t = 1$ ,  $s_{v_2}^t = 1$ , and  $s_{v_5}^t = 0$ . Moreover, the set of active neighbors of user  $u$  is represented by  $\psi_u^t = \{v_1, v_2, v_3, v_4, v_6\}$ . As shown in Figure 1, we study whether the action statuses of user  $u$  will be influenced by the surrounding friends and forward this message. Next, we will formalize the problem of user retweet behavior prediction.

*Problem 1* (user retweet behavior prediction, [18]). User retweet behavior prediction models the probability of  $u$ 's action states conditioned on her  $r$ -ego network and the action states of her  $r$ -neighbors. More formally, given  $G_u^r$  and  $S_u^t = \{s_v^t : v \in \Gamma_u^r \setminus \{u\}\}$ , it can be concluded that the user retweet behavior prediction formula of user  $u$  after a given time interval  $\Delta t$  is as follows:

$$A_v = P(s_u^{t+\Delta t} \mid G_u^r, S_u^t). \quad (1)$$

Practically,  $A_v$  denotes the predicted social action status of user  $u$ . Suppose we have  $N$  instances, and each instance is a 3-tuple  $(u, a, t)$ , where  $u$  is a user,  $a$  is a social action, and  $t$  is a timestamp. For such a 3-tuple  $(u, a, t)$ , we also know  $u$ 's  $r$ -ego network  $G_u^r$ , the action states of  $u$ 's  $r$ -neighbors  $S_u^t$ , and  $u$ 's future action states at  $t + \Delta t$ , i.e.,  $s_u^{t+\Delta t}$ . We then formulate user retweet behavior prediction as a binary graph classification problem which can be solved by minimizing the following negative log-likelihood objective w.r.t. model parameter  $\theta$ :

$$L(\theta) = - \sum_{i=1}^n \log (P_{\Theta}(s_u^{t+\Delta t} \mid G_u^r, S_u^t)). \quad (2)$$

## 4. Model Framework

In this paper, we formally propose the UserRBPM to address the problem of user retweet behavior prediction. The framework is based on graph neural networks to parameterize the probability in equation (2) and automatically detect the potential driving factors and predictive signals of user retweet behavior prediction. As shown in Figure 2, UserRBPM consists of the pretrained network embedding layer, the input layer, the GCN/GAT layer, and the output layer.

*4.1. Sampling Near Neighbors.* Given a user  $u$ , the most straightforward way to extract its  $r$ -ego network is to perform a Breadth-First-Search (BFS) starting from user  $u$ . However, for different users, the  $r$ -ego network scale (regarding the number of nodes) may vary greatly. Meanwhile, the size of

user  $u$ 's  $r$ -ego network can be very large due to the small-world property in social networks [40]. In real-world application scenarios, when sampling neighbor nodes of an ego user node, the problem that may arise is that each node has a different number of neighbors. Specifically, because of the small-world phenomenon in social networks, the size of user  $u$ 's  $r$ -ego network may be relatively very large or small. In addition, these different sizes of data are not suitable for most deep learning models.

In order to address the above problem, we select to perform random walk with restart (RWR) [41] from the original  $r$ -ego network to fix the sample size. Inspired by [42, 43] which suggested that people are more susceptible to be influenced by active neighbors than inactive ones, we start a random walk on from user  $u$  or its active neighbors. The walk iteratively travels to its neighborhood with a probability proportional to the weight of each edge. Besides, the walk returns back to the starting vertex  $u$  with a positive probability at each step. In this way, a fixed number of vertices can be collected, denoted by  $\Gamma_u^{\rightarrow r}$  with  $|\Gamma_u^{\rightarrow r}| = n$ . We then regard the subnetwork  $G_u^{\rightarrow r}$  induced by  $\Gamma_u^{\rightarrow r}$  as a proxy of the  $r$ -ego network  $G_u^r$ , and denote  $S_u^{\rightarrow t} = \{s_v^t : v \in \Gamma_u^{\rightarrow t} \setminus \{u\}\}$  to be the action statuses of  $u$ 's sampled neighbors.

When we use RWR, the starting node can be an ego user or its active neighbors. Firstly, the purpose of setting as described above is to make the starting node in the sequence obtained by walking as much as possible to keep in touch with surrounding neighbors, instead of being relatively single. Thus, it can further support that people are more susceptible to active neighbors. Accordingly, the random walk with a restart strategy can meet this requirement. Secondly, the starting nodes include ego users and active neighbors. This setting allows ego users and active neighbors to participate in the next embedding process as much as possible, which also shows that active neighbors will affect its surrounding nodes.

*4.2. Graph Neural Network Model.* We design an effective graph neural network model to incorporate both the structural properties in  $G_u^{\rightarrow r}$  and action statuses in  $S_u^{\rightarrow r}$ , learn a hidden embedding vector for each ego user, then use it to predict the action statuses  $S_u^{t+\Delta t}$  of the ego user in the next period of time. As shown in Figure 2, the graph neural network model includes the embedding layer, instance normalization layers, the input layer, the graph neural network layer, and the output layer.

*4.2.1. Embedding Layer.* Representation learning [44, 45] has been a hot topic in the academic community in recent years. In the context of graph mining, there have been many studies on graph representation learning. For graph structure data such as social networks, we want to learn the users' social representation from users' relationship network data, that is, our main purpose is to discover the network structural properties and encode them into low-dimensional latent space. More formally, network embedding learns an embedding matrix  $X \in R^{d \times |V|}$ , with each column corresponding to the representation of a vertex (user) in network  $G$ . In our scheme, we learn a low-dimensional dense real number

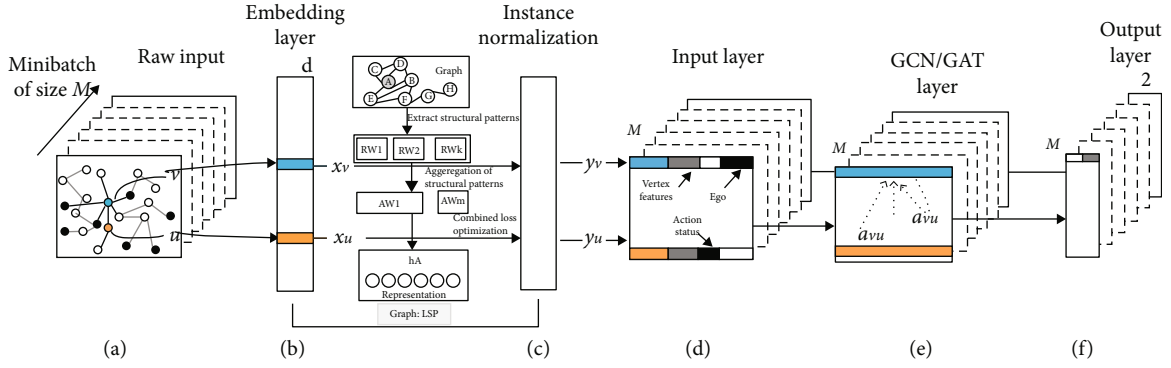


FIGURE 2: Our proposed framework of UserRBPM (User Retweet Behavior Prediction Model).

vector  $x_v \in R^d$  for each node  $v$  in the network, where  $d \ll N$ . The process of network representation learning can be unsupervised or semisupervised.

In social networks, when considering the structural information, we can take the triadic closure patterns with strong ties as an example [46]. As shown in Figure 3, there will be such a case: the figure on the left contains a triadic closure. For the green node, it is equivalent to a different tree structure on the right (there is no triadic closure) after two neighborhood aggregations, which ignore the structural information of triadic closure. Therefore, there is a need for a method of graph representation learning that can adapt to different local structures.

In our work, we utilize the GraLSP model [47] for graph representation learning, which explicitly incorporates local structural patterns into the neighborhood aggregation through random anonymous walks. Specifically, the framework captures the local structural patterns via random anonymous walks, and then these walk sequences are fed into the feature aggregation, where various mechanisms are designed to address the impact of structural features, including adaptive receptive radius, attention, and amplification. In addition, GraLSP can capture similarities between structures and are optimized jointly with near objectives of nodes. The process of the GraLSP model for graph representation learning is shown in Figure 4. In the case of making full use of the structural model, GraLSP can outperform competitors in various prediction tasks in multiple datasets.

**4.2.2. Instance Normalization Layer.** In the training process of the UserRBPM model, we applied *Instance Normalization (IN)* [48] to prevent overfitting, which is a regularization technique that loosens the model and allows for greater generalization. And for such tasks that focus on each sample, the information from each sample is very important. Therefore, we adopt such a technique in the task of retweet behavior prediction. After original data is normalized, the indicators are between  $[0, 1]$ , which is suitable for comprehensive comparative analysis. Furthermore, it helps to speed up learning and also reduces overfitting.

As illustrated in Figure 2(c), for each user  $v \in \Gamma_u \xrightarrow{r}$  after retrieving her representation  $x_v$  from the embedding layer,

the instance normalization  $y_v$  is given by

$$y_{vd} = \frac{x_{vd} - \mu_d}{\sqrt{\sigma_d^2 + \varepsilon}}, \quad (3)$$

for each embedding dimension  $d = 1, 2, \dots, D$ , where

$$\mu_d = \frac{1}{n} \sum_{v \in \Gamma_u \xrightarrow{r}} x_{vd}, \quad (4)$$

$$\sigma_d^2 = \frac{1}{n} \sum_{v \in \Gamma_u \xrightarrow{d}} (x_{vd} - \mu_d)^2. \quad (5)$$

Here,  $\mu_d$  and  $\sigma_d$  are the mean and variance, respectively, and  $\varepsilon$  is a small number for numerical stability. Intuitively, such normalization can remove instance-specific mean and variance, which encourages the downstream model to focus on relative positions of users in latent embedding space rather than their absolute positions. As we will see later in Section 6, instance normalization can indeed help avoid overfitting during training.

**4.2.3. Input Layer.** As claimed in Figure 2(d), the input layer constructs a feature vector for each user. The feature vector considered in our work consists of three parts: (1) The normalized low-dimensional embedding comes from the upstream instance normalization layer. (2) Two binary variables are also considered: the first variable represents the user's action statuses and the second variable represents whether the user is an ego user. (3) The input layer also includes other personalized vertex features, such as spatial-level features (e.g., social roles) and temporal-level features (e.g., similarity, exposure, and retweet rate.)

**4.2.4. GCN Layer.** Social networks can be regarded as graph structure data, where users represent nodes and the connection relationships between users represent edges in the graph. The characteristics of users can no longer fully represent all the information of the individual. It may be due to the loss of errors in the data collection process or the disguise of some individuals, causing some important features to be biased in the training process. Thus, we need the information of its

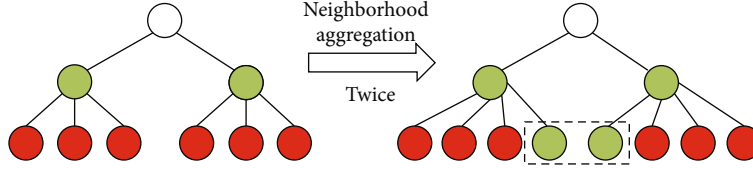


FIGURE 3: Computational tree of a triadic closure graph.

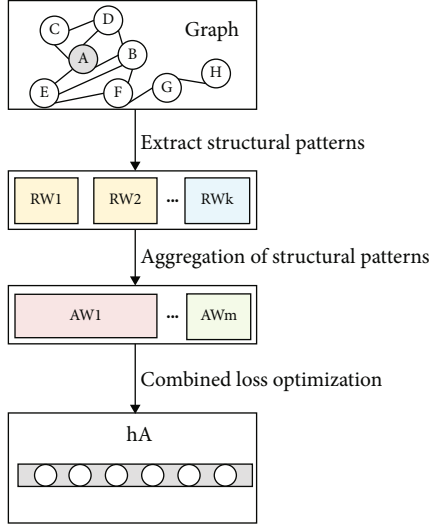


FIGURE 4: Overview of the GraLSP model.

neighbor nodes to supplement the information of the current node. In this way, we can get more complete information than a single individual characteristic. Therefore, how to combine the characteristics of neighbors with the current node is a critical part of its realization.

The recently developed GCN is a successful attempt to generalize the convolutional neural networks used in Euclidean space to graph structure data modeling. The GCN model naturally integrates the connection mode and feature attributes of graph structure data, and it is much better than many state-of-the-art methods on benchmarks. GCN is a semisupervised learning algorithm for graph structure data, which can effectively extract spatial features for machine learning on such a network topology. Simultaneously, it can perform end-to-end learning of node feature and structure information, which is one of the best choices for graph data learning tasks at present.

Suppose an undirected graph has  $n$  nodes, each node has  $d$ -dimensional features, the adjacency matrix of the graph is denoted as  $A$ , and the matrix composed of all node features is denoted as  $\mathbf{X}$ , where  $\mathbf{X} = [x_1, x_2, \dots, x_n]^T \in \mathbf{R}^{n \times d}$  and  $x_i \in \mathbf{R}^d$  is the  $d$ -dimensional feature vector of node  $i$ . If the labels of a set of nodes are given, our goal is to predict the labels of the remaining nodes. Thus, for the GCN network, the input is a node feature matrix  $\mathbf{X} \in \mathbf{R}^{n \times d}$  and the propagation between layers can be expressed as follows:

$$\mathbf{H}^{l+1} = \sigma(\mathbf{L}(G)\mathbf{H}^l\mathbf{W}^l), \quad (6)$$

where  $\mathbf{H}^l$  is the  $l$ th activation matrix,  $\mathbf{W}^l$  is the trainable weight matrix of the  $l$ th layer, and this layer functions as a feature map.  $\sigma$  is a nonlinear activation function,  $\mathbf{L}(G)$  is a  $n \times n$  matrix that can capture the structure information of graph  $G$ , and GCN uses symmetric normalization to perform aggregation operations. The conventional graph convolution operation is as follows:

$$\mathbf{L}(G) = \mathbf{D}^{-1/2} \tilde{\mathbf{A}} \mathbf{D}^{-1/2}, \quad (7)$$

where  $\mathbf{A} = [a_{ij}] \in \mathbf{R}^{n \times n}$  is a nonnegative adjacency matrix,  $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ ,  $\mathbf{I}$  is the identity matrix,  $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_n)$  is the degree matrix of  $\mathbf{A}$ ,  $d_i = \sum_j a_{ij}$  is the degree of node  $i$ .

We can divide the above learning process into three parts. The first part is transformation: transform and learn the current node features. The second part is aggregation: aggregate the features of neighboring nodes to get the new features. The third part is activation: use an activation function to increase nonlinearity.

**4.2.5. GAT Layer.** Essentially, both GCN and GAT are aggregation operations that aggregate the characteristics of neighbor nodes into the central node. GCN uses the Laplacian matrix to perform graph convolution operations, while GAT introduces the attention mechanism into GCN, which can add weight to the influence of neighboring nodes, thereby differentiating the influence of neighboring nodes. GAT assigns different weights to each node, paying attention to those nodes with greater effects, while ignoring some nodes with smaller effects. To a certain extent, the performance ability of GAT will be stronger, because the correlation between node features will be better integrated into the model.

**4.2.6. Output Layer.** In the output layer, each node corresponds to a two-dimensional representation, which is used to represent the behavior state (retweet/unretweet, cite/uncite, etc.). The calculation process is shown in equation (8). By comparing the representation of the ego user with ground truth, we then optimize the log-likelihood loss:

$$\hat{Y} = \log_{\text{softmax}}(\mathbf{H}'\Theta + \mathbf{b}), \quad (8)$$

where  $\Theta$  is a weight matrix,  $\mathbf{b}$  represents the bias term.

## 5. Experiment Setup

In this section, we first introduce the construction process of the dataset and the experimental hypothesis. Then, we

present the existing representative methods and evaluation metrics. Finally, we introduce the implementation details of the UserRBPM framework.

### 5.1. Dataset Presentation and Processing

*5.1.1. The Presentation of Raw Datasets.* We use real-world datasets to quantitatively and qualitatively evaluate the proposed UserRBPM framework. We used the Weibo dataset in the work of Zhang et al. and Qiu et al. [18, 49], and Ulyanov et al. [48] also used the Weibo dataset in their work, and then we performed data preprocessing according to our research question. The microblogging network used in our research work is used to crawl data from Sina Weibo, similar to Twitter social media, which is a broadcast-style social network platform that shares brief real-time information through the follow mechanism. The follow mechanism of Weibo is divided into one-way following and mutual following. Particularly, when user  $u_1$  follows user  $u_2$ ,  $u_2$ 's activities (such as tweet and retweet) will be visible to  $u_1$ . User  $u_1$  can choose to tweet or retweet user  $u_2$ . User  $u_1$  is called the follower of user  $u_2$ , and user  $u_2$  is called the followee of user  $u_1$ . The dataset was crawled in the following ways. To start with, 100 users are randomly selected as seed users, and then user information of their followers and followees was collected. A total of about 1.8 million users and 300 million social relationships were obtained during the crawling process. A total of 1 billion microblogs were generated in this process, and all user profiles were also crawled which contain the name, gender, verification status, #bifollowing, #followers, #followees, and #microblogs. Besides that, to protect the privacy of users, we have desensitized the user id. The statistical information of raw data is shown in Table 1.

*5.1.2. The Assumption of Experiments.* The main problem studied in our paper is as follows: when a certain microblog is visible to a certain user, we predict whether the user will retweet the microblog within a certain period, which is a supervised binary classification problem. The experiments in our work are based on the following assumption:

- (i) The assumption of visibility: as long as a user sends (an original post or retweeted) a microblog, all his followers will see the microblog
- (ii) We aim to predict the action status of users at a specific time
- (iii) The assumption of timeliness: if the retweeting time of a user is more than 72 hours from the time that a microblog was first sent out, the sample will be discarded
- (iv) For a microblog, only when a user has active neighbors, that is, the user has followed the original creator or his followers have retweeted the microblog, we only consider whether the user will retweet this microblog
- (v) For a microblog, if a user has appeared as a positive sample at time  $t$ , we will not predict the action status after time  $t$

Specifically, the assumption of Weibo Visibility is set because, with more than 100 million daily active users of Sina Weibo, each user receives a large amount of information every day. If a user follows a large number of users, some microblogs will likely be overwhelmed by other messages before the user can decide whether to retweet them.

We give a restriction on whether a microblog will be retweeted at a certain time. Because for a user and a microblog, the user may not retweet the microblog when he saw it for the first time. At this time, we generate a negative sample. When he retweeted the microblog the second time he saw it, we create a positive sample. In other words, users have different action statuses in different periods. For users who did not retweet, they were regarded as negative samples since the first time they saw a microblog. If the status of each time is retained and predicted, there will be many negative samples generated. Therefore, we give the second assumption. Besides, as Zhang et al. [18] pointed out that for most microblogs, 72 hours after their first posting, the number of retweets drops dramatically, and there will be almost no user retweeting. So, we set up the third assumption.

Sina Weibo does not restrict users from retweeting messages sent by users who have not followed them in real application scenarios. There are often phenomena that users actively search for microblogs and retweet them or retweet popular microblogs. This situation is beyond the scope of our work, so we set up the fourth assumption. The last assumption is that once a user has a retweeting behavior, it will be regarded as a positive sample at every subsequent time point. Repeated predictions in the experiment are of little significance.

*5.1.3. The Generation of Samples.* In retweeting behavior prediction, since we can directly learn from the microblogs' record which users have retweeted the microblogs, the extraction process of positive samples is relatively simple. Thus, for a user  $v$  who is affected by others, he performs a social action at a certain timestamp  $t$ , then we generate a positive sample. Compared with the extraction of positive samples, it is impossible to directly know from the microblogs' records which users saw the message but did not retweet the microblogs. Therefore, the extraction method of negative samples is much more complicated. Our work is based on the assumption of visibility. We suppose that after a user posts a microblog, all of her followers can see the microblog. If someone saw the microblog but did not retweet it, we create a negative sample.

Through the analysis of the experimental data, we found that about 1.25% are original microblogs, most of which were retweeted microblogs, and 85.39% of the microblogs have been retweeted at least ten times, verifying that the retweeting behavior of users is pervasive. However, for our research scenarios, in the process of solving research problems, there are two data imbalance problems in our dataset. The first one comes from the number of active neighbors. As Zhang et al.

TABLE 1: Statistics of raw data.

Dataset	#users	#followrelationship	#original microblogs	#retweet	#ego users
Weibo	1,776,950	308,489,736	300,000	23,755,810	779,164

[18] observed, structural features are significantly related to user retweeting behavior when the ego user has a relatively large number of active neighbors. However, in most social influence datasets, although retweeting behavior is ubiquitous, the ratio between the number of active neighbors and the number of inactive neighbors is not balanced. For example, in the Weibo dataset, 80% of users have only one active neighbor and users with more than 3 active neighbors account for only 8.57%. Therefore, when we train our model on such an imbalanced dataset, the model will be controlled by observation samples with few active neighbors. To address the issues caused by data imbalance and to illustrate the superiority of our proposed model in capturing local structural information, we established a balanced subdataset *Edata* (as shown in Table 2) for fair data analysis and a further training-test scheme. Specifically, we filter out samples in which the followers or followees did not have Weibo content. Besides, we only considered samples in which ego users have at least 3 active neighbors.

Imbalanced labels are the second problem. For instance, in our Weibo data set, the ratio between positive and negative instances is about 1 : 300. Normally, the model is trained to optimize the overall accuracy, and the weights of different types of misclassification are the same when calculating the overall error. As a result, the trained model tends to judge samples belonging to the minority classification as the samples of the majority classification. Moreover, the generalization ability of the model is poor and the minority classification cannot be accurately judged. Our goal is to find out those users who will retweet, rather than pay attention to those who do not retweet. Therefore, the datasets need to be balanced.

To address the above problem, the most direct way is to select a relatively balanced dataset, that is, set the ratio of positive samples and negative samples to 1 : 3. In addition, we also used the global random downsampling method and microblog granularity-based downsampling method to process imbalanced datasets. Among them, when we use the global random downsampling method, the number of microblogs involved in the negative samples in the obtained dataset is small, and there is a case where only positive samples of the same microblog are not sampled to their corresponding negative samples. The downsampling method based on microblog granularity can try its best to ensure that the number of positive and negative samples of the same microblog is also the same. The data statistics of the balanced sample set are shown in Table 3. To visually verify from the results that the downsampling strategy adopted in our work is more suitable for our research scenarios, we conducted a comparative analysis in Section 6.

*5.1.4. The Features of Our Design.* We made detailed data observations and analyzed how the characteristics of users

at the spatial and temporal levels influence users' retweeting behavior in addition to the structural attributes of social networks. To visualize the observation results, we design several examples of statistical information, which, respectively, represent spatial-level features and temporal-level features. These characteristics can be regarded as user node features. In our work, the spatial-level features are specifically analyzed in terms of social roles. We studied the influence of social roles played by different users on the prediction performance of retweeting behavior. Inspired by the previous research work of Wu et al. [50] and Yang et al. [51], we divide users into three groups according to their network attributes: opinion leaders (OpnLdr), structural hole spanners (StrHole), and ordinary users (OrdUsr). Specifically, we consider that 5% of users with the highest PageRank score are opinion leaders, 5% of users with the lowest Burts Constraint score are structural hole spanners, and the rest are ordinary users. A detailed analysis of users' social roles and social behaviors is shown in Table 4. For the temporal-level feature, we mainly analyzed the content of the messages posted by users, and we defined the following features:

- (i) Similarity: the TF-IDF similarity between ego user's and followees' post content within a month
- (ii) Exposure: the number of microblogs posted by followees within a month
- (iii) Retweet rate: the retweet rate of ego users to their followees

*5.2. Comparison Methods.* In order to verify the effectiveness of our proposed framework, we compared the prediction performance of UserRBPM in this paper with existing representative methods. Firstly, we compared UserRBPM with previous retweeting behavior prediction methods which usually extract rule-based features. Secondly, by comparing the GraLSP method with other network embedding methods, it is verified that the local structure information plays a more important role in the prediction of retweeting behavior than the global information. The comparison method is as follows:

- (i) Handcrafted features + Logistic Regression (LR): we use the logistic method to train the classification model. The features we constructed manually include two categories: one is the user node features designed in our work, including spatial-level and temporal-level features; the other is the ego network features designed by Qiu et al. [49]. The features we used are listed in Table 5
- (ii) Handcrafted features + Support Vector Machine (SVM): we also use SVM as the classification model. The model uses the same features as the LR method

TABLE 2: Statistics of subdataset *Edata*.

Subdataset	#users	#followrelationship	#original microblogs	#retweet	#ego users
Weibo	1,500,290	20,297,550	274,150	15,755,810	151,300

TABLE 3: Data statistics of the balanced sample set.

Methods	The size of negative samples	The number of tweets involved
Before subsampling	32,855,360	189,163
Subsampling of microblog granularity	899,756	122,568
Global random subsampling	899,756	109,330

- (iii) DeepWalk: DeepWalk [52] is a network embedding method that learns a social representation of a network by truncated random walks to obtain the structural information of each vertex. It can get better results even though there are few vertices in the network
- (iv) Node2vec: Node2vec [53] further extends the DeepWalk method by changing the way that the random walk sequence is generated. This is a network embedding method that designs a biased random walk that can tradeoff between homophily and structural equivalence of the network
- (v) Our proposed method: in our proposed UserRBPM framework, we use GraLSP to extract the structural attributes of the  $r$ -ego network, design the user node features at the spatial level and temporal level, and finally apply GCN and GAT to learn latent predictive signals

### 5.3. Evaluation Metrics

**5.3.1. Performance Metrics.** In order to quantitatively evaluate our proposed framework, we use the following metrics to evaluate the performance of retweeting behavior prediction. Specifically, we evaluate the performance of the UserRBPM in terms of Area Under Curve (AUC) [35], Precision, Recall, and F1-score.

- (i) Precision: it is for the predicted result. It measures the probability that a predicted positive instance would be the true positive
- (ii) Recall: it is for the original sample. It measures the probability that the true positive would be predicted to be the positive instance
- (iii) Area Under ROC Curve (AUC): it measures the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

- (iv) F1-score: it is a comprehensive evaluation metric that integrates precision and recall

**5.3.2. Parameter Sensitivity.** In addition, parameter sensitivity is also considered in our work. We analyzed several hyperparameters in the model and tested how different hyperparameter choices affect the prediction performance.

**5.4. Implementation Details.** There are two stages for training our UserRBPM framework. In the first stage, we pretrain each module of UserRBPM, and in the second stage, we integrate the three modules of UserRBPM for fine-tuning.

**5.4.1. Stage I: Pretraining of Each Module.** For our framework, UserRBPM, we first perform a random walk with a restart probability of 0.8 and set the size of the sampled sub-network to be 30. For the embedding layer, the embedding dimension of the GraLSP model is set to three dimensions of 32, 64, and 128, and we train GraLSP for 1000 epochs. Then, we choose to use a three-layer GCN or GAT network structure; the first and second GCN/GAT layers both contain 128 hidden units, while the third layer (output layer) contains 2 hidden units for binary prediction. In particular, for UserRBPM with multihead graph attention, both the first and second layers consist of  $K = 8$  attention heads, and each attention head computes 16 hidden units (total  $8 \times 16 = 128$  hidden units). The network is optimized by the Adam optimizer with a learning rate of 0.1, a weight decay of  $5e-4$ , and a dropout rate of 0.2. To evaluate the model performance and prevent information leakage, we performed fivefold cross-validation on our datasets. Specifically, we select 75% instances for training, 12.5% instances for validation, and 12.5% instances for testing. In addition, the minibatch size is set to be 1024 in our experiments.

**5.4.2. Stage II: Global Fine-Tuning.** In the global fine-tuning stage, if the dimension of an embedding layer is set too large, then the training process will be too slow, while a small setting will affect the performance of our model. After fine-tuning the model, we found that the model performance is relatively stable when the embedding dimension is set to 64. Then, we fix the parameters of the pretrained embedding module, and train the GCN/GAT layer with the Adam optimizer for 1000 epochs, with a learning rate of 0.001. A larger learning rate can make the model learn faster, thereby accelerating the convergence speed, but the performance of the model will be affected to some extent. Therefore, we set a relatively large learning rate at the beginning, and then we gradually decrease it during training. Finally, we choose the best model by stopping using the loss on the validation sets as early as possible. The training process of the UserRBPM model is shown in Algorithm 1.

TABLE 4: The statistics of social roles and relation statuses.

Social role	OrdUsr	OpnLdr	StrHole	Sum
Retweet behavior	6,617,440 (42%)	3,623,836 (23%)	5,514,534 (35%)	15,755,810
Original post	68,537 (25%)	123,367 (45%)	82,245 (30%)	274,150
Sum	1,125,217 (75%)	150,029 (10%)	225,043 (15%)	1,500,290

TABLE 5: List of features used in this work.

Name	Description
Spatial-level features	Social role_Opinion leader (OpnLdr)
	Social role_Structure hole (StrHole)
	Social role_Ordinary users (OrdUsr)
Temporal-level features	The TF-IDF similarity between an ego user and its followees' post content within a month (similarity)
	The number of microblogs posted by the followees within a month (exposure)
	The retweet rate of ego users to their followees (retweet rate)
Handcrafted ego network features [49]	The number/ratio of active neighbors
	Density of subnetwork induced by active neighbors
	Connected components formed by active neighbors

```

Input: Datasets(train_loader, valid_loader, test_loader),
       Learning rate, Weight decay, Epochs, Batch size.
Output: The predictive value of the test samples and the prediction performance of the model.
1: Training process:
2: Define the model: Three layers of GCN (two hidden layers and one output layer)
3: Load the datasets: train_loader, valid_loader
4: # model.train()
5: for epoch = 1 to Epochs do:
6:   All data is propagated forward, and the activation function uses Leaky ReLU
7:   Use cross-entropy loss function to calculate the loss value
8:   Clear the gradient: optimizer.zero_grad()
9:   Backpropagate and calculate the gradient of the parameter
10:  Use Adam optimizer to update the gradient: optimizer.step()
11:  Calculate the accuracy of the current model on the training set
12:  Calculate the accuracy of the current model on validation set
13: end for
14: Test process:
15: # model.test()
16: Load the trained model
17: Load the dataset: test_loader
18: Calculate the predicted value  $y_{pred}$  of the test samples
19: Evaluate the model performance: Precision, Recall, F1-score, AUC

```

ALGORITHM 1: The training process of UserRBPM model.

## 6. Experiment Results

In this section, we give the quantitative and qualitative results of retweeting behavior prediction, then analyze the structural attributes and the interaction between spatial-level features and temporal-level features. Finally, the robustness of the UserRBPM framework is verified.

### 6.1. Prediction Performance Analysis

**6.1.1. Overall Performance Analysis.** To verify the influence of the structural attributes of users' ego network and the characteristics of user nodes (extracted from the spatial and temporal levels) on the prediction performance, as well as the interaction between features at different levels, we made the

TABLE 6: Prediction performance of different methods for retweeting behavior (%).

Methods	Precision	Recall	F1-score	AUC
Spatial- & Temporal-level & Handcrafted features + LR (ST&HC+LR)	69.74	71.58	70.65	77.27
Spatial- & Temporal-level & Handcrafted features + SVM (ST&HC+SVM)	68.38	69.15	68.76	78.01
DeepWalk + ST + GAT	78.21	78.46	78.28	82.81
DeepWalk + ST&HC + GAT	79.68	80.24	79.96	82.75
Node2vec + ST + GAT	78.54	81.50	79.99	82.53
Node2vec + ST&HC + GAT	79.88	81.25	80.55	82.96
Our method (UserRBPM)	81.97	82.58	82.27	83.21

following comparison. The prediction performance of different models is shown in Table 6.

Based on the analysis of four evaluation metrics used in our work, the performance of UserRBPM is better than the abovementioned benchmark methods, which demonstrate the effectiveness of our proposed framework. From the comparison among DeepWalk + ST&HC + GAT, Node2vec + ST&HC + GAT, and UserRBPM, we can observe that the GraLSP model we leverage in the embedding layer can indeed capture local structural patterns and significantly outperforms the first two methods in the experiment, confirming that the GraLSP can indeed capture local structural patterns in retweeting behavior prediction. Experiment results show that UserRBPM outperforms DeepWalk + ST + GAT by 3.76% in terms of precision and by 0.40% in terms of AUC. Moreover, the performance is also better from the perspective of Recall and F1-score. Meanwhile, from the comparison among ST&HC + LR, ST&HC + SVM, and UserRBPM, we notice that UserRBPM achieve an improvement of 13.59% in terms of precision. Such improvement verifies that the end-to-end learning framework UserRBPM can effectively detect potential driving factors and predictive signals in retweeting behavior prediction.

Comparing the first four methods (HC + LR, ST&HC + LR, HC + SVM, and ST&HC + SVM) with our proposed UserRBPM, it can be shown that the model which takes handcrafted features as input hardly represents interaction effects, while network embedding technology and graph attention can effectively extract high-dimensional structural attributes and can express highly nonlinear interaction mechanisms. Furthermore, from the comparison between HC + LR and ST&HC + LR (HC + SVM and ST&HC + SVM), Figure 5 shows that ST&HC + LR is notably better than HC + LR for retweeting behavior prediction. It reveals that users' spatial-level features and temporal-level features are the potential driving factors of retweeting behavior in social networks. Additionally, we observe that ST&HC + LR performs 4.42% better than HC + LR in terms of precision, verifying that the spatial-level and temporal-level features we designed have improved the prediction performance to a certain extent.

From the comprehensive analysis of the part to the whole, the space-level features and time-level features we designed have improved prediction performance. The combination of structural attributes and node features can further

improve prediction performance. Therefore, it demonstrates the effectiveness of our UserRBPM framework for retweeting behavior prediction.

*6.1.2. Prediction Performance of Different Sampling Strategies.* When dealing with data imbalance, we apply downsampling based on microblog granularity, instead of completely random downsampling. This sampling method would ensure that the number of positive and negative samples covered by the same microblog is the same, and the number of microblogs covered by negative samples is sufficient.

We use three sampling methods to obtain different training models. Among them, the directly sampling method (DSM) represents that we directly extract relatively balanced samples based on the ratio of the original positive and negative samples, that is, the ratio between positive and negative samples is set to 1:3. The number of positive samples and negative samples in completely random downsampling (CRDM) and our downsampling method (ODM) is the same. Experiment results are illustrated in Figure 6. Compared to the completely random downsampling method, the model trained with the samples obtained by our downsampling method has better prediction performance. The better the prediction effect of the model obtained by the training data training, the more it shows that the dataset has universal significance and the learned model has a stronger generalization ability. In the original imbalanced datasets, the direct extraction of positive and negative samples with a ratio of 1:3 is simple, but the difference in the number of microblogs covered by the positive and negative samples is ignored. Therefore, the downsampling method based on microblog granularity is more suitable for the user retweeting behavior prediction problem that we researched.

*6.1.3. Comparative Analysis of Graph Convolution and Graph Attention.* Table 7 is the prediction performance of two variants of graph deep learning, that is, the experimental results of using graph convolutional network (GCN) and graph attention mechanism (GAT) to build models, respectively. In previous research work, we have seen the success of GCN in classification tasks. However, in the application scenarios of our work, we observe that the performance of GCN in models constructed by different graph embedding



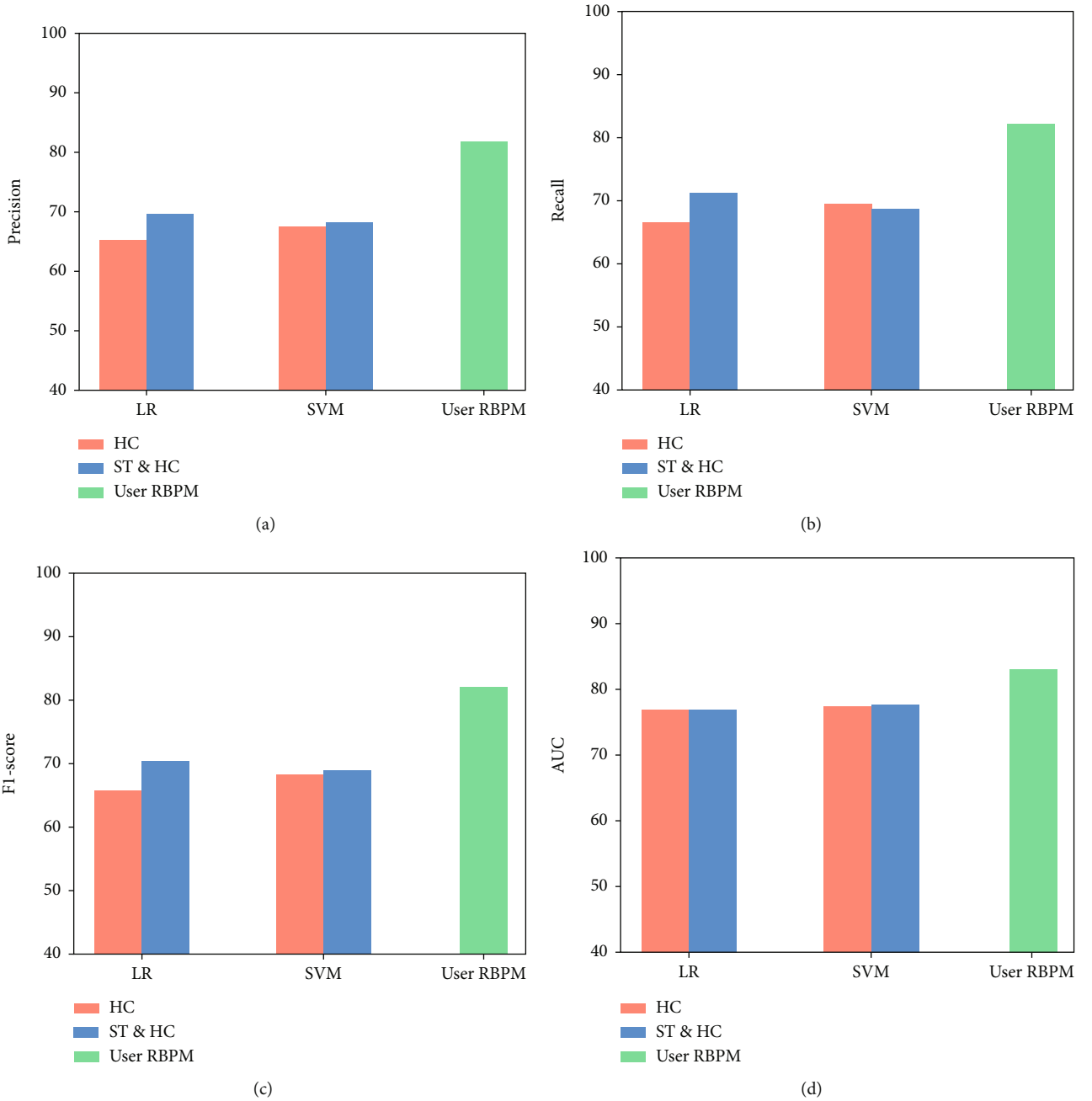


FIGURE 5: Prediction performance of different features.

technologies is generally worse than that of GAT. We attribute its disadvantage to the homophily assumption of GCN, that is, connected vertices tend to be similar (for example, have the same label). Under such assumption, for a specific vertex, GCN computes its hidden representation by taking an unweighted average over its neighbors' representation. This homophily exists in many real networks, but in our research scenario, different neighbor nodes may have different importance. Therefore, the graph attention mechanism (GAT) is introduced to assign different weights to different neighboring nodes. In essence, GAT is an aggregation func-

tion that focuses on the differences between neighbor nodes, rather than simple mean aggregation.

Besides, we wanted to avoid using handcrafted features and make UserRBPM a pure end-to-end learning framework, so we compared the prediction performance with additional vertex features and without additional vertex features. Comparison results of prediction performance with or without vertex features are presented in Table 8. It is observed that UserRBPM\_GCIN with handcrafted vertex features achieved an improvement of 2.18% in terms of precision, 1.46% in terms of recall, 0.44% in terms of F1-score, and

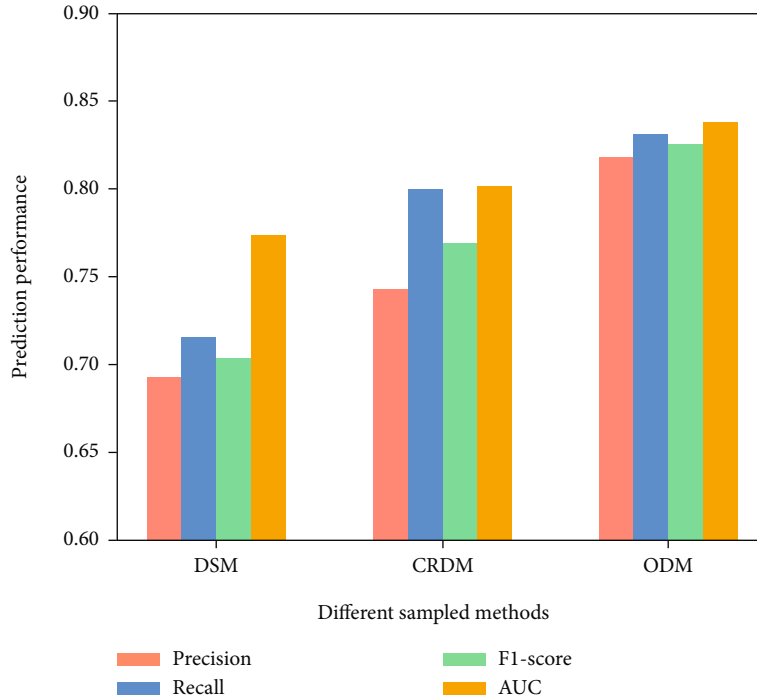


FIGURE 6: Prediction performance of different sampling strategies.

TABLE 7: Prediction performance of variants of UserRBPM (%).

Methods	Precision	Recall	F1-score	AUC
DeepWalk + ST&HC + GCN	77.49	79.28	78.37	74.89
DeepWalk + ST&HC + GAT	79.68	80.24	79.96	82.75
Node2vec + ST&HC + GCN	78.56	80.07	79.31	79.64
Node2vec + ST&HC + GAT	79.88	81.25	80.55	82.96
UserRBPM_GCIN	80.82	80.58	80.70	82.23
UserRBPM_GAT	81.97	82.58	82.27	83.21

TABLE 8: The effect of handcrafted vertex features on prediction performance (%).

Methods	Handcrafted vertex features	Precision	Recall	F1-score	AUC
UserRBPM_GCIN	Y	80.82	80.58	80.70	82.23
UserRBPM_GCIN	N	78.64	79.12	80.26	81.64
UserRBPM_GAT	Y	81.97	82.58	82.27	83.21
UserRBPM_GAT	N	80.49	81.77	81.12	81.92

0.59% in terms of AUC. UserRBPM\_GAT with handcrafted vertex features outperforms UserRBPM\_GAT without handcrafted vertex features by 1.48% in terms of precision, 0.81% in terms of recall, 1.15% in terms of F1-score, and 1.29% in terms of AUC. Experiment results demonstrate that, in addition to the pretrained network embedding, we can still obtain comparable performance even without considering handcrafted features.

**6.2. Parameter Sensitivity Analysis.** In addition, we consider parameter sensitivity in our work. We analyzed several hyperparameters in the model and tested how different hyperparameter choices affect the prediction performance.

**6.2.1. Robustness Analysis.** To verify the robustness of the UserRBPM framework, we changed the proportion of the training set, validation set, and test set and then redo the experiments. The results in Figure 7 show that the model is effective under limited training data size. Even with a small size of the training set (20%-40%), our model can still have an acceptable and steady performance.

**6.2.2. Effect of Instance Normalization.** As mentioned in Section 4, this paper studied the technique used to accelerate model learning called *Instance Normalization* (IN). This technique provides benefits to improve the classification performance. For instance, it can learn faster while maintaining or even increasing accuracy. Moreover, it also partially serves as a parameter tuning method. Therefore, we applied IN and obtained a boost in both performance and generalization. Figure 8 shows the changes in the training loss of UserRBPM\_GAT with or without the IN layer during training. We can see that when there is an instance normalization layer, as the number of epochs increases, the training loss first drops rapidly and then remains stable. Instance normalization significantly avoids overfitting and makes the training process more stable.

However, as shown in Figure 9, we observe that the model without the IN layer takes about 1011 seconds per epoch during the training process, while the model with the

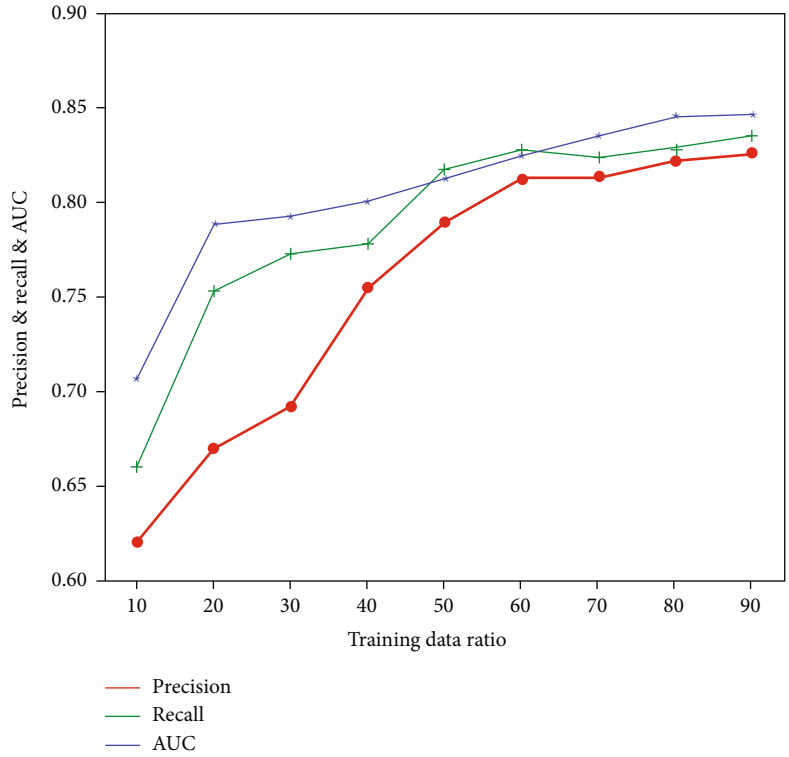


FIGURE 7: Prediction performance with different training and test data size.

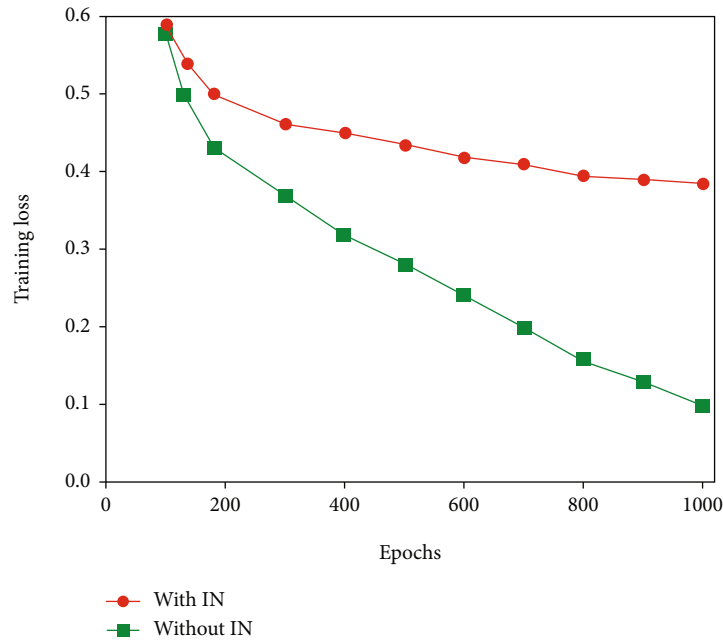


FIGURE 8: The effect of instance normalization.

IN layer takes about 1892 seconds per epoch. It was calculated that the model with the IN layer increased the training time for each epoch by about 87% compared to the model without the IN layer. Yet, we believe it is worthwhile to apply

IN, as the additional training time is compensated with a faster learning rate (it requires less number of epochs to reach the same level of precision) and can ultimately achieve higher testing precision.

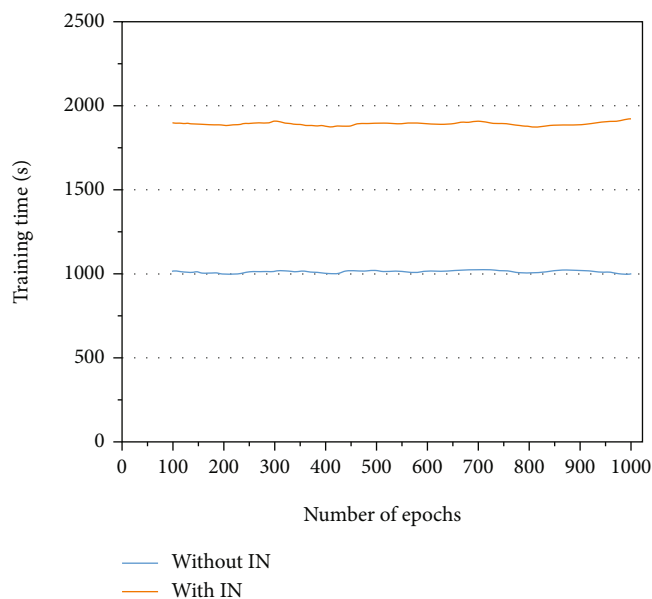


FIGURE 9: Time overhead during each epoch.

## 7. Conclusion

In this work, we focus on user-level social influence in social networks and formulate the problem of user retweet behavior prediction from a deep learning perspective. Unlike previous work that built a prediction model of retweet behavior based on network topology maps of information dissemination or feature engineering-based approaches, we proposed a UserRBPM framework to predict the action status of a user given the action statuses of her near neighbors and her local structural information. Experiments on a large-scale real-world dataset have shown that the UserRBPM significantly outperforms baselines with handcrafted features in user retweet behavior prediction. This work explores the potential driving factors and predictable signals in user retweet behavior in hope that the deep learning framework has the better expressive ability and prediction performance.

For future research, experimental datasets related to this research field still contain rich social dynamics, which are worth further exploration. We can study user behavior in a semisupervised manner, develop a generic solution based on heterogeneous graph learning, and then extend it to many network mining tasks, such as link prediction, social recommendation, and similarity search. Through such a learning scheme, we can leverage both unsupervised information and limited labels of users to build the predictor, and verify the effectiveness and rationality of user behavior analysis on real-world datasets.

## Data Availability

The original data used to support the findings of this study is from public data resources (<http://arnetminer.org/Influencelocality>). The data processing and parameter settings have been introduced Experiment Setup, so the data were not additionally provided.

## Disclosure

This is an extended version of a conference paper, and the conference name is EAI MOBIMEDIA 2021-14th EAI International Conference on Mobile Multimedia Communications.

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the National Key R&D Program of China (2017YFB0801805, 2016YFB0801100), the National Natural Science Foundation of China (62072359, 62072352), and the Key Research and Development Plan of Xinjiang Production and Construction Corps. (No. 2019AB001).

## References

- [1] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th International Conference on World Wide Web*, pp. 981–990, 2010.
- [2] F. Riquelme and P. González-Cantergiani, "Measuring user influence on Twitter: a survey," *Information Processing & Management*, vol. 52, no. 5, pp. 949–975, 2016.
- [3] T. R. Zaman, R. Herbrich, J. Van Gael, and D. Stern, "Predicting information spreading in twitter," in *Workshop on Computational Social Science and the Wisdom of Crowds, Nips*, vol. 104, no. 45pp. 17599–17601, Citeseer, 2010.
- [4] M. C. Yang, J. T. Lee, S. W. Lee, and H. C. Rim, "Finding interesting posts in twitter based on retweet graph analysis," in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1073–1074, 2012.
- [5] L. Hong, O. Dan, and B. D. Davison, "Predicting popular messages in twitter," in *Proceedings of the 20th International Companion on World Wide Web*, pp. 57–58, 2011.
- [6] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: a self-exciting point process model for predicting tweet popularity," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1513–1522, 2015.
- [7] W. Fang, X. Yao, X. Zhao, J. Yin, and N. Xiong, "A stochastic control approach to maximize profit on service provisioning for mobile cloudlet platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 4, pp. 522–534, 2016.
- [8] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn, "Social influence in social advertising: evidence from field experiments," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, pp. 146–161, 2012.
- [9] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: model and implications," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 6–14, 2012.

- [10] J. Wang and Y. Q. Wang, "SIR rumor spreading model with network medium in complex social networks," *Chinese Journal of Physics*, 2015.
- [11] Y. Liu, J. Zhao, and Y. Xiao, "C-RBFNN: a user retweet behavior prediction method for hotspot topics based on improved RBF neural network," *Neurocomputing*, vol. 275, pp. 733–746, 2018.
- [12] J. Xiong, X. Chen, Q. Yang, L. Chen, and Z. Yao, "A task-oriented user selection incentive mechanism in edge-aided mobile crowdsensing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2347–2360, 2020.
- [13] B. Yi, X. Shen, H. Liu et al., "Deep matrix factorization with implicit feedback embedding for recommendation system," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4591–4601, 2019.
- [14] R. Bi, Q. Chen, L. Chen, J. Xiong, and D. Wu, "A privacy-preserving personalized service framework through Bayesian game in social IoT," *Wireless Communications and Mobile Computing*, vol. 2020, 13 pages, 2020.
- [15] R. M. Bond, C. J. Fariss, J. J. Jones et al., "A 61-million-person experiment in social influence and political mobilization," *Nature*, vol. 489, no. 7415, pp. 295–298, 2012.
- [16] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 807–816, 2009.
- [17] Y. A. Kim and J. Srivastava, "Impact of social influence in e-commerce decision making," in *Proceedings of the Ninth International Conference on Electronic Commerce*, pp. 293–302, 2007.
- [18] J. Zhang, J. Tang, J. Li, Y. Liu, and C. Xing, "Who influenced you? Predicting retweet via social influence locality," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 3, pp. 1–26, 2015.
- [19] B. Jiang, J. Liang, Y. Sha et al., "Retweeting behavior prediction based on one-class collaborative filtering in social networks," in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 977–980, 2016.
- [20] J. Zhang, J. Tang, Y. Zhong et al., "Structinf: mining structural influence from social streams," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017no. 1.
- [21] Q. Li, Z. Han, and X. M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018no. 1.
- [22] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*, pp. 1725–1735, 2020.
- [23] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
- [24] J. Sun, X. Wang, N. Xiong, and J. Shao, "Learning sparse representation with variational auto-encoder for anomaly detection," *IEEE Access*, vol. 6, pp. 33353–33361, 2018.
- [25] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 793–803, 2019.
- [26] Y. Ota, K. Maruyama, and M. Terada, "Discovery of interesting users in twitter by overlapping propagation paths of retweets," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 3, pp. 274–279, Macau, China, 2012.
- [27] N. J. Yuan, Y. Zhong, F. Zhang, X. Xie, C. Y. Lin, and Y. Rui, "Who will reply to/retweet this tweet? The dynamics of intimacy from online social interactions," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pp. 3–12, 2016.
- [28] J. Tang, S. Wu, and J. Sun, "Confluence: conformity influence in large social networks," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 347–355, 2013.
- [29] X. Zhang, D. D. Han, R. Yang, and Z. Zhang, "Users' participation and social influence during information spreading on Twitter," *PLoS One*, vol. 12, no. 9, article e0183290, 2017.
- [30] Z. Luo, M. Osborne, J. Tang, and T. Wang, "Who will retweet me? Finding retweeters in Twitter," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 869–872, 2013.
- [31] C. Li, J. Ma, X. Guo, and Q. Mei, "DeepCas: an end-to-end predictor of information cascades," in *Proceedings of the 26th International Conference on World Wide Web*, pp. 577–586, 2017.
- [32] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang, "Retweet prediction with attention-based deep neural network," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 75–84, 2016.
- [33] X. Wang, Z. Guo, X. Wang, S. Liu, W. Jing, and Y. Liu, "Nnmlinf: social influence prediction with neural network multi-label classification," in *Proceedings of the ACM Turing Celebration Conference—China*, pp. 1–5, 2019.
- [34] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329, 2018.
- [35] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li, "Panther: fast top-k similarity search on large networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1445–1454, 2015.
- [36] J. Zhang, J. Tang, C. Ma, H. Tong, Y. Jing, and J. Li, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018no. 1.
- [37] F. Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: unsupervised and semi-supervised graph-level representation learning via mutual information maximization," 2019, <https://arxiv.org/abs/1908.01000>.
- [38] W. Chen, Y. Gu, Z. Ren et al., "Semi-supervised user profiling with heterogeneous graph attention networks," *IJCAI*, vol. 19, pp. 2116–2122, 2019.
- [39] W. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *The World Wide Web Conference*, pp. 417–426, 2019.
- [40] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
- [41] H. Tong, C. Faloutsos, and J. Y. Pan, "Fast random walk with restart and its applications," in *Sixth International Conference*

- on *Data Mining (ICDM'06)*, pp. 613–622, Hong Kong, China, 2006.
- [42] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, “Group formation in large social networks: membership, growth, and evolution,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 44–54, 2006.
  - [43] J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg, “Structural diversity in social contagion,” *Proceedings of the National Academy of Sciences*, vol. 109, no. 16, pp. 5962–5966, 2012.
  - [44] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, 2016.
  - [45] P. Cui, X. Wang, J. Pei, and W. Zhu, “A survey on network embedding,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.
  - [46] H. Huang, J. Tang, L. Liu, J. D. Luo, and X. Fu, “Triadic closure pattern analysis and prediction in social networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3374–3389, 2015.
  - [47] Y. Jin, G. Song, and C. Shi, “GraLSP: graph neural networks with local structural patterns,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 4, pp. 4361–4368, 2020.
  - [48] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: the missing ingredient for fast stylization,” 2016, <https://arxiv.org/abs/1607.08022>.
  - [49] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, “DeepInf: social influence prediction with deep learning,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2110–2119, 2018.
  - [50] H. Wu, Z. Hu, J. Jia, Y. Bu, X. He, and T. S. Chua, “Mining unfollow behavior in large-scale online social networks via spatial-temporal interaction,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34no. 1, pp. 254–261, 2020.
  - [51] Y. Yang, J. Jia, B. Wu, and J. Tang, “Social role-aware emotion contagion in image social networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016no. 1.
  - [52] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, 2014.
  - [53] A. Grover and J. Leskovec, “node2vec: scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, 2016.

## Research Article

# Privacy-Preserving Federated Learning Framework with General Aggregation and Multiparty Entity Matching

Zhou Zhou <sup>1,2</sup>, Youliang Tian <sup>1,2</sup> and Changgen Peng <sup>1,2</sup>

<sup>1</sup>State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

<sup>2</sup>Institute of Cryptography and Data Security, Guizhou University, Guiyang 550025, China

Correspondence should be addressed to Youliang Tian; [youliangtian@163.com](mailto:youliangtian@163.com)

Received 3 January 2021; Revised 27 January 2021; Accepted 19 June 2021; Published 28 June 2021

Academic Editor: Zhipeng Cai

Copyright © 2021 Zhou Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The requirement for data sharing and privacy has brought increasing attention to federated learning. However, the existing aggregation models are too specialized and deal less with users' withdrawal issue. Moreover, protocols for multiparty entity matching are rarely covered. Thus, there is no systematic framework to perform federated learning tasks. In this paper, we systematically propose a privacy-preserving federated learning framework (PFLF) where we first construct a general secure aggregation model in federated learning scenarios by combining the Shamir secret sharing with homomorphic cryptography to ensure that the aggregated value can be decrypted correctly only when the number of participants is greater than  $t$ . Furthermore, we propose a multiparty entity matching protocol by employing secure multiparty computing to solve the entity alignment problems and a logistic regression algorithm to achieve privacy-preserving model training and support the withdrawal of users in vertical federated learning (VFL) scenarios. Finally, the security analyses prove that PFLF preserves the data privacy in the honest-but-curious model, and the experimental evaluations show PFLF attains consistent accuracy with the original model and demonstrates the practical feasibility.

## 1. Introduction

In 2016, AlphaGo used 300,000 sets of flag games as training data and beat the world's top professional go players. Artificial intelligence (AI) has shown great potential and is expected to show itself in many fields and make important contributions [1]. In traditional AI, data processing needs to aggregate a large amount of data for model training. However, due to industry competition, privacy protection requirements, business management, and other issues, data of various industries forms islands, which are difficult to share. Therefore, data quality and availability is one of the constraints on AI development [2]. On the other hand, data privacy and security have become the focus of the world's attention [3] following the devastating losses caused by data leaks in recent years. The European Union recently introduced a new law—General Data Protection Regulations (GDPR) [4]—that shows the increasingly strict management of user data privacy and security will be the world trend. So

the enactment of laws and regulations also brings new challenges to the traditional AI processing mode.

How to solve the problem of data isolation and data fusion on the premise of protecting users' privacy has become an urgent task for the development of AI. The federated learning (FL) framework, first proposed by Google in 2016 [5, 6], well meets those requirements. In the FL model, each participant keeps the local data training model, only transmits the parameters of each model to an aggregation server using the new cryptography technology, and the server returns the aggregation parameters to each participant for updating after the completion of parameter aggregation [7]. In the end, establishing the virtual common model, using the encryption mechanism to complete the parameter exchange is consistent with the optimal model trained from the data aggregation [8] under the traditional model. Recently, research of the federated learning has become a hot topic, and a lot of deep learning works focusing on privacy protecting have been done. In 2019, Yang et al.

systematically introduced the federated learning framework, application, and research direction [9], which helps us to control and understand federated learning as a whole. The federated learning framework has been applied and extended to Deep Neural Networks (DNN), eXtreme Gradient Boosting (XGBoost), Random Forest (RF), and other algorithms [2, 10, 11], of which the used techniques include secret sharing [12], differential privacy [13], and homomorphic cryptography [14].

At present, there are still the following problems about privacy-protecting FL. First, there are few protocols that involve multiple user entity matching [15] and ensuring their privacy in concurrent mode. In addition, too much interaction between users is required when encrypted gradients or parameters are passed to the server. Furthermore, the protocol only considers that all users participate online throughout the training cycle without going offline or that the recovery of correct data requires the assistance of other participants when one participant goes offline. Finally, the existing schemes are of poor generality and are often only for specific machine learning algorithms and application scenarios.

To solve the above problems, we propose a novel PFLF with general aggregation and multiparty entity matching. In this framework, we propose a general aggregation model (GAM) that can be used in many applications where aggregation is required and privacy is protected. Under our GAM, we construct a multiparty entity matching protocol (MEMP), which can complete the confirmation of the common user data without leaking any disjoint entity information. In addition, we design the vertical federated logistic regression (VFLR) algorithm while keeping the data in the local database. In summary, our contributions can be summarized as follows:

- (i) We propose a PFLF that includes the GAM, MEMP, and VFLR to achieve multiscenario data aggregation, multiparty matching, and privacy protections
- (ii) We exploit the homomorphic encryption and the improved Shamir secret reconstruction to ensure that only the aggregator receives messages from at least  $t$  participants; it can recover the secret and remove mask to acquire correct parameters or product. In the GAM, there is little interaction with other participants
- (iii) We propose MEMP to confirm common entities based on GAM and the multiplicative homomorphism of RSA. It can enable participants with different data characteristics to determine common entities without leaking or inferring other useful information
- (iv) We design a privacy-preserving VFLR by using Paillier homomorphic encryption and merging GAM and LR. It can train a secure VFLR and support the withdrawal of participants. In addition, the prediction accuracy of the model is not affected
- (v) We give a comprehensive security analysis for our framework. We claim that the attackers will not acquire any useful information even if there is no

more than  $t$  participant collusion. Besides, extensive experiments are operated to confirm that our framework is effective and efficient

The rest of the paper is organized as follows. In Section 2, we describe the preliminaries and the main technology. In Section 3, we describe the system architecture, security model, and problem description. In Section 4, we describe the algorithm details of our GAM with privacy protection and construct the secure MEMP and VFLR model. In Section 5, we demonstrate the security analysis of framework. Experimental evaluation and related work are discussed in Sections 6 and 7, respectively. Finally, we give the conclusion of the paper in Section 8.

## 2. Preliminary

*2.1. Logistic Regression Algorithm.* Consider a dataset  $\{x^{(i)}, y^{(i)}\}_{i=1}^m$  with  $d$  dimension, in which  $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}) \in \mathbb{R}^d$ ,  $y^{(i)} \in (0, 1)$ . The predicted value is mapped between 0 and 1 by the sigmoid function [16]  $h_\theta(x) = 1/(1 + e^{-\theta^T x})$ , where  $\theta^T x = \sum_{j=0}^d \theta_j x_j$  and  $x_0 = 1$ . The objective function is defined as follows:

$$L(\theta) = \frac{1}{m} \sum_{i=1}^m \left[ -y^{(i)} \log(h_\theta(x^{(i)})) - (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^d \theta_j^2. \quad (1)$$

The gradient descent method is used to minimize the value of the objective function, and the model parameters are updated as follows:

$$\theta_j = \theta_j - \frac{\alpha}{m} \sum_{i=0}^m \left( h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{m} \theta_j. \quad (2)$$

When given a new data  $x^{\text{new}}$ , the predictive value of logistic regression is set to

$$y^{\text{new}} = \begin{cases} 1 & h_\theta(x^{\text{new}}) \geq 0.5, \\ 0 & h_\theta(x^{\text{new}}) < 0.5. \end{cases} \quad (3)$$

*2.2. Homomorphic Encryption.* The Paillier scheme satisfying the additive homomorphism [17] is as follows:  $p, q$  are large primes of equal length chosen randomly;  $n = pq$ ,  $\varphi(n) = (p-1)(q-1)$  are calculated. Given the random number  $g \in \mathbb{Z}_n^*$ , then we have the public key  $\text{pk} = (n, g)$  and the private key  $\text{sk} = (\varphi(n), \varphi(n)^{-1} \bmod n)$ . For the encryption, given the random number  $r$  satisfying  $0 < r < n, r \in \mathbb{Z}_n^*$ , we have the ciphertext  $C = g^m r^n \bmod n^2$ , where  $M$  is the plaintext. In the decryption phase,  $m$  is obtained by computing  $m = L(c^{\varphi(n)} \bmod n^2) \varphi(n)^{-1}$ , where  $L(x) = (x-1)/n$ .



We mainly use the following properties of Paillier homomorphic encryption. Additivity can be indicated as  $D(E(m1, r1)E(m2, r2) \bmod n^2) = m1 + m2$ .

**2.3. Secret Sharing.** The secret sharing (SS) scheme [18] adopted in our scheme is used to mask the data ciphertext transmitted by participants, but ensures that the aggregator recovers the ciphertext product and it also helps the scheme support the withdrawal of participants. For  $(t, n)$  SS scheme, the secret  $s$  is split into  $n$  shares.  $s$  can be recovered only if at least  $t$  random shares are provided. The share generation algorithm is illustrated as  $SS.share(s, t, n) = \{(u, s_u) \mid u \in U\}$ , in which  $n$  represents the number of participants involved in SS and  $U = \{1, 2, \dots, n\}$  is the set of participants and  $s_u$  is the share for each user  $u$ . The secret can be recovered by at least  $t$  participants contained in  $U'$  ( $U' \subseteq U$ ) using Lagrange interpolation base  $SS.recon(\{(u, s_u) \mid u \in U'\}, t)$  as follows:

$$s = \sum_{i=1}^{|U'|} s_i t_i(0), \quad (4)$$

where  $t_i(0) = \prod_{j=1, j \neq i}^{|U'|} id_j / (id_j - id_i)$  is computed. Here, we can use  $id_i$  representing the identity of the  $i$ th participant.

### 3. System Architecture

In this section, we introduce the system architecture, illustrated in Figure 1. Some frequently used notations of the paper are listed in Table 1.

**3.1. System Model.** Our framework involves three types of participating entities: a key generation center (KGC), a center server (CS), and a set of average participants (AP). Details are presented as follows.

*Key Generation Center.* KGC primarily performs key generation and distribution. Its main purpose is to initialize the system, generate public and private keys for homomorphic encryption, generate subsecrets based on Shamir secret sharing, assign corresponding public and private keys to CS, and distribute subsecrets to each general participant. Afterwards, it will go offline.

*Center Server.* CS is often the initiator of a federated learning mission. It is the one who has data labels, coordinating the execution of the entire process. It aggregates the parameters uploaded by all online participants. In MEMP, it calculates the user intersection and returns the common entity. In VFLR, it returns the calculated sample error. In the process, we hope that CS can infer nothing except the uploaded ciphertext and the final result.

*Average Participants.* AP refers to participants who participate in model training without tags. It involves multiple average participants, namely,  $AP = \{U_1, U_2, \dots, U_n\}$ . In the aggregation framework, they are usually done with local encryption and send values for aggregation.

**3.2. Security Model.** Based on [19], in our scheme, we assume that the interaction channels through CS and AP are secure and not subject to risks such as tampering, and all partici-

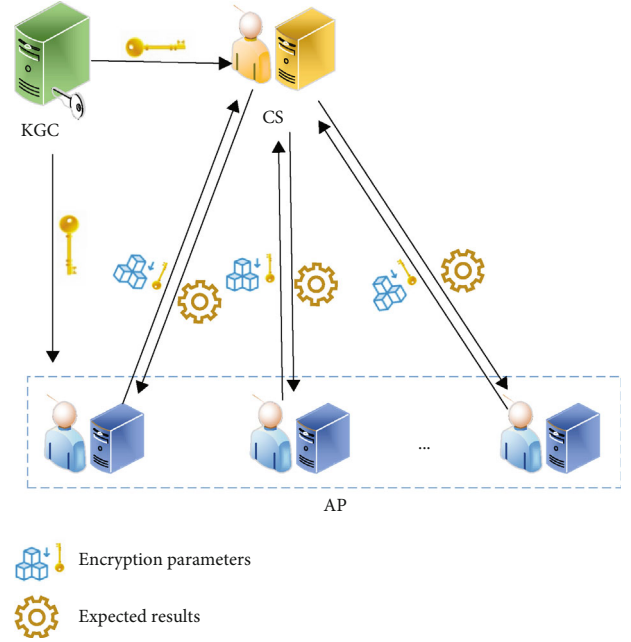


FIGURE 1: System architecture.

pants except KGC are considered to be honest-but-curious. KGC is a trusted party which always performs its tasks honestly and does not collude with any entity. CS and AP honestly follow the agreed process, but may try to learn all possible information that is of interest to them from their received messages. We define a threat model with an honest-but-curious adversary  $\mathcal{A}$  who can corrupt at most  $t - 1$  parties and obtain their inputs or other private information. In the entity matching protocol, what  $\mathcal{A}$  wants to know is users' information and CS's private key. In the model building and prediction phase,  $\mathcal{A}$  makes full use of the information it holds to learn about the data including data characteristics and weights of other honest parties. Our model needs to meet the following security requirements.

*Data Privacy.* CS cannot recognize any private data uploaded by  $U_i$ , and other  $U_j$  ( $j \neq i$ ) cannot infer the private data of others. For example, mark matrix and model parameters must not be exposed.

*Secure Withdrawal.* CS and  $U_i$  cannot continue to use the information of the exiting participants for subsequent calculations, and the process of recovering the aggregated value cannot reveal the parameters of the exiting participants if any participant drops in a round. There should be a safe way to deal with delayed transmissions and not be mistaken for offline.

**3.3. Problem Description.** In order to achieve a GAM that can enable aggregated messages to be decrypted only if they come from at least  $t$  participants, while ensuring that individual parameter ciphertext is not exposed, we introduce some cryptographic tools. For example, the transformed Shamir secret sharing scheme helps achieve threshold aggregation and cover the homomorphic ciphertext of each participant. Homomorphic encryption features facilitate obtaining the product or sum of parameter plaintext through ciphertext

TABLE 1: Notation table.

Notation	Description
$E_k$	The homomorphic encryption with public key $k_p$
$e, d$	The public key and private key for RSA
$\mathbb{Z}_p^*$	A cyclic group of order $P$ , primitive $g$
$s_i$	The secret share distributed to the participants $U_i$
$X_k^i$	The $k$ th sample of the $i$ th participant
$\Theta_i$	The characteristic weight of the $i$ th participant
$H()$	A hash function
$h_\theta()$	Function prediction of logistic regression

aggregation. Furthermore, the same entity between different participants needs to be determined for multiple participants with different characteristics in the vertical federated mode. Firstly, we should design a MEMP with privacy protection, through which multiple participants obtain their overlapping entity IDs without exposing their respective data. After then, we use these common entities' data with different characteristics to train the learning model while ensuring local data privacy. To achieve these two goals, under our aggregation model, we use RSA blind signature to generate data identity libraries, record the matching results by token matrix, and use RSA and Paillier as homomorphic encryption for specific functional requirements. In particular, the prediction accuracy of VFLR realized by the framework is unchanged, and it can support the withdrawal of participants.

#### 4. Construction of PFLF

Our PFLF implements a systematic FL process, including three main functions. Firstly, it can realize multiparty data aggregation without data leakage. Secondly, it can find the common set of entities of multiple participants without revealing useful information. For VFL scenarios, subsequent joint training can only be completed if the common entities are identified. When using logistic regression algorithm in VFL scenario, secure data aggregation is necessary after entity matching is completed. So, thirdly, we design the VFLR. In particular, the aggregation in our framework is generic, not only for a specific machine learning algorithm but also for all application scenarios based on thresholds. In this section, we present the details of our GAM and its role in constructing MEMP and VFLR.

*4.1. A Novel GAM.* A common aggregation model is suitable for such application scenarios where the aggregation server CS can decrypt and obtain the desired results through homomorphic encryption only when messages received are from at least  $t$  participants. Through this model, the participants' private data is fully protected in the process of achieving the interaction purpose according to the protocol, and when there are participants offline, the aggregated messages that do not involve the offline information can be recovered quickly. Here, firstly, based on the mentioned SS scheme, we can make the following transformation.

Each user  $U_i$  chooses a random number  $\beta_{ik}$ , where  $i = \{1, 2, \dots, t\}$  is the number of participants who reconstructed the secret and  $k$  is the number of samples. For the secret reconstruction formula  $s = \sum_{i=1}^{|U'|} s_i t_i(0)$ , let us multiply both sides of this equation by random numbers  $\beta_{ik}$  and the equation is transformed into the following form:

$$\begin{aligned} \beta_{1k}s &= \beta_{1k}(s_1 t_1(0) + s_2 t_2(0) + \dots + s_t t_t(0)), \\ \beta_{2k}s &= \beta_{2k}(s_1 t_1(0) + s_2 t_2(0) + \dots + s_t t_t(0)), \\ &\dots \\ \beta_{tk}s &= \beta_{tk}(s_1 t_1(0) + s_2 t_2(0) + \dots + s_t t_t(0)). \end{aligned} \quad (5)$$

We can sum both sides of this equation and get

$$\sum_{i=1}^t \beta_{ik}s = \sum_{i=1}^t \beta_{ik}s_1 t_1(0) + \sum_{i=1}^t \beta_{ik}s_2 t_2(0) + \dots + \sum_{i=1}^t \beta_{ik}\beta_{ik}s_t t_t(0). \quad (6)$$

When used for threshold encryption, it converts to

$$\begin{aligned} &\left\{ \begin{matrix} \sum_{i=1}^t \beta_{ik}s_1 t_1(0) \\ g^{\sum_{i=1}^t \beta_{ik}s_1 t_1(0)} \end{matrix} \right\} \left\{ \begin{matrix} \sum_{i=1}^t \beta_{ik}s_2 t_2(0) \\ g^{\sum_{i=1}^t \beta_{ik}s_2 t_2(0)} \end{matrix} \right\} \dots \left\{ \begin{matrix} \sum_{i=1}^t \beta_{ik}s_t t_t(0) \\ g^{\sum_{i=1}^t \beta_{ik}s_t t_t(0)} \end{matrix} \right\} \\ &= \left\{ \begin{matrix} \sum_{i=1}^t \beta_{ik}s_1 t_1(0) + \sum_{i=1}^t \beta_{ik}s_2 t_2(0) + \dots + \sum_{i=1}^t \beta_{ik}s_t t_t(0) \\ g^{\sum_{i=1}^t \beta_{ik}s_1 t_1(0) + \sum_{i=1}^t \beta_{ik}s_2 t_2(0) + \dots + \sum_{i=1}^t \beta_{ik}s_t t_t(0)} \end{matrix} \right\} \prod_{i=1}^t E_k(m_{ik}) \\ &= \left\{ \begin{matrix} \sum_{i=1}^t \beta_{ik}s \\ g^{\sum_{i=1}^t \beta_{ik}s} \end{matrix} \right\} \prod_{i=1}^t E_k(m_{ik}). \end{aligned} \quad (7)$$

We assume that in such a scenario, each participant  $U_i$  having a message  $m_{ik}$  needs to ask CS to help calculate  $\sum_{i=1}^t m_{ik}$ , but does not want to disclose  $m_{ik}$  to others and also does not want CS to infer some private data through the message sent by themselves to carry out various possible calculations. Figure 2 shows its workflow. In this way, they can do it like this:  $U_i$  first computes  $g^{\sum_{i=1}^t \beta_{ik}s_i t_i(0)} E_k(m_{ik})$  and sends it to CS according to the above equation, the receiver with the private key  $k_s$  can decrypt and get  $\sum_{i=1}^t m_{ik}$  when each participant makes public the value  $g^{\beta_{ik}s}$  corresponding to  $\beta_{ik}$ , and there exists the secret  $s = k_p$  and the public key  $k = k_p$ . Besides,  $E_k()$  satisfies homomorphic encryption which refers to multiplicative homomorphism in our entity matching protocol and additive homomorphism in our joint model training. How the model supports users' withdrawal is described in detail in Section 4.3. If the model is used for horizontal federated learning,  $m_{ik}$  can be gradient and other important parameters. Later in the VFLR section, we will focus on using logistic regression to explain the framework.

*4.2. Secure Multiparty Entity Matching.* As shown in Algorithm 1, the secure multiparty entity matching protocol

completes the confirmation of the common entity of multiple participants under the premise of protecting privacy. In the protocol, there is a CS with data sample identity  $u_{sk}$  and a set of average participants  $U = \{U_1, U_2, \dots, U_n\}$  with their own data sample  $u_{ik}$ , which represents the identity of the  $k$  th sample for the  $i$  th participant. Note that we briefly describe the situation of sending a message  $M$  from  $A$  to  $B$  as  $A \Rightarrow B : M$ . The protocol workflow is shown in Figure 3. The process of the protocol is shown as below.

In the initial parameter setting phase, the CS sets the penalty term, coefficient  $\lambda$ , and maximum iteration number  $\text{max\_iter}$  of the model.  $T$  generates a public-private key  $(k_p, k_s)$  for homomorphic encryption of the later model building and predicting. The algorithm introduces an RSA encryption with a blind factor to mask confidential information, so the public-private key  $((N, e), d)$ ,  $((N, e_i), d_i)$ ,  $(i \in (1, 2, \dots, n))$  are generated by  $T$ . In the following, we omit the modulus  $N$  for RSA. In addition,  $T$  also generates subshares  $s_i$  and  $s'_i$  of the public key  $k_p$  and  $e$  for  $U_i$  based on the identity of  $U_i$  using  $\text{SS.share}(k_p, t, n)$  or  $\text{SS.share}(e, t, n)$ .

In the exchange of information phase, each participant  $U_i$  chooses a random number  $r_i$ , computes  $v_{ik} = (r_i)^e (H(u_{ik})^{d_i})$ , and sends it to CS. CS chooses a random number  $r_{cs}$  to reflect the randomness of the interaction and uses the private key  $d$  for signature  $(v_{ik})^d r_{cs}$  and sends disturbed  $c_{ik} = r_i (H(u_{ik})^{d_i})^{d_i} r_{cs}$  to  $U_i$ . Disturbing the order of  $c_{i1}, c_{i2}, \dots, c_{ik}$  can eliminate the correspondence between  $c_{ik}$  and  $u_{ik}$  so that  $U_i$  cannot lock the IDs corresponding to the intersection of  $U_i$  and CS in the comparison phase. Furthermore, CS calculates random identifiers of each entity for different participants:  $d_{ij} = H(H(u_{sj})^d r_{si}^{e_i})$ . Then, CS sends  $[d_{11}, d_{12}, \dots, d_{1j}], \dots, [d_{n1}, d_{n2}, \dots, d_{nj}]$  to corresponding participants  $U_1, U_2, \dots, U_{n-1}$  for comparison.

In the comparison phase, each participant  $U_i$  eliminates the blind factor and opens its signature for  $c_{ik}$  to obtain  $(H(u_{ik})^d r_{si}^{e_i})$  and uses the hash function to compute  $g_{ik} = H((H(u_{ik})^d r_{si}^{e_i}))$ . By comparing them with  $d_{ij}$ , CS gets an  $l$ -dimensional matrix  $[b_{i1}, b_{i2}, \dots, b_{ij}]$ .  $b_{ij} = 1$  if  $d_{ij}$  belongs to  $g_{ik}$ . If not,  $b_{ij} = a_{ij}$ , where  $a_{ij}$  is a random number and  $a_{ij} > 1$ . In this way, each participant creates a comparison matrix.

In the solution phase, each participant  $U_i$  chooses a set of random number  $P_{ij}$  (i.e.,  $\{p_{11}, p_{12}, \dots, p_{1j}\}, \dots, \{p_{n1}, p_{n2}, \dots, p_{nj}\}$ ) and encrypts its matrix by computing  $g^{s_i t_i(0) \sum_{j=1}^n p_{ij}} (b_{ij})^e$  with RSA.

After then,  $U_i$  sends their encrypted matrix to CS for aggregation. At last, CS aggregates matrix values of  $n$  participants by computing  $\prod_{i=1}^n g^{s_i t_i(0) \sum_{j=1}^n p_{ij}} (b_{ij})^e$  and uses its private key  $d$  to decrypt, obtaining  $\prod_{i=1}^n b_{ij}$ . Because CS can recover the secret  $e$  by computing  $\sum_{i=1}^n t_i(0)$ . Specific analysis can refer to Secure Model Building.

In the identification phase, CS finds the corresponding entity by the value of  $\prod_{i=1}^n b_{ij}$ . Because each  $d_{ij}$  comes from CS, then if it is 1, it means that each participant has a corresponding  $d_{ij}$ , and if not, it means that at least one participant

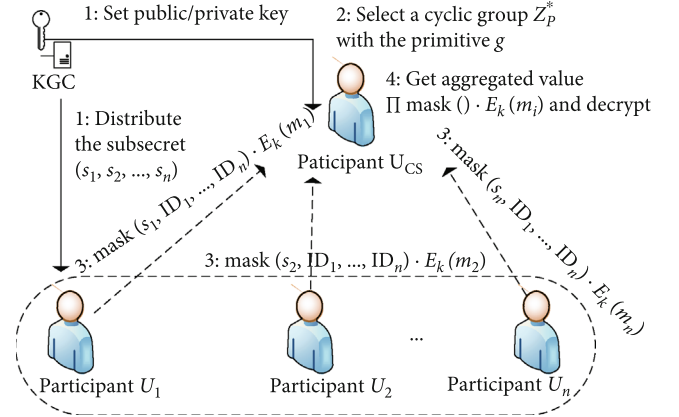


FIGURE 2: Workflow of the GAM.

does not have  $d_{ij}$ . Therefore, CS can find the right entity  $u_{sj}$  based on  $\prod_{i=1}^n b_{ij}$ . In the end, CS broadcasts common entity IDs to other participants for following model training.

### 4.3. Secure Model Building

**4.3.1. Secure Training.** For logistic regression to find the better model with gradient descent method, the part that needs to be computed jointly is error using the predicted value and the sample label. It is better for CS to do the aggregation and calculation since sample labels are mastered by CS. The workflow is shown in Figure 4. To protect the confidentiality of each participant's data, the aggregated data is received in ciphertext. Consequently, we chose to use Paillier homomorphic encryption to do the computation. However, each participant's data cannot be decrypted separately by CS; for this reason, we apply Shamir secret sharing scheme such as Formula (7) to ensure that CS could decrypt only after the aggregation operation was completed. The detailed process is shown in Algorithm 2.

The number of common entities of all participants is  $m$ , the average participants in the joint modeling are  $U_i (U_i \in U)$ , and each average participant secretly keeps a subsecret as  $s_i$ . Now given a cyclic group  $G$  and its primitive  $g$ ,  $U_i (i \in [1, n])$  computes  $D_k^i = \Theta_i X_k^i$ , in which  $X_k^i$  is the  $k$  th sample for the  $i$  th participant  $U_i$ . With subsecret  $s_i$  and identities,  $U_i$  can compute its own  $s_i t_i(0)$ . In order to keep the subsecret  $s_i$  dynamic, for each sample,  $U_i$  adds a random factor or the time stamp  $\beta_{ik}$  to calculate  $C_k^i = g^{s_i t_i(0) \sum_{i=1}^n \beta_{ik}} E_{k_p}(D_k^i)$  and sends it to CS after other participants release their  $g^{\beta_{ik}}$ . At this point, although CS gets the ciphertext of each participant, he cannot decrypt it because he cannot get the subsecret  $s_i$ . Only when messages are received from at least  $t$  participants can aggregation  $\prod_{i=1}^n C_k^i$  that can be generated, i.e.,

$$\prod_{i=1}^n C_k^i = \prod_{i=1}^n g^{s_i t_i(0) \sum_{i=1}^n \beta_{ik}} E_{k_p}(D_k^i) = g^{\sum_{i=1}^n \left\{ s_i t_i(0) \sum_{i=1}^n \beta_{ik} \right\}} E_{k_p} \left( \prod_{i=1}^n D_k^i \right), \quad (8)$$

**Input:** a central server CS, a set of participants  $U = \{U_1, U_2, \dots, U_n\}$ , and a trusted party  $T$ .

**Output:** common entity IDs.

- 1: CS sets the parameters for model training penalty,  $\lambda$ ,  $\max_i \text{iter}$ .
- 2:  $T$  generates a public-private key  $(k_p, k_s)$  for homomorphic encryption, a public-private key  $((N, e), d)$  for RSA encryption for CS, also generates average participants' public-private keys  $((N, e_i), d_i)$  and subshares of the public key  $k_p$  and  $e$  based on the identity of the participants, i.e.,  $T \Rightarrow U : \{(u, s_u) \mid u \in U\} = \text{SS.share}(k_p, t, n)$ . Here,  $U_1, U_2, \dots, U_n$  get subshares  $\{s_1, s_2, \dots, s_n\}$  for  $k_p$  and subshares  $\{s'_1, s'_2, \dots, s'_n\}$  for  $e$ .
- 3: Each participant  $U_i$  chooses a random number  $r_i$ , computes  $v_{ik} = (r_i)^e (H(u_{ik})^{d_i})$ , and operates  $U \Rightarrow CS : v_{ik}$ .
- 4: CS chooses a random number  $r_{s_i}$ , uses the private key for signature  $(v_{ik})^d r_{s_i}$ , gets each  $c_{ik} = r_i (H(u_{ik}))^{d_i} r_{s_i}$ , and returns them to  $U_i$  after disturbing the order of  $c_{ik}$ .
- 5: **for**  $i = 1 \rightarrow n$  **do**
- 6:   **for**  $j = 1 \rightarrow l$  **do**
- 7:     CS computes for the  $l$  entities:  $d_{ij} = H(H(u_{sj})^d r_{s_i}^{e_i})$ .
- 8: CS sends  $[d_{11}, d_{12}, \dots, d_{1j}], \dots, [d_{n1}, d_{n2}, \dots, d_{nj}]$  to corresponding participants  $U_1, U_2, \dots, U_{n-1}$ .
- 9: **for**  $i = 1 \rightarrow n$  **do**
- 10:   Each participant  $U_i$  eliminates the blind factor and  $d_i$  for  $c_{ik}$ , obtains  $(H(u_{ik}))^d r_{s_i}^{e_i}$ , and computes their hash values  $g_{ik} = H((H(u_{ik}))^d r_{s_i}^{e_i})$ .
- 11:   **for**  $j = 1 \rightarrow l$  **do**
- 12:     Each participant generates its own  $l$ -dimensional matrix  $[b_{i1}, b_{i2}, \dots, b_{ij}]$  by determining whether  $d_{ij}$  belongs to its  $g_{ik}$ .
- 13:     **if**  $d_{ij} \in g_{ik}$  **then**
- 14:        $b_{ij} = 1$ .
- 15:     **else**
- 16:        $b_{ij} = a_{ij}$ , where  $a_{ij}$  is a random number and  $a_{ij} > 1$ .
- 17: Each participant  $U_i$  chooses a set of random number  $p_{ij}$  and encrypts its matrix with  $g^{s_i r_i(0) \sum_{j=1}^n p_{ij}} (b_{ij})^e$ , operating  $U_i \Rightarrow CS$  : encryptedmatrix.
- 18: CS aggregates matrix values by computing  $\prod_{i=1}^{n,j} g^{s_i r_i(0) \sum_{j=1}^n p_{ij}} (b_{ij})^e$  and obtains  $\prod_{i=1}^{n,j} b_{ij}$  by decrypting.
- 19: **if**  $\prod_{i=1}^{n,j} b_{ij} == 1$  **then**
- 20:   CS finds the corresponding  $u_{sj}$ .
- 21: CS broadcasts  $u_{sj}$  to other participants  $U_1, U_2, \dots, U_{n-1}$ .
- 22: **return** common entity IDs:  $u_{sj}$ .

ALGORITHM 1: MEMP.

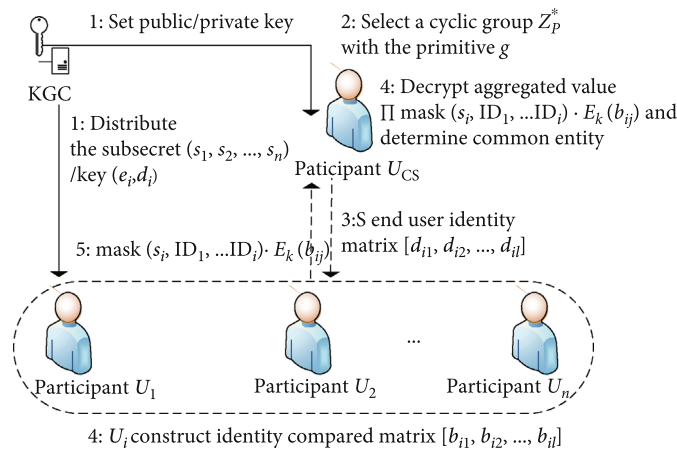


FIGURE 3: Workflow of the MEMP.

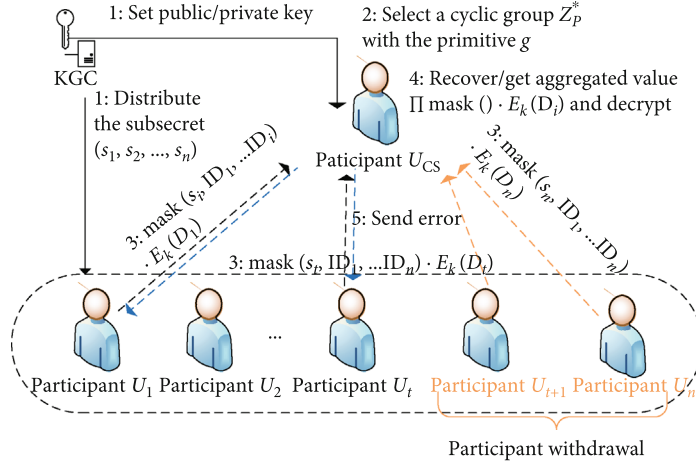


FIGURE 4: Workflow of the VFRL.

**Input:** a central server CS, a set of participants  $U = \{U_1, U_2, \dots, U_n\}$ , instance space of  $m$  samples of each participants, subshares  $\{s_1, s_2, \dots, s_n\}$ , cyclic group  $G$ , and its primitive  $g$ .

**Output:** federated logistic regression model.

1: **for**  $k = 1 \rightarrow m$  **do**

2: **for**  $i = 1 \rightarrow n$  **do**

3:  $U_i$  computes  $D_k^i = \Theta_i X_k^i$ .

4:  $U_i$  chooses random number  $\beta_{ik}$  and makes public its  $g^{\beta_{ik}}$ .

5:  $U_i$  uses others'  $g^{\beta_{ik}}$  to compute  $C_k^i = g^{s_{t_i}(0) \sum_{i=1}^n \beta_{ik}} E_{k_p}(D_k^i)$  and sends it to CS.

6: **for**  $k = 1 \rightarrow m$  **do**

7: **if** someone exits **then**

8: CS eliminates the value involving information of quitters in  $t_i(0)$ .

9: CS performs the aggregation  $\prod_{i=1}^n C_k^i$  and decrypts to get  $\sum_{i=1}^n \Theta_i X_k^i$ .

10: CS computes  $\text{error}_k = y^{(k)} - 1/(1 + e^{-(\sum_{i=1}^n \Theta_i X_k^i + \Theta_s X_k^s)})$ .

11: broadcasts  $\text{error}_k$ .

12: Each participant and CS can update weight parameter by computing  $\Theta_i = \Theta_i + (\alpha/m) \sum_{k=1}^m \text{error}_k X_k^i - (\lambda/m) \Theta_i$ .

13: Repeat all until reaching the termination condition.

14: **return** built model.

ALGORITHM 2: VFRL.

where  $g^{\sum_{i=1}^n \{s_{t_i}(0) \sum_{i=1}^n \beta_{ik}\}} = g^{\sum_{i=1}^n \beta_{ik} k_p}$ . Because  $k_p$  and  $g^{\sum_{i=1}^n \beta_{ik}}$  are public to CS, CS can decrypt and get  $\sum_{i=1}^n \Theta_i X_k^i$  to compute  $\text{error}_k$ . The next step, CS will broadcast  $\text{error}_k$  to current participants  $U'$  ( $U' \subseteq U$ ). Each participant and CS can update weight parameter by computing  $\Theta_i = \Theta_i + (\alpha/m) \sum_{k=1}^m \text{error}_k X_k^i - (\lambda/m) \Theta_i$ . All steps are repeated until the maximum number of iterations is reached.

**4.3.2. Withdrawal of Participants.** Some participants may withdraw from federated learning, such as being unwilling to contribute models or dropping offline. In order to deal with the above situation, we can make a contract to reduce the occurrence of withdrawal. It is assumed that each average participant  $U_i$  will be paid a certain amount based on their contribution in each iteration. The total expenditure and maximum number of iterations set by CS are Amounts and  $t$ . The contract signed by all participants is as follows: (1)  $n$  average participants submit a deposit to CS, and the deposit

is Amounts/ $t$ . (2) In the FL, if CS receives all the messages within the maximum allowable period, the errors will be returned normally to all participants according to the protocol. (3) Else, CS will send withdrawal confirmation request to participants who did not send the message. If they report it is delayed, the delayed messages can still be used to compute the aggregated value. But these delayed participants will not get paid this round. (4) Once the participant reports withdrawal, the deposit will be distributed to other online participants, including CS. Rational participants generally do not withdraw in order to maximize their own interests. (5) Upon completion of the FL, deposits will be returned to all the online participants. Particularly, if some participants withdraw during training, CS requires each participant to resend the message without the identity of the quitters in order to decrypt. Because of the randomness of  $C_k^i$ , CS cannot use the message sent twice to perform comparison calculation and get useful data. The reason is that CS still cannot get the subsecret to reconstruct the polynomial.

**Input:** federated logistic regression model, results inquirer  $R$ , and its instance space.  
**Output:** predicted results.  
1: **for**  $i = 1 \rightarrow n$  **do**  
2:  $R$ , respectively, encrypts the data  $E_{k_p}(X_k^i)$  belonging to the characteristics of different participants including CS.  
3: Each participant computes  $(E_{k_p}(X_k^i))^{\Theta_i}$  and communicates it to CS.  
4: CS does an aggregate operation  $(E_{k_p}(X_k^s))^{\Theta_s} \prod_{i=1}^n (E_{k_p}(X_k^i))^{\Theta_i}$  and returns it to  $R$ .  
5:  $R$  decrypts and gets  $\Theta_i X_k^i$ .  
6:  $R$  gets predicted results  $h$  by  $h_\theta(x) = 1/(1 + e^{-\theta^T x})$ .  
7: **return** result  $h$ .

ALGORITHM 3: Secure predicting.

**4.4. Secure Predicting.** Secure prediction should ensure that user data privacy is not compromised and that model parameters are not exposed. As described by Algorithm 3, results inquirer  $R$  intends to provide a set of data for prediction without privacy leakage and all data characteristics correspond to all participants in the current model. First,  $R$ , respectively, encrypts the data with  $E_{k_p}(X_k^i)$ , for example, ( $X_k^i$ ) belongs to a characteristic that corresponds to  $U_i$ . Each participant computes  $(E_{k_p}(X_k^i))^{\Theta_i}$  through  $R$ 's public key  $k_p$  and communicates it to CS. The aggregation operation is still done by CS to protect the parameters  $\Theta_i$  from being exposed. Then, CS computes  $(E_{k_p}(X_k^s))^{\Theta_s} \prod_{i=1}^n (E_{k_p}(X_k^i))^{\Theta_i}$  and returns it to  $R$ . The joint value  $\Theta_i X_k^i$  could be get with private key  $k_s$  by  $R$ , which computes predicted results  $h$  using  $h_\theta(x) = 1/(1 + e^{-\theta^T x})$ .

## 5. Security Analysis

In this section, we prove that our scheme is secure based on the simulator under the honest-but-curious setting. Recall that the involved parties are  $n$  participants  $U_1, U_2, \dots, U_n$  and the CS. We assume an honest-but-curious adversary  $\mathcal{A}$  who can corrupt participants but at most  $t-1$ . According to [20], we define the security of our framework by comparing the real interaction and ideal interaction. In the real interaction, there is an environment  $\mathcal{Z}$  which chooses inputs and receives outputs of uncorrupted participants. The adversary  $\mathcal{A}$  who can interact arbitrarily with the environment  $\mathcal{Z}$  forwards all received messages to  $\mathcal{Z}$  and acts as instructed by  $\mathcal{Z}$ . We let  $\text{REAL}[\mathcal{Z}, \mathcal{A}, \pi]$  represent the view of  $\mathcal{A}$ . Similarly, we let  $\text{IDEAL}[\mathcal{Z}, \mathcal{S}, \mathcal{F}]$  represent the view of  $\mathcal{S}$  where adversary  $\mathcal{S}$  and honest participants interact with the environment  $\mathcal{Z}$  running the dummy protocol in the presence of functionality  $\mathcal{F}$ .

**Definition 1.** A protocol  $\pi$  is secure if for every admissible adversary  $\mathcal{A}$  attacks the real interaction, there exists a simulator  $\mathcal{S} = [\mathcal{S}_u, \mathcal{S}_{cs}]$  attacking the ideal interaction, such that the environment  $\mathcal{Z}$  cannot distinguish between the ideal view of  $\mathcal{S}$  and the real view of  $\mathcal{A}$ .

**5.1. Security of GAM.** In the GAM, although CS can collude with at most  $t-1$  participants to obtain the privacy of honest participants, they get nothing but aggregated results. Since each participant's data is encrypted as  $g^{\sum_{i=1}^t \beta_{ik} s_{1i}(0)} E_k(m_{ik})$ , based on the security of Shamir secret sharing and homomorphic encryption, only the privacy of  $m_{ik}$  is discussed here.

**Theorem 2.** For secure aggregated framework, there exists a PPT simulator  $\mathcal{S}_u$  or  $\mathcal{S}_{cs}$  that can simulate the ideal view  $\text{IDEAL}[\mathcal{Z}, \mathcal{S}, \mathcal{F}_{gam}]$  which is computationally indistinguishable from the real view of  $\mathcal{A}_u$  or  $\mathcal{A}_{cs}$ .  $\mathcal{F}_{gam}$  is illustrated as Table 2, where a subset  $W \subseteq U \cup \text{CS}$  represents joint attackers.

According to whether CS is involved in collusion, the discussion is divided into two situations.

*Case 1. Excluding CS from  $W$ .*

*Proof.* Since CS is not compromised, the view constructed by simulator  $\mathcal{S}_u$  is independent of the input of CS. So the simulator  $\mathcal{S}_u$  can execute a simulation by asking  $\mathcal{F}_{gam}$  to generate fake data as inputs of the honest users, but the true inputs for honest-but-curious users. When sending  $b(r_i)f_k(x_i)$ , the simulator utilizes random number instead of true data. As aggregating and decrypting, CS returns aggregated result that does not indicate which special participants'  $b(r_i)f_k(x_i)$  are aggregated. Hence, the ideal view simulated by  $\mathcal{S}_u$  is indistinguishable from the real view of  $\mathcal{A}_u$  since it is impossible to determine that  $\sum_{i=1}^{|U \setminus W|} x_i$  is obtained from real data.  $\square$

*Case 2. Including CS in  $W$ .*

*Proof.* To prove the indistinguishability of the views in Case 2, the simulator gradually makes some improvements to the protocol. There exists  $\text{hyb1}, \text{hyb2}$  that imply secure modification to the original protocol, ensuring the indistinguishability of the changed protocol from the original protocol, in our hybrid argument.

$\text{hyb1}$ : in this hybrid, the simulator  $\mathcal{S}_{CS}$  generates the masked input for honest participants as below:

$$y_i = r_i E_k(m_{ik}), \quad (9)$$

TABLE 2: Definition of  $\mathcal{F}_{\text{gam}}$ .

---

Give function  $f_k(x_1, x_2, \dots, x_n) = k^{-1} \prod_{i=1}^n b(r_i) f_k(x_i)$  with homomorphic encryption function  $f_k$  and  $\prod_{i=1}^n b(r_i) = k$  when  $i \geq t$ .

$\mathcal{F}_{\text{gam}}$ 's operation is as follows:

- (1) On input (Input, sid,  $m_i$ ) from  $U_i$ , set  $x_i = m_i$  and send (Input, sid,  $U_i, |m_i|$ ) to adversary  $\mathcal{A}$ .
- (2) On input (Compute, sid,  $U_i$ ) from  $U_i$ , choose  $r_i \in_R Z_p^*$  randomly, compute  $b(r_i) f_k(x_i)$ , and send them to CS if  $i \geq t$ . If  $U_i \in W \subseteq U$  is corrupted, send  $(x_i, r_i)$  to it.
- (3) On input (Aggregate, sid, CS), compute  $f_k^{-1}(k^{-1} \prod_{i=1}^n \text{Enc}(r_i) f_k(x_i))$  and send it to CS.

---

instead of utilizing

$$y_i = g^{\sum_{i=1}^t \beta_{ik} s_i t_i(0)} E_k(m_{ik}). \quad (10)$$

Because  $g^{\sum_{i=1}^t \beta_{ik}}$  is a random number, we can get  $g^{\sum_{i=1}^t \beta_{ik} s_i t_i(0)}$  is also a random value. The DDH assumption ensures that it is easy to infer they are indistinguishable.

hyb2: in this hybrid, the simulator generates encrypted  $E_k(t_i)$  by replacing  $m_{ik}$  with a random number  $t_i$ . The security of the encryption algorithm ensures the indistinguishability of the two ciphertexts. Therefore, the simulator submits

$$y_i = r_i t_i, \quad (11)$$

instead of sending

$$y_i = r_i E_k(m_{ik}). \quad (12)$$

Accordingly, the simulation has been completed since  $\mathcal{S}_{\text{CS}}$  successfully simulates the real view without acquiring  $x_i(m_{ik})$  and subsecret  $s_i$  and we can infer that the output of this hybrid is indistinguishable from the real one.  $\square$

**5.2. Security of MEMP.** In the process of confirming the identity of the common entity, no entity information other than a common identity is available between participants. Thus, not only can the true identity of the entity not be exposed but its hash value cannot be either. The reason is that some honest-but-curious participants can calculate the hash value of a possible identifier to determine whether it belongs to  $U_i$ . It is necessary to cover the confidential information with a random factor similar to a blind signature. We prove the following two theorems by constructing two separate simulators  $\mathcal{S}_u$  and  $\mathcal{S}_{cs}$  to show that the real views and ideal views are computationally indistinguishable.

**Theorem 3.** For group entity matching, there exists a PPT simulator  $\mathcal{S}_u$  or  $\mathcal{S}_{cs}$  that can simulate the ideal view  $\text{IDEAL}[\mathcal{L}, \mathcal{S}, \mathcal{F}_{em}]$  which is computationally indistinguishable from the real view of  $\mathcal{A}_u$  or  $\mathcal{A}_{cs}$ .

TABLE 3: Definition of  $\mathcal{F}_{em1}$ .

---

$\mathcal{F}_{em1}$ 's operation is as follows:

- (1) On input (Init, sid, id,  $(ID_{i1}, ID_{i2}, \dots, ID_{ik})$ ) from  $U_i$ , generate random numbers  $(h_{i1}, h_{i2}, \dots, h_{ik})$  for the corresponding  $ID_{i1}, ID_{i2}, \dots, ID_{ik}$ , store  $(id_i, (ID_{i1}, h_{i1}), (ID_{i2}, h_{i2}), \dots)$ , and send  $(id_i, (ID_{i1}, h_{i1}), (ID_{i2}, h_{i2}), \dots)$  to corrupted adversary  $A$ .
- (2) On input (Init, sid, CS,  $(ID_{s1}, ID_{s2}, \dots, ID_{sl})$ ) from CS, generate  $l$  random numbers  $(h_{s1}, h_{s2}, \dots, h_{sl})$ . Once the IDs of the CS and  $U_i$  are equal, there sets  $h_{sj} = h_{it}$ , ( $j \in [1, k], t \in [1, l]$ ). For corrupted CS, send  $(id_s, (ID_{s1}, h_{s1}), (ID_{s2}, h_{s2}), \dots)$  to it. Store them.
- (3) On input (signature, sid,  $U_i, d_i$ ) from  $U_i$ , choose  $r_i \in_R Z_p^*$  randomly, and compute  $\text{sign}(d_i, r_i^{ee_i} h_{it})$ . Send them to CS and store  $(U_i, \text{sid}, r_i)$ .
- (4) On input (Bsignature, sid, CS,  $d, \text{sign}(d_i, r_i^{ee_i} h_{it})$ ) from CS, compute  $\text{sign}(d, \text{sign}(d_i, r_i^{ee_i} h_{it}))$ . Send  $r_{s_i} \text{sign}(d, \text{sign}(d_i, r_i^{ee_i} h_{it}))$  to  $U_i$ , where  $r_{s_i}$  are random numbers.
- (5) On input (Open, sid,  $U_i, \text{sign}(d, \text{sign}(d_i, r_i^{ee_i} h_{it}))$ ) from  $U_i$ . Check whether there exists  $(U_i, \text{sid}, r_i)$ . If exiting,  $h_{it}^d r_{s_i}^{e_i}$  can be got, or abandon it. Generate random  $H_{it}$  and store  $(U_i, \text{sid}, (h_{it}^d r_{s_i}^{e_i}, H_{it}))$ .
- (6) On input (Sign, sid, CS,  $U_i, d$ ), compute  $(\text{sign}(d, h_{sj} r_{s_i}^{ee_i}))$ . Generate random  $H_{sj}$  for every sign  $(d, h_{sj} r_{s_i}^{ee_i})$ . If existing sign  $(d, h_{sj} r_{s_i}^{ee_i}) = (h_{it}^d r_{s_i}^{e_i})$ , setting  $H_{sj} = H_{it}$  and storing  $(CS, \text{sid}, (h_{it}^d r_{s_i}^{e_i}, H_{sj}))$ . Send  $(CS, \text{sid}, H_{sj})$  to the corresponding  $U_i$ .
- (7) On input (Compare,  $U_i, \text{sid}, CS$ ), generate the  $l$ -dimensional comparison matrix using 1 and random numbers, where there exist  $H_{sj}$  in  $(U_i, \text{sid}, (h_{it}^d r_{s_i}^{e_i}, H_{it}))$ , 1 will be set, or a random number will be set.

---

Let us divide  $\mathcal{F}_{em}$  into two parts, such as  $\mathcal{F}_{em1}$  and  $\mathcal{F}_{em2}$ .  $\mathcal{F}_{em1}$  is the process from the beginning to the establishment of the comparison matrix, and  $\mathcal{F}_{em2}$  means the process from the encrypted comparison matrix to the end, similar to the previous  $\mathcal{F}_{gam}$ . The description of  $\mathcal{F}_{em1}$  is shown in Table 3.

Here, we only complete the proof of the ideal view  $\text{IDEAL}[\mathcal{L}, \mathcal{S}, \mathcal{F}_{em1}]$  which is computationally indistinguishable from the real view of  $\mathcal{A}_u$  or  $\mathcal{A}_{cs}$ . There are still two cases to prove.

*Case 1* (excluding CS from  $W$ ). Just consider the average participants  $U_i$  that are compromised.

*Proof.* Suppose a group of participants is corrupted in the beginning. Because the views of corrupted participants and the inputs of honest participants are irrelevant and the values of all honest participants are meaningless for corrupted participants according to the real protocol  $\pi_{em}$ , so the simulator  $\mathcal{S}_u$  first can run the protocol with the true inputs while using dummy data as the inputs of honest participants, namely,  $\mathcal{S}_u$  asks  $\mathcal{F}_{em1}$  to generate random numbers as the inputs of honest participants. After the blind signature is executed, the value returned from CS is added with a random number of

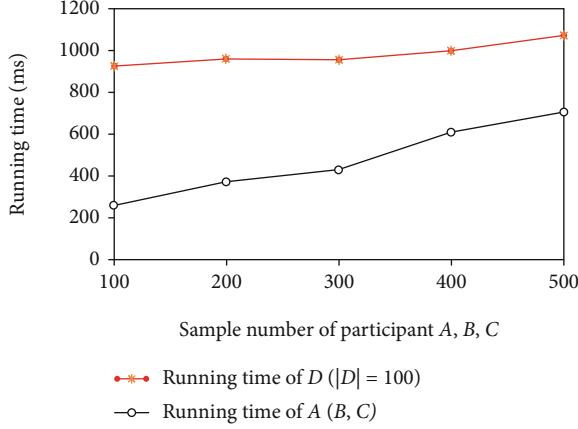
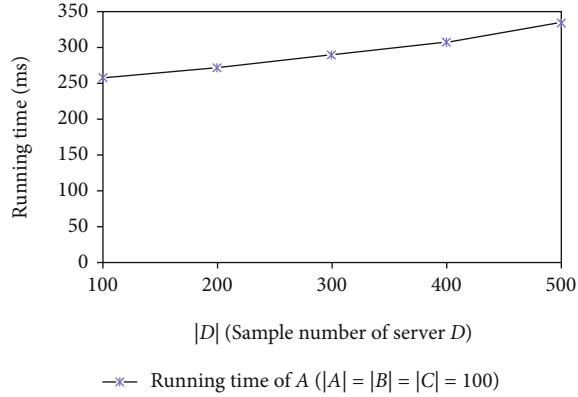
FIGURE 5: Running time with different sample sizes in  $U_i$ .

FIGURE 6: Running time of A with increasing samples in D.

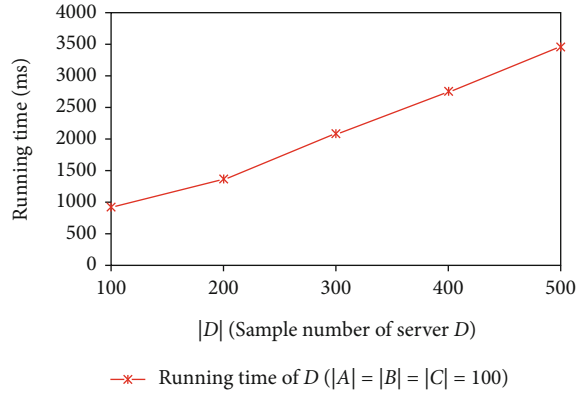


FIGURE 7: Running time of D with increasing samples in D.

CS so that  $\mathcal{S}_u$  cannot distinguish whether the values are generated from real data. Then, an  $l$ -dimensional matrix is generated by comparing dummy data with a set of generated comparative values from CS. Since the participant's entity data does not leave the local, the generated matrix is derived from random numbers. Hence, the simulated  $l$ -dimensional matrix is indistinguishable from the one that is using true data for comparison.  $\square$

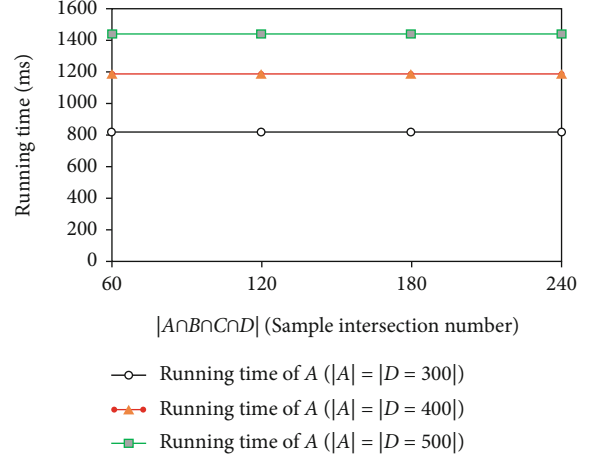


FIGURE 8: Running time of A with different amounts of intersection.

Case 2 (including CS in  $W$ ). Namely, consider the corrupted CS and  $U_i$ .

*Proof.* For the corrupted CS and  $\mathcal{S}_u$ , denote the views of CS and participants  $U_i$  as  $V_{CS} = \text{view}_{CS}$  and  $V_u = \{\text{view}_{u_1}, \text{view}_{u_2}, \dots, \text{view}_{u_n}\}$ . Based on the process of the MEMP, we can derive  $\text{view}_{CS} = \{v_{ik}, c_{ik}, d_{ij}\}$  and  $\text{view}_{u_i} = \{v_{ik}, c_{ik}, g_{ik}\}$ , where  $i \in [1, n]$ ,  $j \in [1, l]$  and  $ik$  refers to the  $k$ th entity's identity of the  $i$ th participants. It can be found that the elements belonging to  $\text{view}_{CS}$  and  $\text{view}_{u_i}$  can be treated as random values. Therefore, we can infer that  $\text{view}_{CS}$  and  $\text{view}_{u_i}$  are simulatable for  $\mathcal{S}_{CS}$  and  $\mathcal{S}_u$ , and the simulated views cannot be distinguished computationally by the adversary.

The proof of the second part about  $\mathcal{F}_{em2}$  is similar to Theorem 2, so we will not go into details.

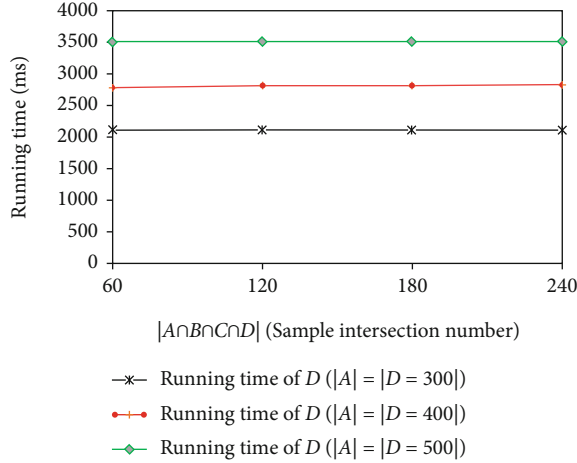
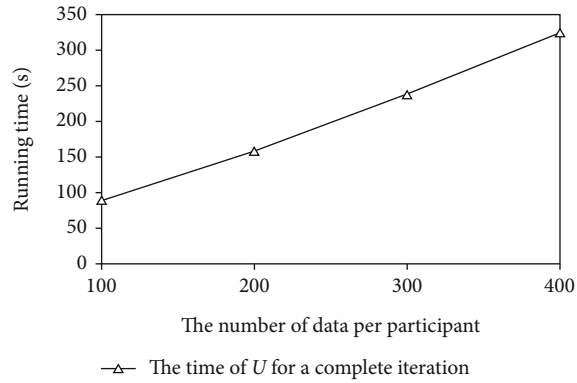
5.3. *Security of VFLR.* In the model training, CS needs to get messages sent by at least  $t$  participants to decrypt the correct value so that the data for each participant cannot be retrieved and messages retrieved by CS can only be aggregated values without revealing anything else due to the combination of homomorphic encryption and threshold methods. The following theorem will be proved to show the security of the VFLR model.

**Theorem 4.** *For secure VFLR model, there exists a PPT simulator  $\mathcal{S}_u$  or  $\mathcal{S}_{cs}$  that can simulate the ideal view  $\text{IDEAL}[\mathcal{L}, \mathcal{S}, \mathcal{F}_{ml}]$  which is computationally indistinguishable from the real view of  $\mathcal{A}_u$  or  $\mathcal{A}_{cs}$ .*

Our VFLR calls the previous aggregation framework, and its security is based on the proof of Theorem 2. Since Theorem 2 has been proved, here we only do a simple description for VFLR.

*Proof.* Similar to the proof of Theorem 2, for a group of corrupted participants,  $\mathcal{S}_u$  can run them using their local data while for honest participants  $\mathcal{S}_u$  simulate them with dummy data. Therefore,  $\mathcal{S}_u$  run  $\mathcal{F}_{ml}$  to generate random values to



FIGURE 9: Running time of  $D$  with different amounts of intersection.FIGURE 10: Running time of  $U_i$  with different amounts of data.

replace the masked  $\Theta_i X_k^i$  as the inputs of honest participants. In the model building, what the honest-but-curious participants get is the errors rather than some information that reflects real data and they cannot identify whether the aggregated values used to calculate the errors are based on real data. Therefore, the view of  $\mathcal{S}_u$  is indistinguishable from a real one.  $\square$

Considering the corrupted CS,  $\mathcal{S}_{CS}$  run  $\mathcal{F}_{ml}$  to generate dummy labels, with which  $\mathcal{F}_{ml}$  computes the errors after obtaining the aggregation. Because local data are within honest participants, the outputs cannot reveal specific information. Therefore, there exist a PPT simulator  $\mathcal{S}_{cs}$  that can simulate the ideal view which is computationally indistinguishable from the real view of  $\mathcal{A}_{cs}$ .

## 6. Performance Evaluation

In this section, the effectiveness and efficiency of the experiment are presented.

**6.1. Experiment Configuration.** Four clients  $A, B, C,$  and  $D$  are built to simulate the feasibility and performance, in which  $A, B,$  and  $C$  refer to the average participants and  $D$  means the aggregator with sample labels. We carry out our experiments

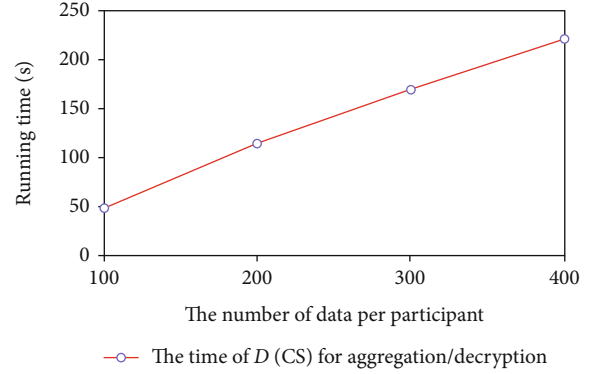


FIGURE 11: Running time of CS with different amounts of data.

TABLE 4: Comparison of functionality.

Scheme	GAM	M-p	EM	WP
Cheng et al.'s scheme [30]	×	√	×	×
Fu et al.'s scheme [31]	√	√	×	×
Our scheme	√	√	√	√

on the device with CPU i7-6700, 3.2 GHz, and Memory 24 G. The programs in the experiments are implemented in Python, and the length of  $p$  and  $q$  are set to 512 bits for RSA and Paillier. We adopt a finite field  $\mathbb{Z}_p^*$  with  $P = 2^{11}$  and the standard Shamir's  $(t - n)$  secret sharing to generate the shares of secret.

In the MEMP, we generated different random values between 5225280000000000000 and 522528000000000000550 as identification of user samples for  $A, B, C,$  and  $D,$  respectively, satisfying that the number of intersection of  $A, B, C,$  and  $D$  is 60, 120, 180, and 240. In VFLR, we selected 300 digits from the handwritten digits dataset (handwritten digit dataset: [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\\_digits.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html)), which contains 64 features.  $A, B,$  and  $C,$  respectively, hold 20 features in these samples, while  $D$  contains 4 features and 1 label. Since the dataset is multiclassified, we set 0 to 4 as category 1 and 5 to 9 as category 2.

**6.2. Performance Analysis of the MEMP.** For the MEMP, in the simulations, we get the execution time of  $U_i$  ( $A, B, C$ ) and the CS ( $D$ ) when increasing the amount of data from 100 to 500. Note that the time to initialize the system ignored in all experiments and  $|A|$  represents the size of samples in  $A,$  the same thing for  $B, C, D.$   $U_i$  ( $A, B, C$ ) first calculates the median value for comparison through CS, the computation of which is related to the amount of their own data, and then the size of the calculated comparison matrix is related to the amount of CS's data. When the number of samples' identification of  $A, B,$  and  $C$  increases, the increase of computation time of  $A, B,$  or  $C$  is mainly reflected in generating sample identification library, and the most time-consuming in  $D$  is the blind signature. Figure 5 shows the calculation time of  $A$  and  $D$  when the number of samples in  $D$  is 100 and the samples in  $U_i$  ( $A, B, C$ ) fluctuate from 100 to 500. Figures 6 and 7 show the changing trend of running time of  $A$  and  $D$

TABLE 5: Comparison of computation and communication.

Scheme	Participant	Aggregator	Comm-rounds	Mask number
Xu et al.'s scheme [25]	1R + 2SC	$t$ SC	2	$n \cdot t + 1$
Fu et al.'s scheme [31]	2SC + 2CR	1CR	1	$n$
Our scheme	1R + 1HE + 1P	1HE + 1P	1	$n$

with the increase of  $|D|$  when the amounts of  $A$ ,  $B$ , and  $C$  are all 100. As the sample amount of  $D$  is increasing, the encryption time is proportional to the data volume of  $D$ . Because the time of aggregation and decryption is related to its own data volume, its identification volume affects the size of an identification matrix, thereby affecting the aggregation volume, resulting in a linear increase in the time of aggregation and decryption. However, changes in the intersection of  $A$ ,  $B$ ,  $C$ , and  $D$  will not affect the respective calculation time that is only related to the amount of data of all parties, as is shown in Figures 8 and 9.

**6.3. Performance Analysis of the VFLR.** In the VFLR, there is no approximation algorithm applied here so that the updated parameters in our VFLR are exactly the same as those in the traditional logistic regression. Therefore, the training accuracy is also consistent. The main observation here is the running time of the VFLR. The extra cost in training is mainly from power exponent and homomorphism. In the training phase,  $t_e$  represents the time of a complete encryption and  $t_s$  represents the time of an aggregation and decryption for  $n$  participants. When the sample size is  $m$ , complexity time of encryption and aggregation, respectively, is  $o(t_e \cdot m)$  and  $o(t_s \cdot m)$ . We selected 400 samples and set the number of features at each end to 20. When the number of data amount is 100, 200, 300, and 400, respectively, the time for a single end to complete an iteration and the time for  $D$  to complete aggregation and decryption are captured. (See Figures 10 and 11.) As the figures show, we gradually adjust the data volume from 100 to 400, and the running time increases approximately linearly.

**6.4. Communication Overhead.** Since the establishment of the user identification library and the generation of the comparison matrix can be done offline in the MEMP, we discuss the communication overhead of the MEMP from the perspective of the proposed aggregation model. The same is true for the VFLR, because its execution process is completely consistent with the GAM. Let the length of the ciphertext of each participant be  $cl$  and the length of the  $H()$  and error, respectively, be  $|H|$  and  $|\text{error}|$  where  $cl > |H|$  and  $cl > |\text{error}|$ . In the MEMP, it only takes one turn to complete the comparison and identify the common entity. If CS sends  $l$  entity mask, the communication load for each participant is  $l \cdot cl$ . In order for participants to complete the comparison, the CS needs to send corresponding entity mask for  $n$  participants so that the communication load of CS is  $|H| \cdot l \cdot n + n \cdot |\text{ID}|$ . In the VFLR, we just consider the communication load for one iteration. Each participant sends ciphertext of size  $cl \cdot m$ . The CS just needs to return errors of  $m$  samples, so the communication load of CS is  $m \cdot |\text{error}|$ . In this way, the total space com-

plexity of MEMP and VFLR can be expressed as  $o(n \cdot l \cdot cl)$  and  $o(n \cdot m \cdot cl)$ . We can see that as the number of participants or samples increases, the total communication overhead also grows linearly.

## 7. Related Work

Many privacy-preserving models for specific machine learning algorithms have emerged. There mainly exist two kinds of technologies adopted in the privacy-preserving training, i.e., differential privacy [21] and cryptography-based approaches. Differential privacy applied to FL can prevent clients from trying to reconstruct the private data of other clients by exploiting the global model, as done in [13, 22]. It adds noise to the original dataset or gradient parameters while ensuring the availability of the data. But it brings low accuracy. The cryptographic technologies can provide privacy protection while ensuring accuracy. Secure multiparty computation, secret sharing, and homomorphic encryption are the common methods. For example, Aono et al. [23] used homomorphic encryption to improve the logistic regression algorithm ensuring the security of the training and predicting data. Liu et al. [24] propose a secret sharing-based federated extreme boosting learning framework to achieve privacy-preserving model training for mobile crowdsensing. Xu et al. [25] proposed a privacy-preserving and verifiable federated learning framework based on homomorphic hash functions, in which clients can verify whether the result returned by cloud server is correct.

Some previous works with privacy preserving over vertical data partition are discussed in [26, 27]. However, there exist potential privacy risks as a result of revealing class distribution over the given attributes. Research on VFL is first proposed in [28], where a federated logistic regression scheme is designed through an additively homomorphic scheme. Nock et al. [29] then provide a formal assessment of how errors in entity resolution impact learning. Cheng et al. [30] propose a novel lossless privacy-preserving tree-boosting system, which conducts a learning process on multiple parties with partially common user samples but different feature sets. Fu et al. [31] combines Lagrange interpolation and Chinese remainder theorem to realize the secure aggregation of gradients. But some works assume sample entities already being matched or they only deal with two VFL participants. The proposed framework is more advantages than those approaches as it can support multiparty VFL by taking into account entity matching and model training with withdrawal of participants. Table 4 shows the functional comparison between our framework and existing two main works from GAM, multiparticipants (M-p), entity matching (EM), withdrawal of participants

(WP). Table 5 shows the comparison of computation and communication mainly for secure aggregation from the participant and aggregator's main function operation, communication rounds (Comm-rounds), and mask number (i.e., the number of message received by the aggregator), where SC represents the times of secret reconstruction, CR represents the times of calculation of Chinese residual theorem, HE represents the times of homomorphic encryption,  $R$  represents the times of pseudorandom generator, and  $P$  represents the times of calculation of the power exponent. It can be seen from Table 5 that our scheme has advantages over reference [25] in terms of calculation and communication overhead. Compared with [25, 31], although our scheme applies the principle of secret sharing, it does not need to spend the overhead of secret reconstruction.

## 8. Conclusion

For privacy protection of data aggregation and joint training in federated learning, as well as entity matching, we designed a PFLF where we proposed a general aggregation model and designed a multiparty entity matching protocol, which can find the common entity of multiple participants without data disclosure. In addition, our GAM was used to improve logistic regression algorithm to ensure the confidentiality of data samples during training and support the withdrawal of participants over VFL. The security analysis of the scheme was given based on the simulator, and the performance of the system was tested with the experimental data. The next research will focus on optimizing the operating load of the system and considering cases where the participants are malicious to construct a verifiable federated learning framework and design incentives to facilitate federated learning.

## Data Availability

The data used to support the finding of this study are included in the article.

## Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61662009 and 61772008), Science and Technology Major Support Program of Guizhou Province (Grant No. 20183001), Key Program of the National Natural Science Union Foundation of China (Grant No. U1836205), Science and Technology Program of Guizhou Province (Grant No. [2019]1098), Project of Innovative Group in Guizhou Education Department (Grant No. [2013]09), Project of High-level Innovative Talents of Guizhou Province (Grant No. [2020]6008), and Science and Technology Program of Guiyang (Grant No. [2021]1-5).

## References

- [1] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [2] P. Kairouz, H. B. McMahan, B. Avent et al., "Advances and open problems in federated learning," 2019, <https://arxiv.org/abs/1912.04977>.
- [3] Y. Tian, Z. Wang, J. Xiong, and J. Ma, "A blockchain-based secure key management scheme with trustworthiness in dwsns," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6193–6202, 2020.
- [4] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR)," in *A Practical Guide*, Springer International Publishing, Cham, 1st edition, 2017.
- [5] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtarik, "Federated optimization: distributed machine learning for on-device intelligence," 2016, <https://arxiv.org/abs/1610.02527>.
- [6] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, *Federated Learning of Deep Networks Using Model Averaging*, 2016.
- [7] K. Bonawitz, V. Ivanov, B. Kreuter et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, Dallas, TX, USA, October 2017.
- [8] R. Xiong, R. Ma, L. Chen et al., "A personalized privacy protection framework for mobile crowdsensing in iiot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4231–4241, 2019.
- [9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
- [10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.
- [11] Y. Liu, Z. Ma, X. Liu, Z. Wang, S. Ma, and K. Ren, "Revocable federated learning: a benchmark of federated forest," 2019, <https://arxiv.org/abs/1911.03242>.
- [12] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.
- [13] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," 2017, <https://arxiv.org/abs/1710.06963>.
- [14] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.
- [15] G. Liang and S. S. Chawathe, "Privacy-preserving inter-database operations," in *International Conference on Intelligence and Security Informatics*, pp. 66–82, Springer, 2004.
- [16] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," *Advances in neural information processing systems*, pp. 289–296, 2009.
- [17] C. Jost, H. Lam, A. Maximov, and B. J. Smeets, "Encryption performance improvements of the paillier cryptosystem," *IACR Cryptology ePrint Archive*, vol. 2015, article 864, 2015.

- [18] D. Bogdanov, *Foundations and Properties of Shamirs Secret Sharing Scheme Research Seminar in Cryptography*, vol. 1, University of Tartu, Institute of Computer Science, 2007.
- [19] P. Mohassel and Y. Zhang, "Secureml: a system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, San Jose, CA, USA, May 2017.
- [20] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *Theory of Cryptography Conference*, pp. 61–85, Springer, 2007.
- [21] A. Blum, C. Dwork, F. McSherry, and K. Nissim, "Practical privacy: the sulq framework," in *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '05*, pp. 128–138, Baltimore, Maryland, USA, 2005.
- [22] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," *International Conference on Artificial Intelligence and Statistics*, pp. 473–481, 2018.
- [23] Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang, "Scalable and secure logistic regression via homomorphic encryption," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 142–144, New Orleans, LA, USA, March 2016.
- [24] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, and R. Deng, "Boosting privately: privacy-preserving federated extreme boosting for mobile crowdsensing," 2019, <https://arxiv.org/abs/1907.10218>.
- [25] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifyfnet: secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.
- [26] J. Vaidya and C. Clifton, "Privacy-preserving decision trees over vertically partitioned data," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 139–152, Springer, 2005.
- [27] J. Vaidya, "A survey of privacy-preserving methods across vertically partitioned data," in *Privacy-preserving data mining*, pp. 337–358, Springer, 2008.
- [28] S. Hardy, W. Henecka, H. Ivey-Law et al., "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, <https://arxiv.org/abs/1711.10677>.
- [29] R. Nock, S. Hardy, W. Henecka et al., "Entity resolution and federated learning get a federated resolution," 2018, <https://arxiv.org/abs/1803.04035>.
- [30] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, and Q. Yang, "Securboost: a lossless federated learning framework," 2019, <https://arxiv.org/abs/1901.08755>.
- [31] A. Fu, X. Zhang, N. Xiong, Y. Gao, H. Wang, and J. Zhang, "VFL: a verifiable federated learning with privacy-preserving for big data in industrial IoT," *IEEE Transactions on Industrial Informatics*, p. 1, 2020.