Research on Financial Data Fraud Detection Based on Federated Learning

Professional Category： Electronic Information

Research Direction： Computer Technology

Author's Name： Guo Zhengxin

Supervisor： Chen Shizhan Professor

Enterprise Mentor： Ma Kai Senior Engineer

Defense Date	2024.05.29		
Defense	Name	Title	Affiliation
Chairperson	Zhou Zhihai	Researcher	National Ocean Technology Center
Committee Member	Xue Xiao	Professor	Tianjin University
	Wang Bo	Associate Professor	Tianjin University

Tianjin University, College of Intelligence and Computing

June 2024

ABSTRACT

Financial institutions need to pay great attention to the security and privacy of financial data, which is an important task that cannot be ignored. The advent of federated learning provides an important approach to protect data security and privacy. However, despite the significant advantages of federated learning, there are two key challenges to fraud detection in financial institutions. The first is the data security threats, which directly affect the accuracy of fraud detection and may lead to data leakage on the client. The second is the communication costs problem of federated learning, where frequent model exchanges put a huge burden on the communication system, which may lead to performance bottlenecks and increased latency. To address the above challenges, this thesis launches a research in federated learning, which involves two research components as follows:

(1) To improve the accuracy of fraud detection and protect the data security and privacy of participants, this thesis proposes a federated learning fraud detection method that incorporates differential privacy preservation. First, based on the IDW-SMOTE algorithm, the feature extraction of the stacked autoencoder and the classification model of the multilayer perceptron are fused to construct an efficient fraud detection model. Then combine the differential privacy algorithm and blockchain to realize the encryption and traceability of local models, and improve the data security and privacy in the process of federated learning training and transmission. Experimental results show that the method of this thesis achieves superior results on all three datasets, and the introduced encryption algorithm effectively protects the security and privacy of the model data.

(2) To reduce the communication costs and save the network bandwidth in federated learning, this thesis improves the communication efficiency from the perspective of compressing the model and optimizing the aggregation time series. First, the knowledge distillation algorithm and multilayer perceptron are fused to realize the compression of the local fraud detection model to reduce the amount of data transmitted by communication. Second, based on mathematical derivation, the correlation between the loss function and the aggregate time series is revealed, so as to construct a better series to reduce the communication rounds between

the client and the server. Experimental results show that the knowledge distillation algorithm based on multilayer perceptron can effectively compress the complex fraud detection model in terms of reducing communication costs. The non-periodic aggregation method further reduces the number of communication rounds, and successfully reduces the number of communication rounds by 10 and 20 respectively compared to the FedProx and FedAvg aggregation algorithms.

In addition, to solve the problem of credit card fraud detection, this thesis deploys the proposed fraud detection method to the actual production environment to further verify the effectiveness of the method.

KEY WORDS: Federated Learning, Blockchain, Fraud Detection, Financial Data Security, Privacy Protection, Communication Costs

CONTENTS

1 Introduction

1.1 Background

As significant entities entrusted with vast amounts of sensitive financial data, financial institutions must place a high priority on and comprehensively safeguard data security and privacy to ensure the integrity of their core business operations and maintain customer trust. However, financial institutions frequently confront the threat of financial fraud, which poses serious risks and negative impacts on their business operations and reputation. Consequently, financial institutions must implement robust measures to counter financial fraud, secure financial data, and protect the stability and sustainable development of the financial system.

Financial fraud encompasses various forms such as credit card fraud, insurance fraud, money laundering, and medical fraud, among others[1–3]. In recent years, the occurrence rate of financial fraud has been steadily increasing, and it has become a severe global issue. The total annual losses due to financial fraud amount to tens of billions of dollars worldwide[4,5]. A 2022 report from the United States revealed that 62% of surveyed financial institutions had experienced financial fraud within the past year, with an average loss of 102 million per institution due to fraud[6]. The financial losses resulting from fraud are typically borne by companies and merchants. For instance, in credit card fraud, merchants ultimately have to compensate for the fraudulent amount and assume the responsibility for management, while also losing consumer trust[7].

To prevent and mitigate financial fraud, financial institutions typically employ a range of security measures and precautions. Traditional financial fraud detection methods often rely on rules and static thresholds, such as establishing financial fraud enforcement networks or designing intuitive financial fraud classifiers[8], to collect extensive information for combating financial fraud and protecting the interests of financial institutions. However, these traditional detection methods are unable to keep up with the ever-evolving financial fraud tactics. With the advancement of artificial intelligence, the financial sector has seen the emergence of fraud detection techniques based on machine learning[3]. This technology can comprehensively and efficiently analyze vast amounts of financial data, effectively detecting financial fraud. Consequently, 86.3% of financial institutions consider machine learning or artificial intelligence as a significant focus for innovation or improvement[6].

Financial fraud detection is generally considered a type of anomaly detection task, with the goal of identifying fraudulent data within financial institutions. To address various types of financial fraud problems, researchers have explored multiple solutions using machine learning, including Multilayer Perceptrons (MLPs)[9–11], Convolutional Neural Networks (CNNs)[12], and Generative Adversarial Networks (GANs)[13,14], among others. Simultaneously, to ensure data security and privacy protection for financial institutions while addressing the growing demands for data processing, federated learning has garnered increasing attention and become a focus for financial institutions.

Federated learning is a decentralized, distributed machine learning method that not only protects data privacy but also enables data sharing[15]. In the financial domain, federated learning has been widely applied to collaborative fraud detection among multiple institutions, ensuring data security and privacy. However, there are still challenges when jointly building financial fraud detection models:

- Data security challenges in fraud detection: In financial fraud detection tasks, the proportion of fraudulent samples to normal samples is often highly imbalanced, resulting in a serious data imbalance problem. This imbalance may cause the model to ignore or underestimate minority class user information and fraudulent behavior during the learning process, which not only affects the overall detection accuracy but may also pose data security risks. Furthermore, during the model exchange process in federated learning, there may be challenges such as model theft and inference of original dataset features. These issues pose a significant threat to the security and privacy of client-side data, necessitating the implementation of effective security protection mechanisms to address them.
- Communication costs challenges in federated learning: In practical applications, particularly those involving complex global model structures and a large number of participants, frequent model exchanges between the server and client sides can lead to a significant increase in federated learning communication data volume. This, in turn, can cause network bandwidth limitations to become the primary communication bottleneck. Therefore, reducing communication costs and overcoming communication bottlenecks remain important challenges that urgently need to be addressed in federated learning.

Addressing the aforementioned challenges holds significant importance for enhancing the accuracy and operational efficiency of fraud detection in financial institutions, as well as ensuring the security and privacy of financial data. By tackling data imbalance issues and safeguarding data security during model transmission, not only can federated learning more accurately identify fraudulent behavior and improve the accuracy of financial institutions in fraud detection, but it can also reduce the risk of data leaks, thereby protecting the security and privacy of financial data. Furthermore, reducing communication costs can conserve valuable network bandwidth resources, prevent communication bottlenecks, and lower operational costs for financial institutions. This enables financial institutions to adapt to larger-scale, more complex data training, thereby improving their efficiency and competitiveness.

1.2 Objectives and Main Content

This thesis proposes a federated learning fraud detection method with integrated differential privacy protection, aiming to achieve efficient fraud detection by collaborating among multiple financial institutions while ensuring the security and privacy of the participating parties' data. To maintain network bandwidth and reduce training costs, it is also necessary to optimize communication costs from the perspectives of model compression and optimizing aggregation time series. Figure 1-1 illustrates the main content of this research and the relationships among the involved algorithms.

(1) Federated Learning Fraud Detection with Differential Privacy

Design a federated learning detection architecture with integrated differential privacy protection to improve detection accuracy while safeguarding the security and privacy of participating parties' data. First, propose and design the IDW-SMOTE algorithm for data preprocessing to address data distribution imbalance. The preprocessed data serve as input for training the fraud detection model. Then, integrate the feature extraction of stacked autoencoders with the classification model of multilayer perceptrons to build an effective fraud detection model that enhances detection accuracy. Finally, combine differential privacy algorithms with off-chain storage mode of blockchain to encrypt and trace local models, protecting data security and privacy during federated learning training and transmission, thus providing a secure environment for the fraud detection model. In terms of experiments, verification tests are designed for the European credit card dataset, user dataset, and bank user dataset. The efficiency of the IDW-SMOTE algorithm and the multilayer perceptron model based on stacked autoencoders are compared, and the effectiveness of federated learning and differential privacy algorithms is validated. Additionally, the traceability of the blockchain is evaluated.

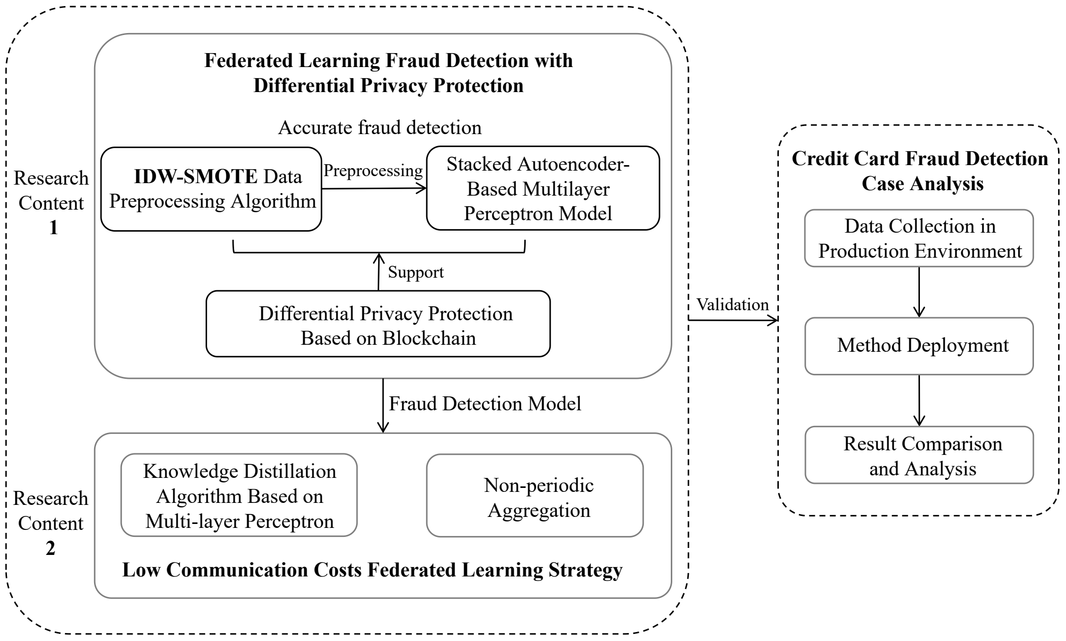

Federated Learning Fraud Detection with Differential Privacy Protection
Accurate fraud detection
Research Content 1
IDW-SMOTE Data Preprocessing Algorithm
Preprocessing
Stacked Autoencoder-Based Multilayer Perceptron Model
Support
Differential Privacy Protection Based on Blockchain
Fraud Detection Model
Validation
Research Content 2
Knowledge Distillation Algorithm Based on Multi-layer Perceptron
Non-periodic Aggregation
Low Communication Costs Federated Learning Strategy
Credit Card Fraud Detection Case Analysis
Data Collection in Production Environment
Method Deployment
Result Comparison and Analysis

Figure 1-1　Research Objectives and Main Content

(2) Communication Costs Optimization Strategy

A strategy for reducing communication costs in federated learning is proposed, optimizing communication costs from two perspectives: compressing model size and building an optimized aggregation time series. First, knowledge distillation algorithms are integrated with multi-layer perceptron models to achieve local model compression and reduce the amount of data transferred, thereby lowering communication costs. Second, based on mathematical derivation, the correlation between the loss function and the aggregation time series is revealed, and the number of data interactions between the client and server is reduced to further decrease the communication costs in federated learning. Experiments are designed to verify the effectiveness of the strategy using European credit card data, user data, and bank user data. The results demonstrate the effectiveness of the knowledge distillation algorithm based on multi-layer perceptron models and validate the efficiency of the non-periodic aggregation algorithm."

In addition, this thesis will apply the proposed fraud detection method to a real-world production environment for detecting fraudulent bank credit card data. The process involves collecting real data from the production environment, deploying the proposed method, and then comparing and analyzing its performance with existing fraud detection methods. This comprehensive evaluation of the proposed method's performance will provide a powerful reference for further research and application.

1.3 Thesis Organization

This thesis is organized into six chapters, as follows:

Chapter 1 serves as the introduction, first presenting the background and significance, then outlining the objectives and main content, and finally summarizing the overall structure of the thesis.

Chapter 2 discusses the risk challenges faced by the financial sector in terms of data security and privacy protection, as well as corresponding solutions. It also explores the current state of research on federated learning and blockchain technology in the financial domain.

Chapter 3 proposes a federated learning fraud detection architecture for financial data, incorporating differential privacy protection. This chapter begins by designing a data preprocessing algorithm to address data distribution imbalance. Then, stacked autoencoders and multilayer perceptrons are combined to improve the accuracy of the fraud detection model. Subsequently, blockchain, differential privacy algorithms, and federated learning are integrated to protect data security and privacy. Finally, a series of comparative experiments are conducted on multiple datasets, and the results are analyzed.

Chapter 4 presents an approach to reduce communication costs in federated learning. This chapter first combines knowledge distillation algorithms with multilayer perceptrons to compress local models and lower communication costs. Then, the relationship between the loss function and the aggregation time series is derived to reduce communication costs by decreasing the number of communication rounds. Finally, various algorithms are compared on multiple datasets, and the experimental results are analyzed.

Chapter 5 applies the proposed fraud detection method to a production environment and compares its performance with that of actual fraud detection methods.

Chapter 6 concludes the thesis by summarizing the research findings and outlining directions for future research.

2 Relevant Literature and Related Theories

To ensure the security of a collaborative fraud detection system built by multiple financial institutions, it is necessary to have a thorough understanding of data security and privacy protection in finance, federated learning technology, and blockchain technology. This chapter first summarizes the risks and challenges faced by data security and privacy protection in the financial domain and their solutions. Then, it focuses on introducing federated learning and its attack and defense strategies. Finally, it outlines blockchain and its application in data security and privacy protection.

2.1 Data Security and Privacy Protection in Finance

2.1.1 Risks and Challenges

As the economy becomes increasingly digital, financial institutions inevitably face risks and challenges related to data security and privacy protection. This is mainly due to two reasons. First, the vast amount of sensitive data held by financial institutions holds immense value for criminals, who can exploit this data for fraud, identity theft, and other malicious activities. Second, the digital transformation of financial institutions requires a high degree of data digitization, resulting in more data being electronized and stored in the cloud, thereby intensifying the risks of data security threats and data privacy leaks. This section will discuss the more prominent risks and challenges currently facing financial institutions in detail.

(1) Data Collaboration and Sharing

Financial institutions consist of various companies of different sizes and types, including deposit institutions, investment companies, insurance companies, credit and financing organizations, and providers of related financial utilities and services. These institutions need to share data in their daily business to achieve more efficient business processes and provide better services. However, financial institutions face certain risks and challenges during data sharing.

First, due to concerns about data security and privacy, each department is reluctant to share its data. Moreover, different departments have varying requirements for data security and privacy, which adds complexity to data sharing[16].

Second, financial institutions are worried that sharing information with competitors may weaken their competitive advantage[17]. Shared data may reveal trade secrets, affecting their market position and profitability.

Furthermore, regulatory authorities may focus on network security cooperation when taking antitrust actions, which also hinders information sharing among financial institutions. Companies are concerned that overly close cooperation may lead to unfair treatment or restrictions by regulatory authorities, affecting their business freedom and competitiveness.

Therefore, financial institutions face risks and challenges in data sharing, including concerns about data security and privacy, weakened competitive advantage, and uncertainty brought by regulatory authorities.

(2) Trust Issues

Financial institutions face multifaceted trust risks.

First, financial institutions often rely on services and technology provided by third-party vendors, which can pose certain risks, especially in terms of data security. Even if the internal security of financial institutions is strong, third-party service providers may be subject to network attacks and cause data breaches. Attackers can send malicious software to providers through legitimate download and upgrade processes, thereby gaining backdoor access to the network. The SolarWinds[18] vulnerability is a typical attack case that affected thousands of financial and government entities.

Second, financial institutions also face internal trust risks. Inappropriate behavior by employees, internal data leaks, and abuse can pose significant threats to the reputation and security of financial institutions[19].

The impact of trust risks on financial institutions is significant. The breakdown of trust can lead to customer loss, legal litigation, regulatory penalties, and reputational damage.

(3) Data Tampering Issues

Data tampering can come from both external attackers and internal employees.

External attacker attacks: Hackers and malicious attackers seek vulnerabilities in financial institutions and inject malicious code into servers[20], which may lead to unauthorized access and data tampering. Especially for unencrypted data, fraudsters or hackers can easily tamper with the data, causing serious problems for financial institutions.

Internal employee tampering: Data manipulation attacks can also come from internal employees. After obtaining access to trusted systems, attackers can make changes to data without being detected to gain personal benefits[21]. For example, an employee may alter customer-related information, which appears legitimate and compliant, making it difficult to detect. The resulting incorrect data will be stored and may cause greater damage.

Data tampering risks have a significant impact on the integrity of financial data and threaten data security and privacy.

(4) Compliance Requirements

Financial institutions face compliance risks that include complex laws and regulations, cost and time pressures, penalties and fines, and restrictions from regulatory bodies. Financial institutions must comply with various data protection laws and privacy regulations, such as the European General Data Protection Regulation, to ensure that customers' personal data is adequately protected. Violating compliance requirements can result in severe penalties and fines[19], and even the risk of class action lawsuits. Furthermore, restrictions from regulatory bodies also pose challenges to financial institutions.

To address these challenges, financial institutions urgently need to comply with relevant laws and regulations, adopt appropriate security measures, build a secure and stable business environment, implement encryption of sensitive data, strengthen identity authentication and access control, to protect customers' sensitive information and ensure the legitimate rights and interests of financial institutions.

2.1.2 Federated Learning and Blockchain-based Protection

Federated learning and blockchain are technologies that involve distributed computing and data security. An increasing number of researchers are applying them to the financial sector to protect the security and privacy of financial data.

The emergence of federated learning has effectively balanced the contradiction between data sharing, data security, and privacy, making it possible to achieve data sharing while protecting data privacy. Applying federated learning in finance has the following prominent advantages: First, federated learning can promote data sharing and improve the efficiency of data exchange. Lu et al.[22] proposed a federated learning-based data sharing architecture that effectively protects data privacy and makes data exchange more efficient and secure. Second, federated learning can resist some attacks, enhance data security, and prevent data tampering. Li et al.[23] proposed a federated learning framework based on blockchain for data sharing and reducing malicious attacks. Furthermore, federated learning can avoid compliance risks to a certain extent. By training locally to avoid direct data exchange, federated learning can meet legal requirements for data sharing and privacy protection. However, additional measures and mechanisms are still needed for other compliance requirements, such as compliance monitoring and anti-money laundering.

Blockchain is a decentralized distributed ledger technology that can effectively protect the security and privacy of financial data and address the following risks and challenges: First, blockchain provides a trust mechanism that effectively solves trust issues in financial institutions. Bandara et al.[24] proposed a blockchain-based federated learning system called Bassa-ML that provides higher transparency and trust for models. Second, blockchain can resist certain attacks and prevent data tampering. Shayan et al.[25] proposed the Biscotti method, which uses blockchain and cryptographic primitives to protect the machine learning process and has the ability to resist known attacks. Additionally, blockchain provides an incentive mechanism. Furthermore, blockchain can also solve some regulatory problems[26].

Table 2-1 Financial Institutions' Issues, Risks & Challenges, and Solutions.

Issues	Risks & Challenges	Solutions
(1)Data Collaboration and Sharing	Data Security & Privacy[16]	Federated Learning[22]
	Competitive Advantage Erosion[17]	
	Regulatory Uncertainty	
(2)Trust Issues	Third-party Security[18]	Blockchain[24]
	Internal Data Leaks[19]	
(3)Data Tampering Issues	External Attacker Attacks[20]	Federated Learning[23]
	Insider Data Tampering[21]	Blockchain[25]
(4)Compliance Requirements	Complex Regulations & Rules	Federated Learning
	Cost & Time Pressure.	Blockchain[26]
	Penalties & Fines[19]	
	Regulatory Restrictions	

Table 2-1 summarizes the risks of data security and privacy protection in the financial domain and the federated learning and blockchain solutions proposed to address these risks. Federated learning and blockchain have been widely applied in finance, effectively maintaining the security and privacy of financial data.

2.2 Security Defense Research in Federated Learning

2.2.1 Federated Learning

With the development and application of artificial intelligence, machine learning faces two major challenges. First, a single machine can no longer meet the needs of large-scale machine learning training with massive data and complex computations. Second, with the increasing awareness of user privacy and the enactment of relevant laws and regulations, the urgent need for data sharing and the limitation of data silos are becoming increasingly conflicting. Achieving the collaborative construction of complex and powerful models by multiple institutions while meeting data privacy and security regulatory requirements has become an urgent problem to be solved. To address these two challenges, federated learning has emerged. Federated learning is a distributed learning framework[27] that combines privacy protection and secure encryption technologies to protect the data security and privacy of each participant while jointly achieving machine learning training. In 2017, Google researchers Brendan et al.[28] first proposed the federated learning framework, which adopts a server-client model. The emergence of federated learning has alleviated the dilemma faced by big data development and has become a new paradigm for artificial intelligence, with widespread development and application in finance, medicine, industry, and other fields.

Federated learning consists of N participants, each of whom retains and trains their own dataset locally with the goal of building a complete machine learning model using these datasets. The general process of federated learning is as follows:

(1) Distribute initial global model: The central server sends the global model to each participating client.

(2) Train local model: After receiving the global model, each client uses it to train their local dataset.

(3) Upload local model: The client uploads the trained local model to the central server and waits for the next training round.

(4) Aggregate new global model: The central server receives and aggregates the local models to obtain a new global model, which is then sent to the clients for the next round of training.

2.2.1.1 Classification of Federated Learning Frameworks

There are two types of federated learning frameworks[29]: centralized federated learning architecture, i.e., server/client mode, and end-to-end federated learning architecture, i.e., peer-to-peer computing. Centralized federated learning architecture is suitable for scenarios where multiple users collaborate for training. As shown in Figure 2-1, the central server in federated learning is responsible for coordinating and managing each client and allocating the global model. The datasets of the clients complement each other, and each client completes machine learning training locally. Additionally, the centralized federated learning architecture typically requires the number of clients to be greater than 1. This is because when there is only one client, the model cannot be updated and aggregated, and the advantages of federated learning cannot be realized.

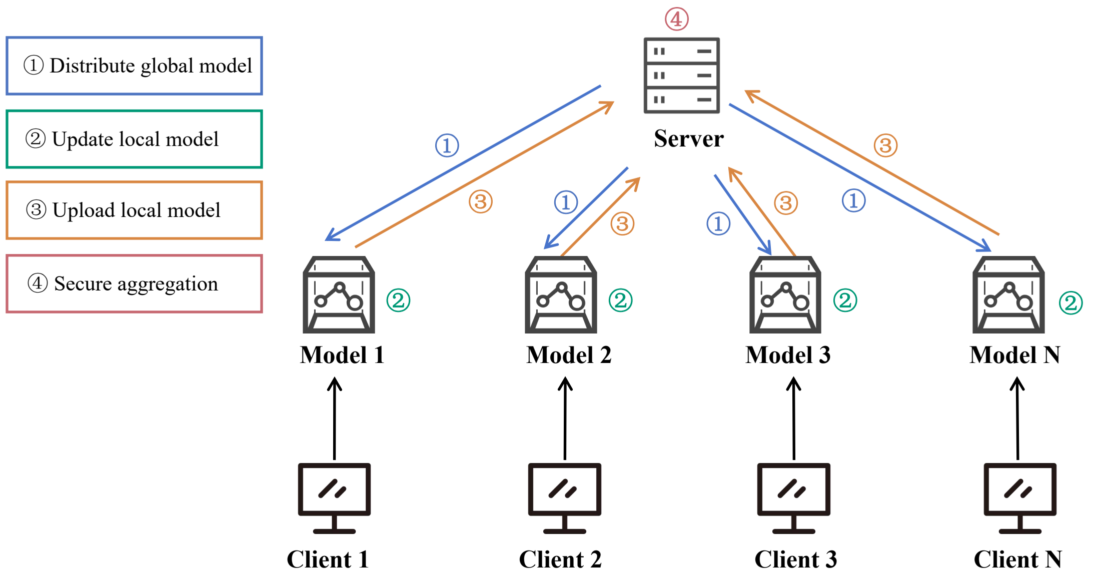


Figure 2-1 Centralized Federated Learning Architecture[29]

When multiple institutions collaborate on training but cannot select a suitable coordinator, an end-to-end federated learning architecture is usually chosen. As shown in Figure 2-2, in end-to-end federated learning, a client with a larger weight is used to replace the central server, and all model transfers occur between clients. After completing machine learning training locally, the client needs to encrypt and transmit the obtained model to the remaining clients to ensure data security and privacy. Therefore, this architecture requires more encryption and decryption operations.

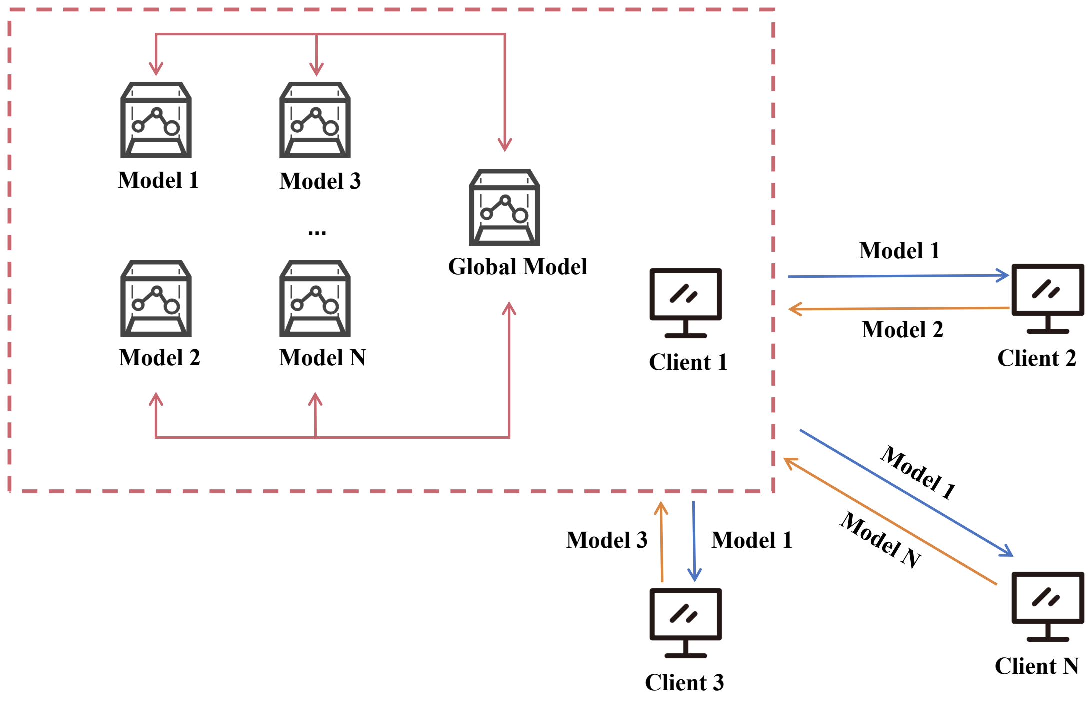


Figure 2-2 End-to-end Federated Learning Architecture[29]

2.2.1.2 Federated Learning Aggregation Algorithms

In federated learning, what is usually uploaded by the client is the gradient obtained through machine learning computation. The gradient is an important mathematical concept commonly used in the field of machine learning, which is a vector representing the rate of change of a function f at a differentiable point. Each element in the gradient vector corresponds to the partial derivative of f with respect to each of its variables. The smaller the gradient, the closer the function is to a minimum value. Machine learning algorithms often need to maximize or minimize a certain function, which is called the objective function. One class of functions that need to be minimized is the loss function, which is used to evaluate the similarity between the predicted results of the machine learning model and the true results, and thus measure the predictive ability of the model. The Stochastic Gradient Descent (SGD)[30–32] is a common method for finding the minimum loss function. In federated learning, a main task is to find the optimal loss function, so each client needs to upload the gradient of its local model, and the central server aggregates these gradients into a global model to determine the direction of change of the global loss function, so as to obtain the optimal loss function.

In federated learning, commonly used aggregation methods include the FedAvg algorithm and the FedProx algorithm. FedAvg[28], proposed by Google in 2016, is a basic federated learning aggregation algorithm that iteratively updates model parameters using SGD. The algorithm steps are as follows:

(1) Initialize global model parameters.

(2) The central server sends the global model to all clients and selects some clients for local training according to a fixed proportion.

(3) The selected clients initialize their local models with the global model parameters and perform multiple rounds of gradient descent using mini-batch gradient descent to update the parameters multiple times.

(4) The selected clients send their local model parameters to the central server.

(5) The central server aggregates the global model parameters using a weighted average method and uses the global model parameters for the next round of training.

The FedAvg algorithm supports heterogeneous data, allowing training on different datasets on local devices, and can also reduce communication overhead with good generalization ability. However, there are still some disadvantages to the FedAvg algorithm. First, in the FedAvg algorithm, the number of local iterations is fixed. However, due to differences in computational power among devices, the time required to execute a fixed number of iterations varies, which may affect subsequent server aggregation operations. Second, if some devices fail to converge within the specified time, their models run the risk of being ”discarded” by the server.

FedProx[33] is an improved federated learning aggregation algorithm designed to handle non-independent identically distributed (Non-IID) data and alleviate device heterogeneity issues. The algorithm makes the following two modifications to FedAvg:

(1) To address system heterogeneity, FedProx allows incompletely trained local models to be included, effectively solving the problem of client models being discarded due to insufficient computational power in FedAvg. This also means that FedProx does not require all local models to be trained with high precision, and clients can adjust the degree of training based on their computational power.

(2) To address statistical heterogeneity, FedProx adds a proximal term to the original loss function, which is used as the objective function for local models. Due to the statistical heterogeneity, excessive updates to local models can lead to bias in the global model. The proximal term penalizes local models that deviate too much from the global model, limiting the deviation of local models from the global model. The addition of the proximal term helps improve model accuracy and promote efficient aggregation. Since the data set on each client is incomplete relative to the total data set, it may not be representative, and a regularization term is needed to optimize the local objective function to

prevent the gap between the local model and the global model from becoming too large. This is one of the functions of the proximal term.

2.2.2 Attacks and Defense Strategies in Federated Learning

Federated learning, as a decentralized machine learning method for achieving privacy protection, can protect the data security and privacy of participants to a certain extent. However, in practical applications, federated learning still faces various data security and privacy threats. To further investigate these threats and provide corresponding defense measures, this thesis classifies and summarizes the attack methods against data security and privacy in federated learning based on published literature[34–36], and outlines the corresponding defense techniques and their research status.

In federated learning, there are two types of threats: security threats and privacy threats. Security threats mainly refer to malicious attacks that can affect the training results and speed of federated learning. Privacy threats mainly refer to attempts to obtain private information during the federated learning process, but this behavior generally does not interfere with the training of federated learning. The security and privacy threats of federated learning can be classified into data processing and training stages. During data processing, federated learning may face security threats such as poisoning attacks and free-riding attacks, as well as privacy threats such as sample leakage. During training, federated learning may face security threats such as poisoning attacks, communication bottleneck attacks, free-riding attacks, and intrusion attacks, as well as privacy threats such as model extraction attacks and inference attacks. The following sections will describe several attack methods in detail and introduce corresponding defense measures.

(1) Poisoning Attacks

Poisoning attacks are a type of security threat to federated learning, aiming to interfere with the training process and results of federated learning models by tampering with training data or manipulating models. This type of attack can result in decreased model performance, reduced accuracy, or misleading outputs. Poisoning attacks can be classified based on the attack method and target. According to the attack method, poisoning attacks can be divided into data poisoning and model poisoning. According to the attack target, poisoning attacks can be divided into Byzantine attacks and backdoor attacks.

Data poisoning: Attackers intentionally manipulate training data to affect the performance of machine learning models. Attackers may add fake data, modify labels, or introduce noise to manipulate the training process. Tolpegin et al.[37] studied label-flipping attacks in data poisoning, where attackers send data with incorrect labels to the training model, causing the global model to be contaminated even with only a few malicious users. To prevent data poisoning attacks, parameters should be checked before aggregation or screened during the training process to avoid the impact of contaminated data. Cao et al.[38] proposed a novel Sniper method, where they detect parameters during aggregation to identify honest users and reduce the impact of data poisoning attacks. However, this method has a small client data size and cannot address the problem of global model poisoning in multi-task scenarios.

Model poisoning: Attackers intentionally tamper with the model itself, causing the model to produce incorrect results during the inference phase. A common attack method is to boost malicious updates to enhance the attack effect[39]. Typically, the server evaluates the contribution of local models to the global model and marks those that do not significantly contribute to the global model. This marking method is gradually accumulated and used to determine potential attackers. The server can also filter out abnormal local models by comparing all local models. Bhagoji et al.[40] proposed an alternating minimization strategy to defend against model poisoning attacks. Cui et al.[41] proposed a federated learning framework based on generative adversarial networks for anomaly detection in smart grids and combined differential privacy algorithms and blockchain technology to protect user privacy and protect the model from poisoning attacks. However, this method consumes relatively high communication and computation costs.

Byzantine attacks: Attackers participate in training as malicious nodes, aiming to hinder the model's convergence or prevent it from achieving optimal performance, and the attack does not target specific users or data samples. Shejwalkar et al.[42] proposed an optimized Byzantine attack that fine-tunes malicious gradients and finds results close to the maximum value to achieve a more effective attack effect. To defend against Byzantine attacks on federated aggregation, Xia et al.[43] proposed a fast aggregation algorithm that screens and excludes gradients with large differences from the average gradient in each aggregation to obtain more accurate gradient information. This defense mechanism helps improve the accuracy and robustness of federated learning models.

Backdoor attacks: Attackers embed specific patterns or trigger conditions to cause the model to produce misleading results on specific backdoor inputs. Backdoor attacks usually deceive the training model into misclassifying attack samples as pre-set target label categories during the testing phase. Sun et al.[44] proposed an innovative backdoor attack method that allows selecting specific samples in the target task for labeling, and the authors efficiently implemented this attack method in TensorFlow's federated learning library TensorFlow Federated (TFF). Model pruning typically removes parameters that contribute little to the global model or may be manipulated and then re-aggregates the remaining parameters. This method is usually used to counter poisoning attacks and backdoor attacks and enhance the model's resistance. Kaviani et al.[45] proposed a pruning technique specifically for backdoor attacks, which can effectively remove malicious parameters and reduce the risk of the model being attacked, thereby ensuring the quality of the model. However, changing the neural network structure may cause the performance of this method to decrease, and its applicability may be limited by different datasets and neural network architectures.

(2) Free-riding Attacks

Free-riding attacks refer to the behavior of some participants in federated learning with incentive mechanisms who benefit from others' contributions without making corresponding contributions themselves. Specifically, malicious users disguise themselves as normal users, consume only a portion or none of their local data, and attempt to obtain model updates and training results from other participants. Wan et al.[46] proposed a new free-riding attack strategy, where they train a model using a small dataset locally and disguise it as a large dataset to obtain more rewards. To counter free-riding attacks, a widely used solution is to apply blockchain technology. Weng et al.[47] proposed a blockchain-based incentive mechanism to encourage active contributions from participants and drive federated learning. However, this method may result in relatively high computation delays and may not be suitable for scenarios with a large number of users.

(3) Communication Bottleneck Attacks

Communication bottleneck attacks refer to the exploitation of bottlenecks or insufficient bandwidth in network communication to attack a system. These attacks are relatively rare but can have severe consequences. Yao et al.[48] emphasized the vulnerability of communication bandwidth and the limitations of single federated learning training models, which result in high communication costs and become the primary challenge affecting the convergence of federated learning models. The direct solution to communication bottleneck attacks is to reduce communication overhead. Knowledge distillation[49] is an effective model compression technique proposed by Hinton that aims to reduce communication overhead, conserve computational resources, and optimize structural models, effectively countering communication bottleneck attacks. Furthermore, knowledge distillation enhances the robustness of models and effectively mitigates the threat of poisoning attacks. Momcheva et al.[50] proposed a framework that combines FedMD with knowledge distillation and transfer federated learning, allowing users with different computing capabilities to design their own network structures to improve the performance of local models. However, users need to sacrifice a certain degree of data privacy to share datasets. Lin et al.[51] proposed an optimized distillation algorithm based on the FedMD framework that integrates the output of local models on unlabeled data for model fusion. This method significantly speeds up model training, reduces communication rounds, and lowers the risk of data privacy leaks. However, the difference between the distillation effect and the effect of directly fusing model parameters requires further investigation, and the distillation effect of this method largely depends on the choice of the public dataset.

(4) Model Extraction Attacks

Model extraction attacks[52] refer to the process in which an attacker tries to replicate a functionally similar or identical machine learning model by repeatedly sending data and observing corresponding responses to infer the model's parameters and functionality. Jagielski et al.[53] proposed an efficient attack method that optimizes the efficiency of model extraction and enables direct extraction of model weights without the need for training. Adversarial training is a commonly used method to counter model extraction attacks. It enhances the system's resistance to interference by introducing minor perturbations during model training. Li et al.[54] proposed a federated adversarial learning (FAL) method and formulated FAL training as a min-max optimization problem, enhancing global robustness and effectively propagating resistance to interference. However, the width of the neural network's hidden layer and the choice of boundaries in this method require further research.

(5) Inference Attacks

Inference attacks refer to the process in which an attacker infers additional information about a target model by observing and querying its inputs. Inference attacks can be categorized into membership inference attacks and attribute inference attacks based on their targets.

Membership Inference Attacks (MIAs): Attackers infer whether a specific sample exists in the training set of a model by analyzing the model's prediction behavior and output features for different input samples. Nasr et al.[55] proposed a membership inference attack method based on white-box models, achieving an accuracy rate of over 70% in white-box scenarios. Solutions for membership inference attacks typically include homomorphic encryption and secure multi-party computation. Homomorphic encryption (HE)[56] is a probabilistic encryption technique that allows data processing without accessing the data itself. Its core objective is to ensure that users with a key can decrypt encrypted data and obtain results consistent with plaintext processing. Ma et al.[57] proposed an optimized fully homomorphic encryption algorithm and constructed a more effective privacy-preserving federated learning framework. However, the computational overhead of this algorithm is significant in practical scenarios. Secure multi-party computation (SMC)[58] is a protocol or framework that enables multiple parties to collaborate securely to solve model or function problems without a trusted third party. Kaissis et al.[59] proposed the PriMIA framework for training federated learning on medical imaging data, achieving secret sharing of functions. However, deploying this framework requires significant computational resources, and encrypted inference leads to high latency. Moreover, no specific countermeasures have been taken against low-quality or malicious contributions.

Table 2-2 System Attacks & Defense in Federated Learning

Threat	Attack	Damage	Stages	Defense Methods
Security	Data poisoning[37]	Integrity	Data Prep	Robust Aggregation[38]
	Model Poisoning[39]	Integrity	Training	Anomaly Detection[40,41]
	Byzantine Attacks[42]	Availability	Training	Robust Aggregation[43]
	Backdoor Attacks[44]	Availability	Training	Pruning[45]
	Free-riding Attacks[46]	Integrity	Data Prep	Blockchain[47]
		Availability	Training	
	Communication Bottleneck[48]	Availability	Training	Knowledge Distillation[50,51]
Privacy	Model Extraction[53]	Confidentiality	Training	Adversarial Training[54]
	MIAs[55]	Confidentiality	Training	HE[56,57]
				SMC[58,59]
	Attribute Inference[60]	Confidentiality	Training	DP[61–63]

Attribute Inference Attacks: Attackers infer sensitive attributes of input data, such as personal information, features, or other sensitive information, by observing the model's output results. Zhang et al.[60] proposed a property inference attack method based on black-box models, allowing attackers to infer sensitive data attributes not involved in training by calling trained models. Although the correlation between these sensitive attributes and training attributes is low, there is still a risk of privacy exposure. Differential privacy (DP) algorithms are commonly used to counter attribute inference attacks. Differential privacy algorithms[61] add noise to hide the true value of transmitted gradients, typically used to protect data confidentiality. Yang et al.[62] proposed the PLU-FedOA algorithm, which aims to optimize local differential privacy protection in horizontal federated learning. This framework is more efficient than homomorphic encryption or secure multi-party computation and satisfies the privacy requirements of participants under the same privacy budget. Cao et al.[63] proposed a differential privacy algorithm based on functional mechanisms, which reduces the impact of noise on gradients by perturbing the objective function, significantly improving model accuracy. However, this algorithm results in high communication and latency overhead.

Table 2-2 summarizes various attack types in federated learning and their corresponding defense methods. Depending on the specific attack method, appropriate defense strategies can be selected to protect data security and privacy in the data processing and training stages of federated learning.

Although federated learning has been widely applied and developed in data security and privacy protection, its availability, reliability, and robustness have been improved. However, financial data contains sensitive user data and a large amount of transaction information, so maintaining the security and privacy of financial data, improving the accuracy of fraud detection models, and ensuring the stability of financial institution business systems are crucial. Federated learning alone may not meet the data security and privacy requirements of financial institutions because federated learning-based fraud detection still faces the following issues:

- Data security challenges in fraud detection include the severe data imbalance problem between fraudulent and normal samples. If not handled properly, this imbalance can cause training bias, affecting the accuracy of the model and even threatening the security and privacy of financial data. Additionally, during the model exchange process in federated learning, there is a risk of model data being stolen and the original dataset's features being inferred. If not properly addressed, this can seriously threaten the security and privacy of client-side data. These issues can reduce the accuracy of fraud detection and compromise the security and privacy of financial data.
- Communication cost challenges in federated learning arise from the increasing number of participants and model complexity, leading to increased data volume between the server and clients, with network bandwidth limitations becoming the primary communication bottleneck. Therefore, reducing communication costs remains an urgent challenge in federated learning. Strategies for optimiz-

ing communication costs mainly involve resolving the conflict between the size of the transmitted model data and model accuracy, as transmitting simpler models can reduce communication costs but may lower the accuracy of model detection. The second strategy involves exploring optimized model transmission mechanisms, as fewer transmission cycles result in lower communication costs. These two aspects remain the focus of research on reducing communication costs in federated learning.

Therefore, in the context of multiple financial institutions collaborating on fraud detection, this thesis proposes a federated learning fraud detection method that integrates differential privacy protection. The goal of this method is to improve the accuracy of fraud detection models and build a secure, efficient federated learning environment.

2.3 Data Protection Research in Blockchain

2.3.1 Blockchain

Blockchain[64] is a special type of distributed ledger introduced by Nakamoto in 2008, primarily used for storing information. It employs cryptographic methods to ensure data security and reliability. Blockchain mainly consists of blocks, nodes, consensus algorithms, and encryption techniques:

- Blocks: Each block consists of a block header and block body. As shown in Figure 2-3, a blockchain is a chain of blocks in sequence. The block header contains identification information for each block, such as version number, hash value of the previous block, and timestamp. The block body stores a large amount of transaction data.
- Nodes: Each node is a computer connected to the network, and they jointly maintain the blockchain.
- Consensus algorithms: Consensus algorithms determine the rules for whether a node can join the blockchain. For example, Bitcoin uses a proof-of-work (PoW) consensus algorithm.
- Encryption techniques: Encryption techniques maintain the security of the blockchain, including public-key cryptography, hash functions, and digital signatures.

Blockchain plays a significant role in data security and privacy protection due to its following features:

- Decentralization: Blockchain uses a distributed storage model where every node equally stores a complete database. When data is read or written to one node, all nodes are updated synchronously to maintain consistency.
- Openness: The system data of the blockchain is transparent and open to all, except for the private information of the transaction parties, which is encrypted.
- Autonomy: To enable nodes to trust each other and exchange data, blockchain uses norms and protocols based on negotiated consensus, significantly reducing human intervention.
- Immutability: In blockchain, once information is added, it is permanently stored. As a single node cannot modify the database, the stability and reliability of blockchain data are extremely high.
- Anonymity: In blockchain, the identities of the transaction parties do not need to be disclosed, effectively protecting personal information.

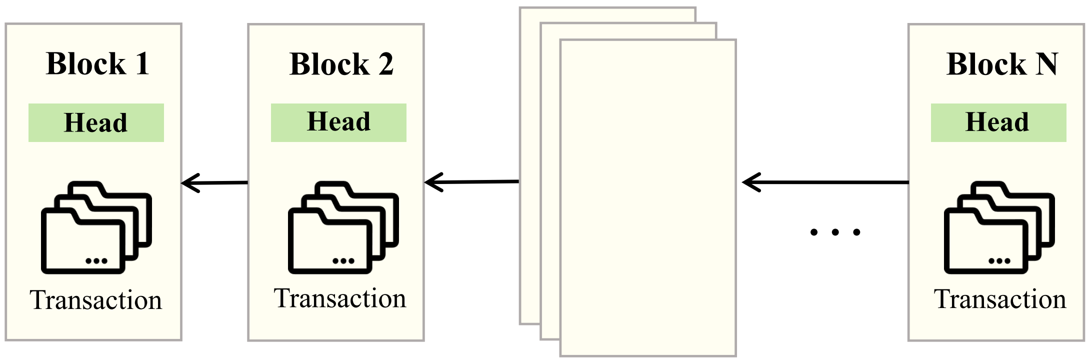


Figure 2-3 Blockchain Diagram

2.3.2 Data Security and Privacy Protection in Blockchain

In recent years, combining blockchain with federated learning has become a hot topic in the field of the Internet of Things (IoT). Blockchain technology provides data security and privacy protection support for federated learning, covering data storage, defense against malicious attacks, incentive protocols, and model tracing functions. Blockchain-based federated learning frameworks effectively ensure data security and privacy, eliminate single point failures, and solve trust issues. Currently, blockchain-based federated learning frameworks have been widely applied in healthcare, drone networks, Industry 4.0, and other fields[65].

Blockchain technology can provide decentralized data storage functionality for federated learning, effectively protecting the security and privacy of participants' data while enabling data sharing. Majeed et al.[66] proposed the FLchain architecture, which aims to protect the data security of federated learning. This framework supports blockchain networks, and each model is stored in the blockchain to ensure its security. Experimental results show that the blockchain network successfully completed computation, verification, and storage with high robustness. Qi et al.[67] proposed a blockchain-based federated learning architecture to address security vulnerabilities in centralized federated learning. Experimental results show that the system has high security and can be used in traffic control systems and other departments. Potap et al.[68] proposed a multi-agent system that aims to divide tasks into different objects to perform different activities while integrating blockchain encryption technology to ensure data security and privacy in the multi-agent system. Experimental results show that the system has high security.

The immutability feature of blockchain provides federated learning with a scheme to resist malicious attacks. Chai et al.[69] proposed a federated learning algorithm with blockchain functionality to share vehicle network data, and the proposed active blockchain framework can effectively resist some malicious attacks. Experimental results show that the blockchain hierarchical framework not only improves the reliability and security of data sharing but also is compatible with large-scale vehicle networks with different features. Short et al.[70] proposed a blockchain defense technique for federated learning systems. Experimental results show that this technique can effectively resist model poisoning attacks, and the enhanced federated learning algorithm has higher security and faster convergence speed.

Blockchain also provides an incentive mechanism for federated learning, encouraging participants to actively participate in federated learning tasks. Gao et al.[71] proposed a blockchain-based federated learning incentive mechanism that evaluates participants based on reputation and contribution indicators, and task publishers fairly reward participants to attract high-efficiency participants while punishing and eliminating malicious participants. Experimental results show that the profit of this incentive system is more than 46.7% higher than the baseline. Lu et al.[72] proposed a blockchain-based federated learning framework that applies blockchain to digital twin federated learning to reasonably allocate IoT resources. Experimental results show that the system can effectively improve efficiency and protect communication security and data privacy. Kang et al.[73] proposed a method using consortium blockchain algorithms to protect reputation data in federated learning to select reliable workers through reputation data. Experimental results show that the designed system has high reliability in the mobile network domain.

Blockchain technology can ensure model transparency and assist federated learning in achieving model traceability. Chen et al.[74] proposed a framework called PPTFL based on federated learning that achieves high-performance privacy protection and traceable federated learning. The authors first proposed hierarchical aggregation federated learning to protect privacy with lower communication overhead. Then, they combined federated learning with blockchain and distributed file systems to make the framework traceable and tamper-resistant. Extensive experiments have proven the effectiveness of the framework.

Table 2-3 The Role and Advantages of Blockchain in Federated Learning

Role		Advantages
Blockchain	Data Storage	Secure and Private Storage Solution[66–68]
	Defense Against Attacks	Resistant to Malicious Attacks[69,70]
	Incentive Mechanism	Encouraging Active Participation[71–73]
	Model Traceability	Participant Model Tracking Solution[74]

Table 2-3 summarizes the role, advantages, and relevant literature of blockchain in federated learning. This thesis combines blockchain with federated learning and applies it to fraud detection in financial data. By designing a reasonable blockchain storage pattern, the security and privacy of data are protected, while achieving traceability of local models in federated learning.

2.4 Summary

This chapter first introduced the risk challenges and solutions in data security and privacy protection in the financial domain. Then, it provided an overview of the relevant background of federated learning and the attack and defense strategies in federated learning. Finally, it discussed the relevant background of blockchain and its contributions to data security and privacy protection. These research findings lay the foundation for the design of the subsequent framework and the implementation of the algorithm.

3 Federated Learning Fraud Detection with Differential Privacy

The proposed architecture uses federated learning with differential privacy to enhance fraud detection accuracy while ensuring data security and privacy in collaborative tasks among financial institutions with distributed user data. It aims to secure model training and data transfer for each institution.

3.1 Problem Description

Current fraud detection practices among financial institutions often involve distributed methods with limitations, such as data security and privacy concerns during dataset transfers. This chapter explores a secure data sharing approach using federated learning with differential privacy protection to balance data sharing and security for fraud detection training. Federated learning offers important advantages in protecting data security and privacy, as clients can use local datasets for training without sharing raw data, reducing data leak risks. Differential privacy algorithms further enhance data privacy protection with low computational cost, ease of implementation, and mathematical privacy guarantees. This chapter aims to provide more reliable services to financial institutions and address security challenges in fraud detection tasks by using federated learning with differential privacy protection.

In a federated learning scenario with N clients, each client i has a local dataset D_i. In each round of iteration, client i calculates the loss function $f(w^i)$ based on D_i and updates the model parameters using an optimization algorithm. In the t-th round, client i calculates $f(w_t^i)$ and updates the model parameters as follows:

$$w_{t+1} = update(w_t^i, D_i), \tag{3-1}$$

where update(·) refers to the function used by participant i to update the model parameters based on their local dataset D_i.

In fraud detection, the optimization objective of federated learning is the loss function and accuracy. The loss function measures the model's prediction error, while accuracy evaluates its classification performance. The global loss function is obtained by averaging the local loss functions of the participants.

$$\hat{f}(w) = \frac{1}{N}\sum_{i=1}^{N} f(w, D_i), \tag{3-2}$$

where $\hat{f}(w)$ represents the global loss function, and $f(w, D_i)$ represents the loss function of participant i with model parameters w. Similarly, the global accuracy is obtained by computing the average of the local accuracies of the participants:

$$\hat{F}(w) = \frac{1}{N}\sum_{i=1}^{N} F(w, D_i), \tag{3-3}$$

where $\hat{F}(w)$ represents the global accuracy, and $F(w, D_i)$ represents the local accuracy of participant i with model parameters w.

3.2 Overall Architecture Design

The federated learning fraud detection framework with differential privacy protection, as depicted in Figure 3-1, comprises three main components: (1) server, (2) client, and (3) blockchain. The server aggregates and distributes the global model, while the clients train local fraud detection models and upload them to the server. The blockchain stores metadata of the global and local models, ensuring trustworthiness and traceability. The subsequent sections will elaborate on the design and operation of the server, client, and blockchain modules.

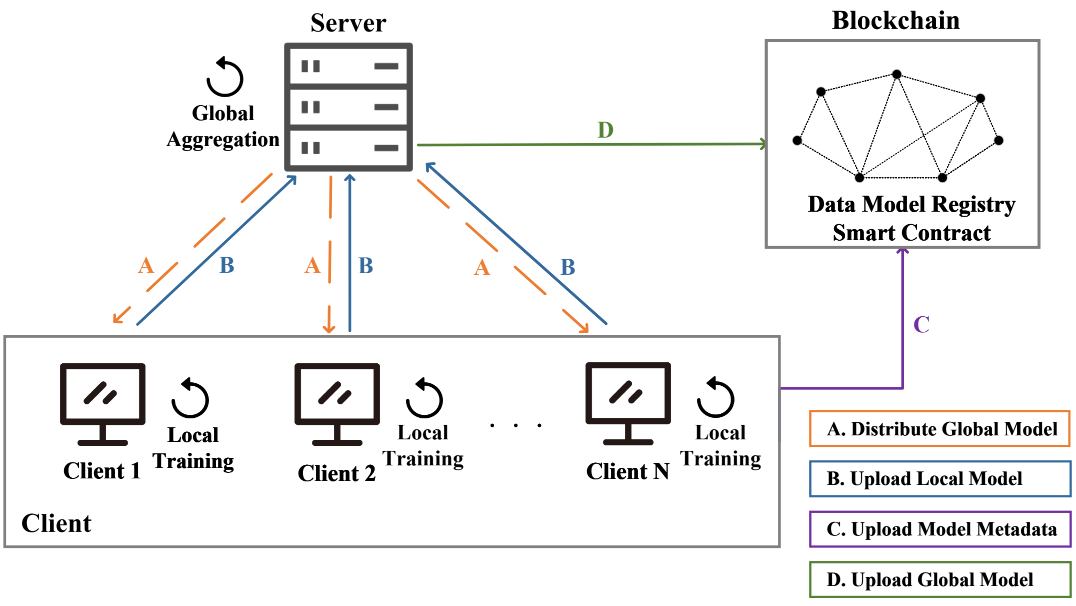


Figure 3-1 Blockchain-based Federated Learning Architecture: Server, Clients, and Blockchain

Server-side Design

As shown in Figure 3-2, the server consists of three components: model deployer, model aggregator, and global database. The model deployer is responsible for generating the initial global model and distributing the updated global model to the clients. The model aggregator receives local models from the clients and aggregates them into a global model. The global database stores the global model.

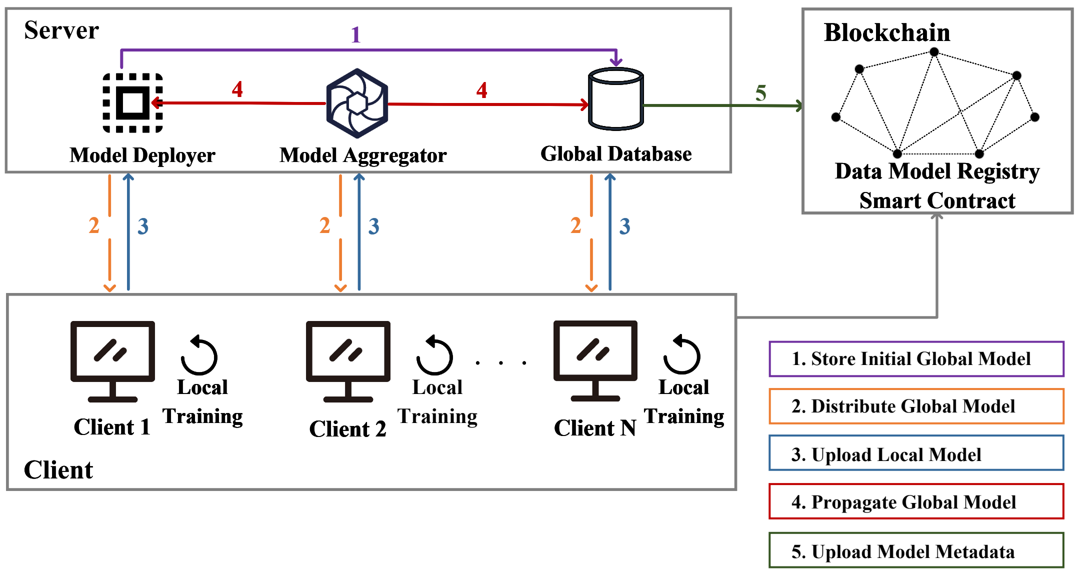


Figure 3-2 Blockchain-based Federated Learning Architecture: Server

- Model Deployer: The model deployer generates the initial global model and transfers it to the global database. Additionally, the model deployer broadcasts the global model to each client for fraud detection.
- Model Aggregator: The model aggregator waits for clients to upload local model parameters and performs aggregation operations after receiving all local models. Then, the model aggregator transfers the updated global model to the global database and the model deployer, which will be broadcast for the next round of training.
- Global Database: The global database stores the global models generated or aggregated by the server and uploads the metadata of the models to the blockchain to determine their origin.

Client-side Design

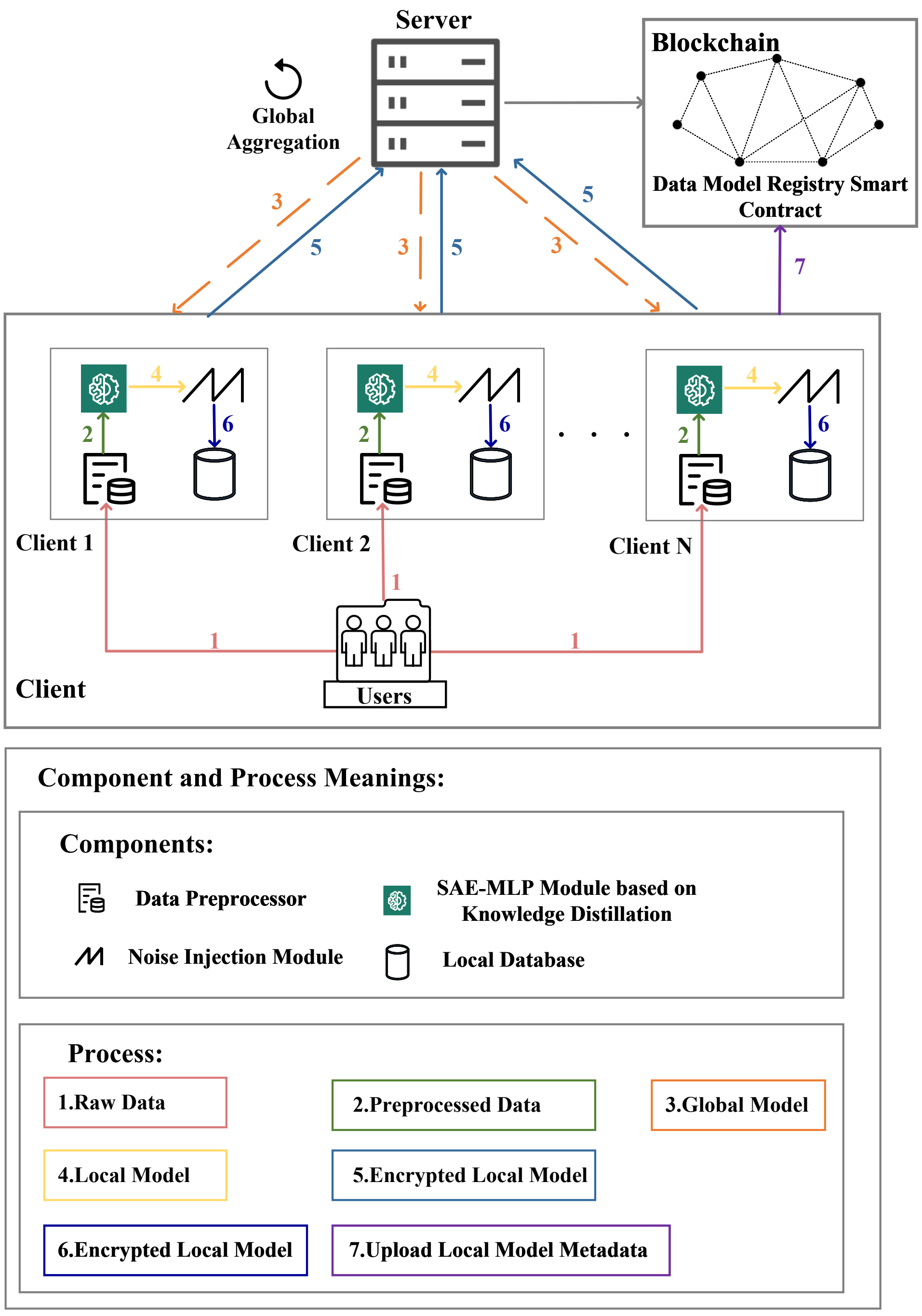


Figure 3-3 Blockchain-based Federated Learning Architecture: Client

As shown in Figure 3-3, the client consists of five components: user, data preprocessor, SAE-MLP based on knowledge distillation module, noise injection module, and local database. The user generates raw data. The data preprocessor is used to preprocess the raw data to address data imbalance issues. The SAE-MLP based on knowledge distillation module is used to train and compress the local fraud detection model. The noise injection module encrypts the model data by executing local differential privacy algorithms. The local database stores the local model.

- User: The user generates raw data, which is collected by the client and transmitted to the data preprocessor for processing.
- Data preprocessor: The data preprocessor processes the imbalanced raw data and sends the processed data to the SAE-MLP based on knowledge distillation module.
- SAE-MLP based on knowledge distillation module: The SAE-MLP based on knowledge distillation module receives the global model distributed by the server and uses it to train the local dataset to obtain the local model.
- Noise injection module: The noise injection module performs differential privacy algorithms on the trained model to add noise and encrypt the local model.
- Local database: The local database is responsible for receiving and storing the encrypted local model data and uploading the metadata of the model to the blockchain to determine the origin of the model.

System Concept Based on Blockchain-supported Environment

Blockchain, as an internet protocol that can solve data trust issues, provides a trustworthy mechanism for the participants of federated learning. Under this mechanism, the model parameters of federated learning can be stored in the blockchain to ensure their security and reliability. However, storing large amounts of data on the blockchain can easily cause chain overload problems. Therefore, this thesis adopts a chain-off data storage design pattern[75] to ensure the efficiency of data storage and the traceability of the model.

As shown in Figure 3-4, in the client-side, the raw data first undergoes preprocessing through the IDW-SMOTE algorithm. The preprocessed data is then used as input for SAE feature extraction to train the fraud detection model. Subsequently, the features extracted by SAE are used to train the MLP classification model. The trained model data is protected by the differential privacy algorithm and stored in a database and blockchain using the chain-off storage design pattern. Finally, the server aggregates all client-side model data and updates the global model. The updated global model is then distributed to each client to begin the next round of federated learning training.

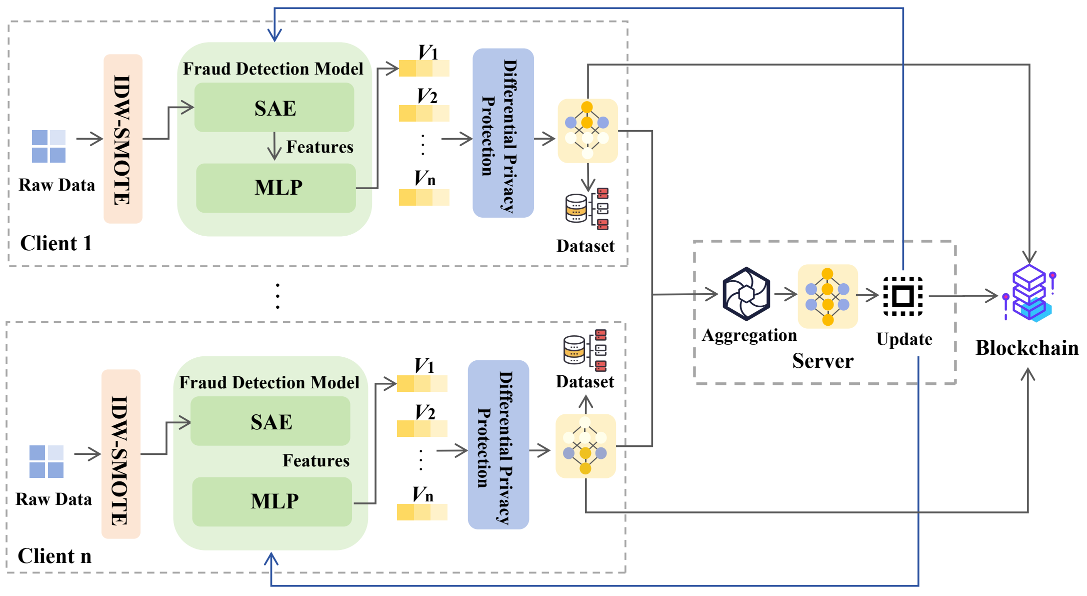


Figure 3-4 A System Running on A Blockchain-supported Environment

The chain-off storage design pattern involves model data, metadata, databases, and blockchain. Model data refers to the local models trained by the client or the global models aggregated by the server. Metadata is data that describes data and provides information about the data, such as attributes, structure, timestamps, data owner signatures, data paths, etc. Databases exist locally on the client or server side and are used to store model data. The blockchain is used to record metadata and provide traceability. The prerequisite for applying this pattern is that the server and each client have generated at least one blockchain node that records local relevant data to ensure the trustworthiness of the participants. The interaction process of this pattern is shown in Figure 3-5:

After local training is completed on the client-side or global aggregation is completed on the server-side, the model data is stored in a local database or a global database. Meanwhile, metadata related to the model data needs to be extracted, including version number, timestamp, and hash value information. The version number is used to distinguish different versions of the model, the timestamp records the time when the model is stored, and the hash value ensures the integrity and consistency of the model. Then, smart contracts are used to generate a blockchain for the metadata and add it to the blockchain network. Finally, the integrity and consistency of the model can be verified based on the metadata in the blockchain. By comparing hash values, the model data in the database can be matched with the metadata in the blockchain to ensure that the model has not been tampered with. Additionally, the version number and timestamp in the blockchain metadata can be used to trace the model's history and training process.

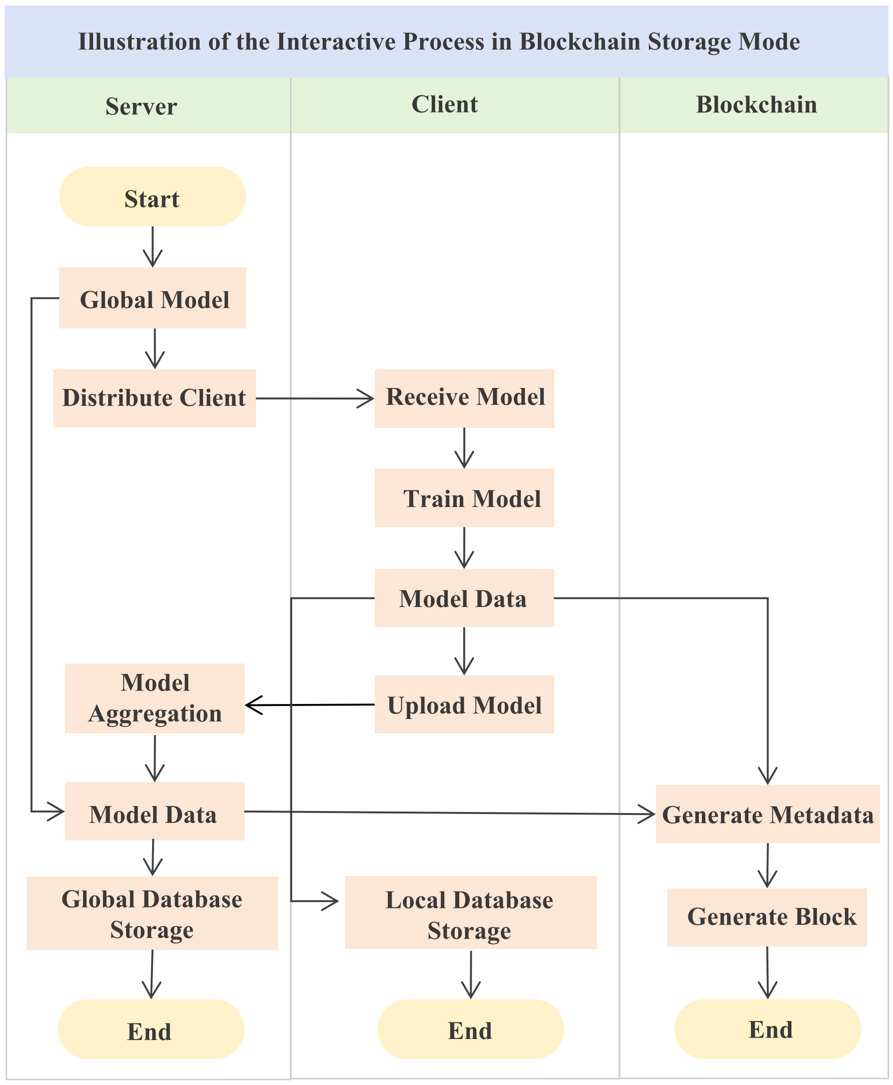


Figure 3-5 Blockchain Storage Model Interaction Diagram

This section has detailed the design of the federated learning fraud detection framework with differential privacy protection, covering the server-side, client-side, and blockchain aspects. Next, the design and implementation of the IDW-SMOTE algorithm, fraud detection model, and differential privacy algorithm will be introduced in turn based on the client-side operation flow.

3.3 IDW-SMOTE Data Preprocessing Algorithm

This thesis proposes the IDW-SMOTE data preprocessing algorithm to address the data imbalance problem in fraud detection and improve the accuracy of detection models. The algorithm generates high-quality synthetic samples with better coverage of the feature space of fraud classes, providing more accurate and comprehensive data distribution for subsequent fraud detection models and improving fraud detection accuracy.

Data imbalance is a common and critical problem in fraud detection, as the number of normal samples greatly outnumbers the number of fraud samples. During training, models may focus more on normal data and ignore fraud data, which can decrease the accuracy of fraud detection results. This bias towards the majority class may prevent models from fully learning and understanding the features and patterns of fraud samples. Therefore, addressing data imbalance in the data preprocessing stage is crucial.

This thesis proposes the IDW-SMOTE data preprocessing algorithm based on the Synthetic Minority Over-sampling Technique[76] (SMOTE) and the Inverse Distance Weighted (IDW) method. SMOTE is an oversampling technique that generates samples for the minority class to balance the sample distribution with the majority class. However, traditional SMOTE does not change the contour features of the original sample distribution, which means that its impact on class boundaries is relatively small, and the generated sample points may not conform to the real sample distribution. To solve this problem, this thesis proposes an improved SMOTE algorithm: IDW-SMOTE.

The pseudocode of IDW-SMOTE is shown in Algorithm 1:

```
Algorithm 1: IDW-SMOTE

Input: Minority class: A, any sample in A: $x$, number of neighboring points:
       N, number of new samples to be generated: M.
Output: Newly generated class: A
begin
    for $i$ = 1,2, ... , $M$ do
        Select a random sample $x$ from A
        Randomly select N samples from A and neighboring points of $x$ to
          obtain the K-nearest neighbors $x_neighbour_j$, $j$ = 1,2,..., N
        Calculate the Euclidean distance $d_j$ between $x$ and $x_neighbour_j$
        Calculate the weight $w_j$ of neighbor $j$ according to formula 3-4
        Generate a new sample point $x_{new}$ according to formula 3-5
        Add $x_{new}$ to A
    end
    return $A$
end
```

(1) For any random sample x in the minority class **A**, select **N** samples from class **A** and its neighboring points to obtain the K-nearest neighbors of sample x. The sampling

number **N** is manually determined based on the sampling imbalance ratio.

(2) For sample x, calculate the Euclidean distance d_j (j = 1,2,..., **N**) between it and each of its neighboring points. The weight of each neighbor is calculated using inverse distance weighting. The calculation formula is as follows:

$$w_j = \frac{1/d_j}{\sum_1^N 1/d_j}, \tag{3-4}$$

where w_j represents the weight of neighbor j.

(3) A new sample point x_{new} is generated according to formula 3-5 and added to class **A**.

$$x_{new} = x + \sum_{i=0}^{N} (-1)^{\text{randam(100)}} (x_i - x) \cdot w_i, \tag{3-5}$$

where randam(100) represents randomly selecting an integer within the range (1,100).

(4) Repeat steps (2) and (3) until the number of samples in the minority class is equal to the number of samples in the majority class.

The IDW-SMOTE data preprocessing algorithm addresses the problem of sample imbalance, and the processed data can be used as input for stacked autoencoders in subsequent model training.

3.4 Multi-layer Perceptron Based on Stacked Autoencoders

This thesis proposes a fraud detection classification model that combines the features of Stacked AutoEncoder[77] (SAE) and Multilayer Perceptron[78] (MLP) to improve the accuracy of fraud detection.

First, the model uses SAE to extract features from the input data processed by IDW-SMOTE. By selecting and extracting key information, SAE obtains more representative feature representations, which is a crucial step in improving model accuracy. In fraud detection tasks, the original data usually contains a lot of redundant information, and fraudulent behavior is often hidden in complex data patterns. Traditional extraction methods may not be able to capture these patterns. SAE, as a deep learning model, consists of multiple AutoEncoders[79] (AE) stacked together and has the ability to extract high-level features layer by layer. Compared with traditional feature extraction methods, SAE has stronger expressive and learning abilities, can extract key and representative features from complex original datasets, and thus improve the performance of fraud detection models. Therefore, using SAE for feature extraction is of great significance and value in fraud detection tasks.

Second, this thesis transforms the fraud detection problem into a classification problem and integrates the MLP classification model to train the extracted feature data to improve the accuracy of fraud detection. Before model training, the IDW-SMOTE algorithm is used to generate an equal number of fraudulent data and normal data, laying the foundation for subsequent classification modeling. Transforming the fraud detection problem into a classification problem has multiple advantages. First, using existing classification models, such as the MLP multilayer perceptron, can fully exert their powerful nonlinear modeling capabilities and adaptability to achieve efficient classification capabilities. Second, classification models have high interpretability and can judge the degree of fraud in the data, providing valuable references for subsequent processing. Furthermore, classification problems have flexibility and scalability and can select different classification algorithms and model structures according to actual needs to adapt to different fraud detection scenarios. Therefore, using MLP to transform the fraud detection problem into a classification problem is a practically meaningful method.

As shown in Figure 3-6, SAE is mainly used for feature extraction, and MLP is mainly used for data classification. The specific operations of using the SAE-MLP model based on the stacked autoencoder for fraud detection are as follows:

(1) Feature Extraction Using Stacked Autoencoders:

A stacked autoencoder is composed of multiple autoencoders that extract key feature representations from input data through layer-by-layer training and learning. The final feature results are stored in the hidden layer of the encoder part. This section will provide a detailed explanation of the working principle of the stacked autoencoder in the model from the perspectives of its structure, learning method, and training process.

In the proposed model, the stacked autoencoder is a deep learning model composed of three autoencoders, including an input layer, three hidden layers, and an output layer. Each autoencoder consists of an encoder and a decoder, so the stacked autoencoder also contains an encoder part and a decoder part. Specifically, the encoder part consists of two encoders, and the decoder part consists of two decoders. In the stacked autoencoder, the output of each autoencoder hidden layer serves as the input for the next autoencoder, forming a layer-to-layer transmission. The output layer of the previous autoencoder is directly connected to the input layer of the next autoencoder. This layer-by-layer transmission enables the autoencoder to progressively learn higher-level feature representations, thereby improving the model’s performance.

The stacked autoencoder learns in a layer-by-layer manner, aiming to reconstruct the sample. Through nonlinear transformation and feature extraction, the stacked autoencoder maps the input layer data to the hidden layer through the encoder and maps the hidden layer representation back to the input data through the decoder. Ideally, the decoder's output can perfectly or approximately recover the original input data. Through this encoding-decoding process, the hidden layer with fewer neurons obtains useful properties, equivalent to representing the input data with fewer features, effectively improving the model's performance. Additionally, the stacked autoencoder reduces the risk of overfitting because each layer can learn different feature representations of the data, reducing the model's dependence on the training data. This feature extraction method enhances the model's performance and reduces the risk of overfitting.

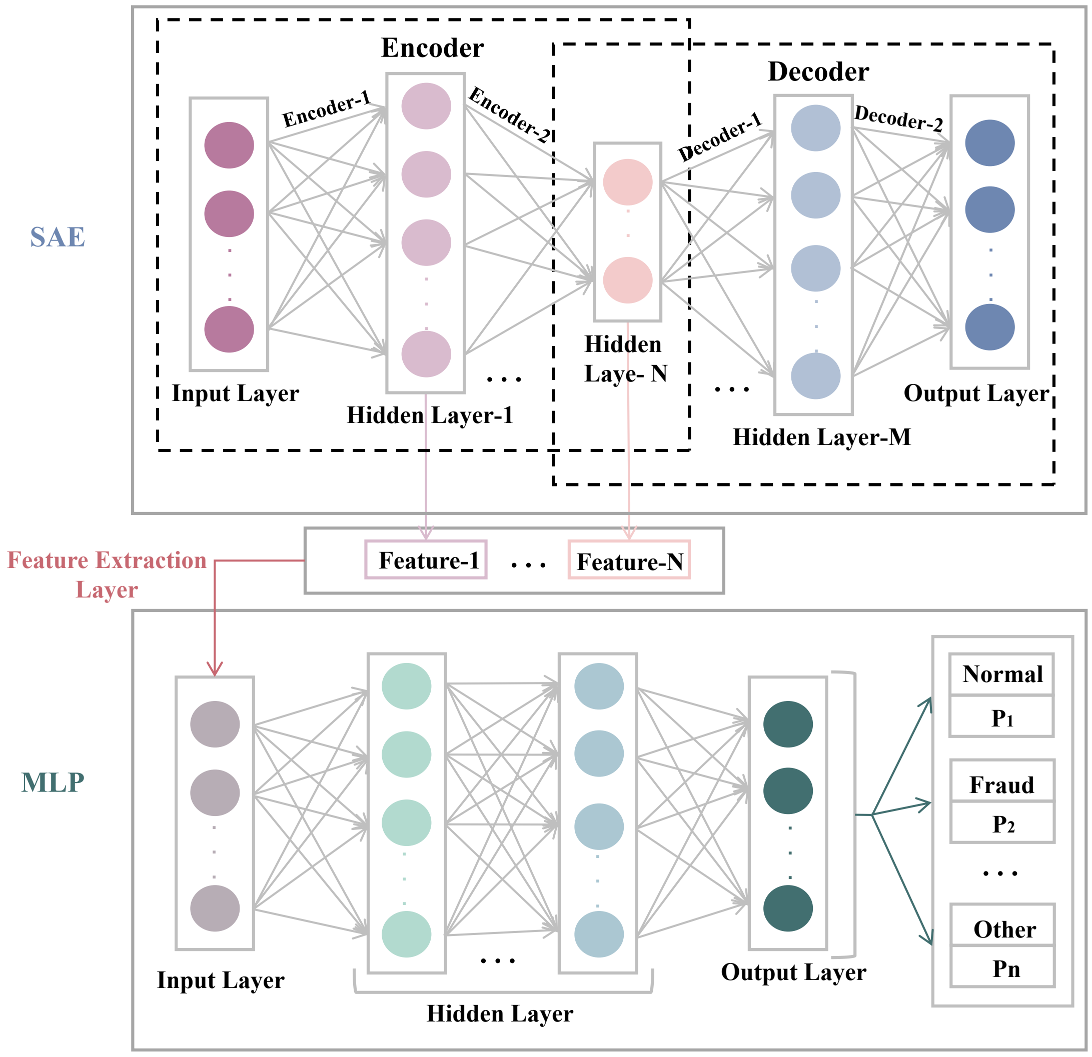


Figure 3-6 MLP Based on SAE

The feature extraction process using stacked autoencoders consists of two phases: the training phase and the feature generation phase. During the training phase, the stacked autoencoder is trained layer-by-layer using unsupervised learning, and the parameters of the neural network are adjusted using backpropagation algorithm and loss function to minimize the reconstruction error, enabling the output to perfectly or approximately recover the original input data. After obtaining the optimal model, the feature generation phase begins. The entire training data set is input into the trained stacked autoencoder, and the encoder portion of the stacked autoencoder is extracted as a feature extractor to obtain high-level feature representations of the original data set. These abstract representations of the original data will be used as input for subsequent classification tasks.

(2) Classification Model Based on Multilayer Perceptron:

In the proposed model, the MLP is used for feature data classification. The MLP consists of an input layer, three hidden layers, and an output layer. The input layer receives feature representations from the stacked autoencoder. The hidden layers use fully connected layers for linear transformation and feature extraction. The output layer is used for data prediction and classification, outputting the probability of each sample belonging to a particular category.

Training the classification model involves two phases: the training phase and the testing phase. During the training phase, supervised learning is used to train the model. The training data is passed through the input layer layer by layer and processed using an activation function to obtain the output result. The difference between the output result and the true label is compared using backpropagation, and the error is calculated and the network parameters are updated using the gradient descent algorithm to improve the model's performance. After training is complete, the testing phase begins. In the testing phase, an independent test data set is used to evaluate the model. The test data is passed through the input layer using the forward propagation algorithm to obtain the model's predicted results, which are compared with the true labels to evaluate the accuracy, precision, and other metrics of the classification model.

(3) Loss Function Calculation

Considering the difference between the model's output probability distribution and the actual labels, this thesis uses the cross-entropy function as the loss function for the MLP classification model. The cross-entropy loss function is constructed as follows:

$$L = \sum (y_i * log(p_i)), \tag{3-6}$$

where y_i represents an element in the actual label vector of the i-th sample, and p_i represents the corresponding element in the output layer's predicted probability distribution for the i-th sample.

The activation function of the output layer in the MLP is the Softmax function, which can convert the output into a probability distribution. Assuming that z_j represents the input of the j-th neuron in the output layer, the probability distribution of the output layer after passing through the Softmax function is:

$$p_j = softmax(z_j) = \frac{\exp(z_j)}{\sum_k \exp(z_k)}. \tag{3-7}$$

Substituting p_j back into the calculation formula of the cross-entropy loss function, the final loss function is obtained:

$$L = \sum(y_i * log(\frac{\exp(z_i)}{\sum_k \exp(z_k)})). \tag{3-8}$$

This section constructs a multi-layer perceptron model based on stacked autoencoders for classifying normal data and fraudulent data. The model will be implemented with encryption and compression in subsequent operations, with details to be described in Section 3.5 and Chapter 4.

3.5 Differential Privacy Protection

In federated learning, there is a risk of information leakage when clients upload their local models to the server. Attackers may intercept the local models during transmission and infer information about the original data. To protect data security during model transfer, this thesis introduces differential privacy algorithms in the federated learning architecture based on blockchain. Local differential privacy algorithms are executed before uploading local models, adding appropriate zero-mean Laplacian noise to the gradients. The pseudocode is shown in Algorithm 2.

(1) Receive Model Data and Perform Gradient Clipping

Gradient clipping is a mechanism in differential privacy algorithms that limits the range of gradients to reduce the information about individual data that may be contained in the gradients. Gradient clipping usually relies on a clipping function (clip) that limits each gradient element $g[j]$ in the model to the range [-δ, δ]. The clipping function is as follows:

$$clip(g[j], -\delta, \delta) = \begin{cases} -\delta, & g[j] < -\delta \\ \delta, & g[j] > \delta. \end{cases} \tag{3-9}$$

Assume that the model gradient of the i-th participant is g_i, which consists of multiple gradient elements $g[j]$. Then, the gradient g_{temp} after clipping for this participant can be represented as $clip\,(g_i, \delta)$.

Algorithm 2: Local Differential Privacy Protection

```
Input: Gradient $g_i$ of the $i$-th participant, clipping threshold $\delta$, scale
        parameter $\lambda$ of Laplacian noise.
Output: Encrypted gradient $G_i$ of the $i$-th participant
begin
    if $g_{temp} \leq \delta$ and $g_{temp} \geq -\delta$
        $g_{temp} = g_i$
    else if $g_{temp} > \delta$
        $g_{temp} = \delta$
    else
        $g_{temp} = -\delta$
    noise = $Laplace(0, \lambda)$
    $G_i = g_{temp} + noise_i$
    return $G_i$
end
```

(2) Add Laplacian Noise Perturbation

Laplacian noise is random noise generated based on the Laplacian distribution. Its probability density function is as follows:

$$f(x|\mu, b) = \frac{1}{2b}e^{\frac{-|x-\mu|}{b}}, \tag{3-10}$$

where μ is the location parameter and b is the scale parameter. In differential privacy, the Laplace distribution with zero mean is commonly used to add noise, denoted as $Laplace(0, \lambda)$, where $\lambda = \Delta f/\varepsilon$, where Δf represents sensitivity, which is a measure of the similarity between two datasets. Sensitivity is determined by the data distribution in the client's dataset, and the added noise is proportional to the size of the sensitivity. ε represents the privacy budget, and a smaller privacy budget means more noise, which better protects data privacy but also affects the accuracy of query results. Therefore, the choice of ε should be decided based on the actual situation, balancing privacy and accuracy.

After executing the local differential privacy algorithm, the noisy gradient is uploaded to the server for aggregation.

3.6 Experiment Design and Analysis

3.6.1 Experiment Setup

The relevant data and code for the experiment have been uploaded to Github[1] for sharing.

(1) Datasets: This thesis uses three datasets for the experiments. Two of them are public datasets, namely the European credit card dataset from Kaggle and the dataset from Github. The European credit card dataset ”creditcard.csv” is a collection of transaction records of European cardholders using credit cards. The dataset consists of 28 features and has 284,807 data points, of which 492 are fraudulent data. The dataset from Github records user rating rules and has 111,269 data points, with malicious rating data accounting for approximately 0.18% of the total data.

This thesis also uses a non-public dataset. The dataset is a collection of user transaction data provided by a bank, but due to a confidentiality agreement, its related information cannot be disclosed, such as the bank providing the data, data format, data attributes, and data features, as disclosing this information may leak user information. Nevertheless, this study hopes to use actual bank data as a sample to verify the security and accuracy of the proposed fraud detection method.

(2) Data Processing: In the three datasets used in this thesis, the fraudulent data accounts for only about 0.2% of the total samples, indicating that the sample data is highly imbalanced. In other words, if a model always predicts the sample as normal data, its accuracy can reach 0.998. However, such a model is meaningless because it cannot predict any fraudulent data. To solve the data imbalance problem, the training first adopts the IDW-SMOTE data preprocessing algorithm to make the number of fraudulent data and normal data samples roughly equivalent. Secondly, there are some features with large numerical variations in the dataset. If these features are not processed and directly used for training, the neural network may consider that the larger the feature value, the more important the record, which affects the accuracy of model training. Therefore, it is necessary to standardize the data to ensure that the importance of each feature is the same. Finally, before model training, the sample data needs to be divided into a training set and a test set. The training set is used to train the model, and the test set is used to verify the accuracy of the model.

(3) Evaluation Metrics: True Positive (TP): Correct prediction of positive class

[1]https://github.com/guozhengxin1999/credit-fraud-fd

samples as positive class. False Positive (FP): Incorrect prediction of negative class samples as positive class. False Negative (FN): Incorrect prediction of positive class samples as negative class. True Negative (TN): Correct prediction of negative class samples as negative class.

(i) Accuracy: Accuracy reflects the performance of the model in correctly classifying samples.

$$P_{\text{Accuracy}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{3-11}$$

(ii) Precision: Precision measures the accuracy of the model's positive predictions.

$$P_{\text{Precision}} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{3-12}$$

(iii) Recall: Recall measures the model's ability to detect actual positive cases.

$$P_{\text{Recall}} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3-13}$$

(iv) F1-score: F1-score is a comprehensive evaluation metric that combines precision and recall, and is the harmonic mean of precision and recall.

$$A_{\text{F}_1\text{-score}} = \frac{2 \cdot P_{\text{Precision}} \cdot P_{\text{Recall}}}{P_{\text{Precision}} + P_{\text{Recall}}}. \tag{3-14}$$

3.6.2 Experiment Result Analysis

3.6.2.1 Comparison of Data Preprocessing Algorithm Accuracy

To verify the effectiveness of data preprocessing algorithms in fraud detection, this thesis first compares the performance of IDW-SMOTE algorithm, SMOTE algorithm, and original data in fraud detection on the European credit card dataset, user dataset, and bank dataset.

As shown in Table 3-1, in the European credit card dataset, data preprocessing with IDW-SMOTE algorithm has a higher F1-score and better fraud detection performance compared to data preprocessing with SMOTE algorithm or using original data. Although the original data achieves high accuracy and recall, its precision and F1-score are relatively low. This is due to the imbalance of the data samples, where the number of samples in different categories differs greatly, causing the model to be biased towards the category with more samples, which in turn affects the precision and F1-score. Furthermore, the precision of the SMOTE algorithm is slightly higher than that of the IDW-SMOTE algorithm, indicating that the SMOTE algorithm has an advantage in detecting normal data in this dataset. Considering all evaluation metrics, the IDW-

SMOTE algorithm introduces adaptive weights and adjusts sample positions, making it the optimal choice for the European credit card dataset.

Table 3-1 Performance of Different Data Preprocessing Algorithms on the European Credit Card Dataset

	Accuracy	Precision	Recall	F1-score
IDW-SMOTE	0.981	0.851	0.991	0.916
SMOTE	0.906	0.925	0.906	0.915
Raw Data	0.999	0.620	0.999	0.765

As shown in Table 3-2, in the user dataset, data preprocessing with the IDW-SMOTE algorithm has the highest F1-score. The evaluation metric scores obtained from training with the original data have the same trend as those in the European credit card dataset. Considering all evaluation metrics, the IDW-SMOTE algorithm, which introduces the inverse distance weighting algorithm, has a clear advantage in the user dataset.

Table 3-2 Performance of Different Data Preprocessing Algorithms on the User Dataset

	Accuracy	Precision	Recall	F1-score
IDW-SMOTE	0.984	0.921	0.982	0.951
SMOTE	0.930	0.923	0.938	0.930
Raw Data	0.999	0.580	0.999	0.734

As shown in Table 3-3, in the non-public bank dataset, the IDW-SMOTE algorithm also has the best performance in terms of F1-score. The high accuracy and recall but low precision and F1-score obtained from training the original data are due to data imbalance, which is also observed in the two public datasets. Based on experiments on three datasets, considering all evaluation metrics, the IDW-SMOTE algorithm can better reflect sample distribution features and is the best choice for the bank dataset.

Table 3-3 Performance of Different Data Preprocessing Algorithms on the Bank Dataset

	Accuracy	Precision	Recall	F1-score
IDW-SMOTE	0.975	0.918	0.975	0.946
SMOTE	0.884	0.918	0.884	0.901
Raw Data	0.999	0.551	0.999	0.710

3.6.2.2 Fraud Detection Model Performance Comparison

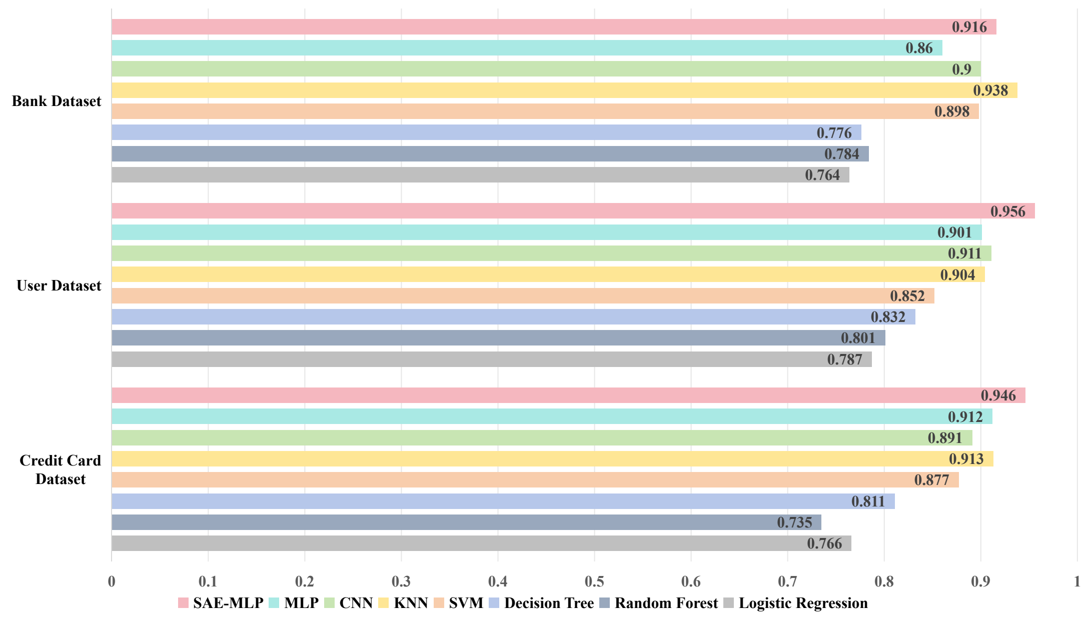


Figure 3-7 F1-score of Different Fraud Detection Models

To verify the accuracy of the classification results of the SAE-MLP model for fraud detection, this thesis conducted a horizontal comparison of various fraud detection models. The detection effects of SAE-MLP, MLP, CNN, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree, Random Forest, and Logistic Regression were compared on the European credit card dataset, user dataset, and bank dataset, in terms of their F1-score. As shown in Figure 3-7, SAE-MLP achieved relatively high F1-scores on all three datasets. On the European credit card dataset, KNN performed the best, followed by SAE-MLP, and Logistic Regression performed the worst, with F1-scores of 0.938, 0.916, and 0.764, respectively. On the user dataset, SAE-MLP performed the best, followed by CNN, and Logistic Regression performed the worst, with F1-scores of 0.956, 0.911, and 0.787, respectively. On the bank dataset, SAE-MLP performed the best, followed by KNN, and Random Forest performed the worst, with F1-scores of 0.946, 0.913, and 0.735, respectively. Overall, the SAE-MLP model has the ability to extract high-level features and effectively achieve classification tasks, outperforming other models on the user dataset and bank real dataset, and also showing good performance on the European credit card dataset.

3.6.2.3 Validation of Federated Learning Feasibility

In a cross-financial institution fraud detection system, federated learning can effectively protect the security and privacy of data. However, each client has an incomplete dataset. Therefore, by comparing the results of single machine training and federated learning training, it is possible to evaluate whether complete model training and accurate fraud data detection can be achieved while protecting data security and privacy when the dataset of each client is incomplete.

Table 3-4 Comparison of Single Machine and Federated Learning Training Results on Different Datasets

	Credit Card Dataset		User Dataset		Bank Dataset	
	Loss	F1-score	Loss	F1-score	Loss	F1-score
Single Machine	0.22	0.915	0.18	0.951	0.15	0.946
Federated Learning	0.32	0.912	0.22	0.950	0.20	0.942

As shown in Table 3-4, the loss of single machine and federated learning training using the credit card dataset is 0.22 and 0.32, respectively, and the F1-score is 0.915 and 0.912, respectively. The loss of single machine and federated learning training using the user dataset is 0.18 and 0.22, respectively, and the F1-score is 0.951 and 0.950, respectively. The loss of single machine and federated learning training using the bank dataset is 0.15 and 0.20, respectively, and the F1-score is 0.946 and 0.942, respectively. The results show that although the training effect of single machine training is better than that of federated learning training, the difference is very small. This proves that federated learning can maintain a training effect close to that of single machine training while ensuring the security of the data of the participating parties.

3.6.2.4 Verification of Blockchain Traceability

To verify the traceability of the blockchain off-chain storage model, this thesis compares partial data from the database and blockchain. Cn represents the client participating in federated learning. The fields in the database include data ID, data model, data hash value, and timestamp, while the fields in the blockchain include block sequence, hash value, version, and timestamp. The data ID is the primary key that uniquely identifies the data and distinguishes different model data. The data model is the specific model gradient. The data hash value is the hash value generated by the MD2 hash algorithm based on the model data, which is also stored in the blockchain. The timestamp is the specific time when the data is stored in the database. The block sequence is the position of the block in the blockchain. The hash value is the hash value of the model data stored in the blockchain. The version records the source and iteration number of the model data. The timestamp is the time when the block is generated.

As shown in Table 3-5, the following conclusions can be drawn by comparing the data in the database of client Cn with the data in the blockchain:

Table 3-5 Partial Storage Records of Database and Blockchain

	Dataset				Blockchain			
	Data ID	Data Model	Hash Value	Time-stamp	Block Sequence	Hash Value	Version	Time-stamp
C1	1	G_1	6b74804d9df 6ba597b161 eec4eae3047	16845 04800	3	6b74804d9df 6ba597b161 eec4eae3047	C1-1	16844 61615
C1	2	G_2	8350e5a3e24 c153df2275c 9f80692773	16844 62802	8	8350e5a3e24 c153df2275c 9f80692773	C1-2	16844 62820
C1	3	G_3	100ffcf37f db14a57fcb2 4f031c60cab	16844 63856	15	100ffcf37f db14a57fcb2 4f031c60cab	C1-3	16844 63873
C2	1	G_1	46b82b0321b 340162dbbf3 f8189100ab	16844 61648	4	46b82b0321b 340162dbbf3 f8189100ab	C2-1	16844 61661
C3	1	G_1	e5c8bbcf19 f47649ede7a c923666989f	16844 61591	2	e5c8bbcf19 f47649ede7a c923666989f	C3-1	16844 61603
C3	2	G_2	8e62c41e0d ddf09472b84 d7967448456	16844 62818	9	8e62c41e0d ddf09472b84 d7967448456	C3-2	16844 62836

(1) Data Integrity

By comparing the data hash value in the database with the hash value in the blockchain, it can be verified whether the hash value has been tampered with during storage or transmission, thereby judging the integrity and security of the data.

(2) Data Traceability

By viewing the version number in the blockchain, it can be traced which participant generated the block in which iteration.

(3) Data Generation Time Sequence

By comparing the timestamps in the database and blockchain, the time sequence of data generation can be determined, thereby understanding the update sequence of the data.

3.6.2.5 Validation of Differential Privacy Feasibility

According to the content in Section 3.5, adding Laplacian noise can protect the original gradients of the model, but it is necessary to balance the relationship between model accuracy and noise confidentiality. The parameters that affect Laplacian noise mainly include sensitivity and privacy budget, and appropriate parameters need to be selected to construct appropriate Laplacian noise to complete differential privacy encryption. In this experiment, the sensitivity was set to 1, and the following experiments were conducted on three datasets: first, comparing the F1-score under different privacy budgets; second, studying the impact of noise on the F1-score. As shown in Figure 3-8, the following two conclusions can be drawn:

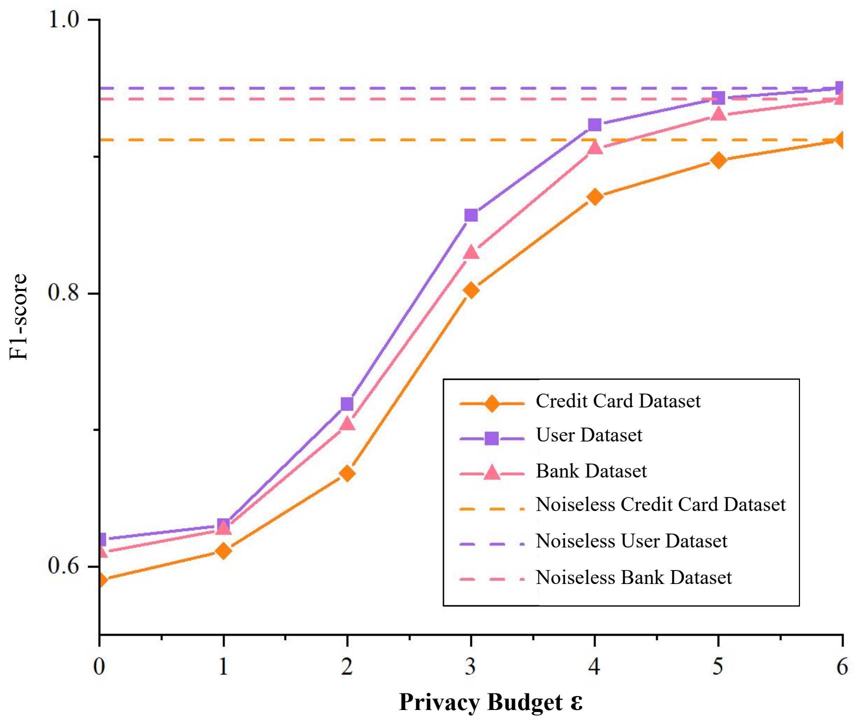


Figure 3-8 Relationship Between Laplacian Noise, Different Privacy Budgets, and F1-socre

(1) The F1-score of the model increases with the increase of the privacy budget. Eventually, on the credit card dataset, user dataset, and bank dataset, when the privacy budget $\varepsilon = 6$, the F1-score reaches 0.912, 0.950, and 0.942, respectively, which is equivalent to the performance level without noise.

(2) Increasing the privacy budget can reduce the impact of noise on the prediction results. An increase in the privacy budget means that less noise is allowed to be added, thereby reducing the disturbance of the prediction results. When the privacy budget increases from 2 to 3, the F1-score grows the fastest, and the impact of noise on accuracy begins to decrease, but the privacy protection effect also gradually decreases. To properly protect privacy, it is recommended to choose a privacy budget below 2, but the specific situation also needs to be determined based on the actual application and demand.

Through related experiments on differential privacy, the performance of the model under different privacy budgets can be evaluated, which is helpful for selecting appropriate privacy budgets and constructing appropriate Laplacian noise to achieve a balance between accuracy and privacy protection in differential privacy encryption.

3.7 Summary

This chapter proposes a federated learning fraud detection framework with differential privacy for collaborative fraud detection among financial institutions. The IDW-SMOTE algorithm is proposed to solve data imbalance, and stacked autoencoders and multi-layer perceptrons are used for fraud detection. Differential privacy algorithms are introduced to encrypt local models for secure communication. The results show that the IDW-SMOTE algorithm effectively solves the data imbalance problem and improves fraud detection accuracy. The SAE-MLP model outperforms other models in fraud detection. Federated learning is shown to maintain training effects while ensuring data security. Blockchain is shown to have traceability compared to databases. Appropriate noise parameters are found by adjusting the privacy budget.

4 Communication Cost Optimization in Federated Learning

As economic technology advances, financial institutions urgently need to undergo digital transformation, leading to an explosive growth in the amount of data they possess. In this context, multiple financial institutions collaborating on fraud detection often face communication bottleneck challenges. Therefore, this thesis proposes a strategy based on federated learning to reduce communication costs, optimizing communication costs from two perspectives: compressing model size and building an optimized aggregation time series.

4.1 Problem Description

In the application and execution process of federated learning, participants need to exchange data and model parameters to achieve collaborative learning. This process incurs certain communication costs, including data transfer time and energy consumption, as well as network bandwidth occupation. However, with the increase of data volume and number of participants, existing federated learning training methods can no longer meet the bandwidth, energy, and security requirements. Therefore, communication costs in large-scale federated learning have become a serious challenge that may cause a series of problems.

First, when there are many participants, the communication volume may become extremely large, causing bandwidth insufficiency problems. Transmitting large amounts of data and model parameters requires more network bandwidth, but existing network infrastructure may not be able to meet this demand, resulting in slow communication processes, increased latency, and even communication failures. Second, the communication process consumes the computing resources of devices. In large-scale federated learning, participants may be mobile devices or embedded devices with limited energy and computing power. Frequent communication and computation may cause energy depletion or insufficient computing power, and even lead to infrequent or incomplete communication between participants, thus reducing the performance and accuracy of the federated learning model. Furthermore, communication costs also involve data security and privacy issues. In federated learning, participants may exchange sensitive data and model parameters. If this information is stolen or leaked, it may cause serious consequences. Increased communication costs raise the risk of data transfer, which may result in data leaks or attacks. Therefore, reducing communication costs not only improves the performance and efficiency of federated learning but also helps ensure the security and privacy of the communication process, which is a very important task.

To address the challenge of communication costs, researchers usually adopt various methods to reduce communication costs. One method is to reduce the model size by compressing and simplifying model parameters to lower communication costs. This can be achieved through various techniques such as model compression, quantization, and sparse representation. By using these techniques, the amount of data transmitted between participants can be reduced, thereby lowering the required bandwidth and communication latency. Another method is to reduce communication rounds by decreasing the number of interactions between participants, improving communication efficiency, and thus reducing communication costs.

Based on the above background, this chapter proposes a strategy to reduce communication costs to improve the performance and efficiency of federated learning. To evaluate the effect of this strategy, this chapter proposes the following evaluation methods:

$$A \rightarrow (f, F, R), \tag{4-1}$$

where A represents the overall evaluation metric, f represents the loss function to measure the model's training effect, F represents the model's accuracy to measure its prediction effect, and R represents the number of communication rounds during federated learning training to measure the size of communication cost. This method can help comprehensively consider the model's training effect, prediction effect, and the size of communication cost, thus better evaluating the impact of the proposed communication cost optimization strategy on federated learning.

4.2 Overall

As shown in Figure 4-1, this thesis first introduces knowledge distillation algorithms in multi-layer perceptrons to achieve model compression and reduce communication costs. Second, on a mathematical level, the relationship between aggregation time series and loss functions is derived to improve the speed of aggregation completion, reduce communication rounds, and optimize communication costs. This chapter designs comparative experiments on public datasets of European credit cards and user ratings and non-public datasets of banks to verify the feasibility and effectiveness of knowledge distillation algorithms based on MLP and non-periodic aggregation algorithms.

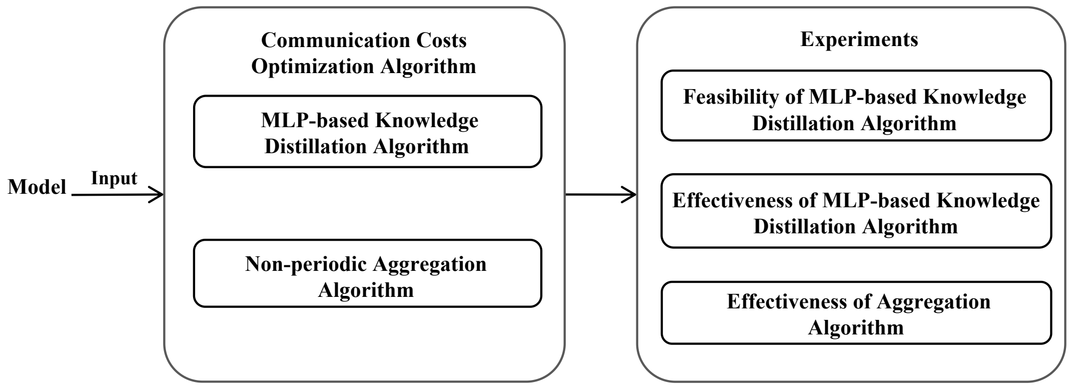

Model
Input
Communication Costs Optimization Algorithm
MLP-based Knowledge Distillation Algorithm
Non-periodic Aggregation Algorithm
Experiments
Feasibility of MLP-based Knowledge Distillation Algorithm
Effectiveness of MLP-based Knowledge Distillation Algorithm
Effectiveness of Aggregation Algorithm

Figure 4-1 Optimizing Communication Cost in Federated Learning

4.3 Knowledge Distillation based on Multi-layer Perceptron

Common model compression methods, such as parameter pruning, precision conversion, and neural architecture search, may destroy the original model's structure during the compression process, resulting in the loss of knowledge learned by the original model. To solve this problem, knowledge distillation algorithms have been proposed. Knowledge distillation transfers the knowledge of the original model to the compressed model through a distillation process, thus preserving the model's knowledge. Although the compressed model is much smaller than the original model, it can still maintain similar performance to the original model. In the knowledge distillation process, the original model is referred to as the teacher network, while the compressed model is the student network. Typically, the teacher network is much larger than the student network.

Chapter 3 proposes a fraud detection model based on a multi-layer perceptron. To reduce communication costs, this thesis introduces knowledge distillation algorithms for model compression. The knowledge distillation algorithm is applied to the client-side of federated learning, where the input data is pre-trained on a complex teacher network, and the trained knowledge is transferred to the student network, enabling the student network to achieve the training effect of the teacher network. Through this approach, the fraud detection accuracy is maintained while achieving model compression and reducing communication costs.

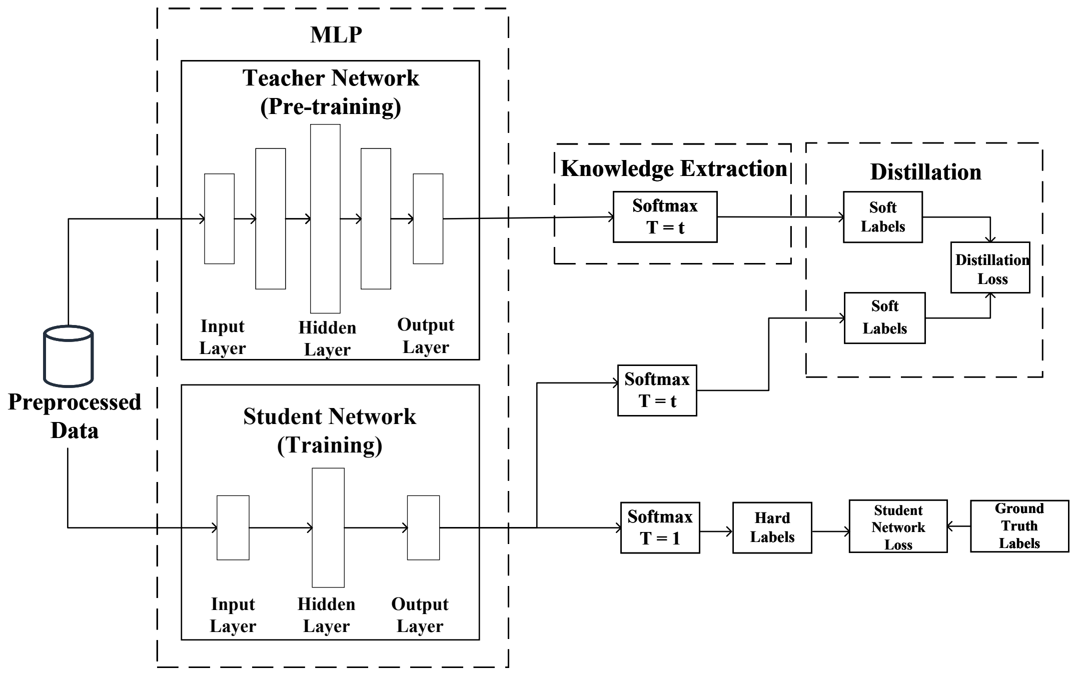


Figure 4-2 Illustration of Knowledge Distillation based on Multi-layer Perceptron

The details of the knowledge distillation algorithm based on MLP are shown in Figure 4-2 and mainly consist of the following components:

(1) Preprocessing data: The preprocessed data is the feature data obtained by training a stacked autoencoder. These feature data are used as input data for the multi-layer perceptron module to train and compress the fraud detection model.

(2) Multi-layer perceptron module: The multi-layer perceptron module consists of a teacher network and a student network, both of which are perceptron models containing input layers, hidden layers, and output layers. The teacher network has a more complex structure (as shown in Figure 4-2, the teacher network has a five-layer structure), while the student network has a simpler structure (as shown in Figure 4-2, the student network has a three-layer structure). The teacher network needs to be pre-trained to extract knowledge through distillation for the student network to learn.

(3) Softmax layer: The Softmax layer is used to convert the classification results into probability distributions, indicating the probability of a sample belonging to each category. The higher the probability, the higher the likelihood that the sample belongs to that category, and the sum of all category probabilities is 1. In the knowledge distillation algorithm, the temperature **T** of the Softmax function is an important hyper-parameter used to adjust the probability distribution of the Softmax function. A higher temperature value can make the output of the Softmax function smoother and the prob-

ability distribution more uniform. In contrast, a lower temperature will make the output of the Softmax function sharper and the probability distribution more concentrated. In other words, a higher temperature can help the model make more balanced predictions for samples with higher uncertainty, making the probability distribution more uniform across different categories.

(4) Hard labels: When the temperature parameter is 1, the Softmax layer generates hard labels, which provide discrete category information. The result is a 0-1 distribution, where the label corresponding to the category with the highest probability is 1, and the labels for all other categories are 0. Hard labels provide clear category predictions and are suitable for tasks requiring discrete classification.

(5) Soft labels: By adjusting the temperature parameter, the perceptron model can generate soft labels through the Softmax layer. Soft labels are a probability distribution that provides richer information than hard labels and reflects the confidence of the sample in each category. The use of soft labels can improve the generalization ability and robustness of the student network, enabling it to more accurately predict the probability of a new sample belonging to each category.

(6) Real labels: The real labels are the actual classification results of the training set. The real labels are determined by manual annotation or known data sources and provide accurate category information for training the model.

(7) Loss function: The loss function of the knowledge distillation module includes the loss function of the student network when the temperature is 1 and the distillation loss function.

The execution process of the MLP-based knowledge distillation algorithm is as follows:

Step 1: Train the MLP teacher network.

Input the feature-extracted data into the teacher network for pre-training and obtain the soft labels of the teacher network through the Softmax layer. Softmax is crucial in knowledge distillation because the output of the teacher network should be 0-1 classification results, but such output is not distinguishable from the hard labels output by the student network, which would result in a small contribution of the teacher network to the distillation loss. Therefore, the Softmax layer is introduced to increase the contribution of the teacher network's loss to the distillation loss. Moreover, the trained model may exhibit overconfidence, i.e., the model is overly confident in its prediction results, which may lead to classification errors. To solve this problem, temperature **T** is introduced in Softmax. When the difference between two logits (probabilities before entering

Softmax) is large, the temperature parameter **T** can maintain the difference between the two outputs, thus preserving the model’s knowledge. Therefore, the Softmax function highlights the correct category probability in a milder way, and temperature **T** turns the ”softness” of Softmax into a tunable hyperparameter. The probability result output by the teacher network through the Softmax layer is as follows:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp\left(z_j/T\right)}, \tag{4-2}$$

where q represents the probability output of the teacher network, **T** represents the temperature, z_i is the original value of each category involved in the classification, and z_j is the original value of project j’s classification.

Step 2: Create an MLP student network.

Create a simple student network as the target model for knowledge distillation compression.

Step 3: Define the loss function.

In knowledge distillation, the loss function consists of both the distillation loss and the student network’s loss. The distillation loss is used to quantify the difference between the teacher network and the student network, while the student network’s loss is defined as the loss function in the traditional supervised learning framework. Considering both parts of the loss function, the knowledge distillation loss function can be represented as:

$$L = \alpha L_{\text{soft}} + (1-\alpha) L_{\text{hard}}, \tag{4-3}$$

where α is a weight value ranging from 0 to 1. L_{soft} is the distillation loss, which is the cross-entropy between the soft labels of the teacher network’s Softmax output and the student network’s Softmax output at the same temperature. L_{hard} is the loss related to the hard labels when the student network is at $\mathbf{T} = 1$. The calculation formula for L_{soft} is as follows:

$$L_{\text{soft}} = \mathrm{CE(q, p)} \cdot \mathrm{T}^2, \tag{4-4}$$

where CE stands for cross-entropy, p is the probability output of the student network at temperature $\mathbf{T} = \mathrm{t}$, and q is the probability output of the teacher network at temperature $\mathbf{T} = \mathrm{t}$. L_{hard} is the cross-entropy between the hard label of the student network’s Softmax output at $\mathbf{T} = 1$ and the true label. The calculation formula for L_{hard} is as follows:

$$L_{\text{hard}} = \mathrm{CE(y, p)}, \tag{4-5}$$

where CE stands for cross-entropy, y is the true label, and p is the predicted value of the student network when $\mathbf{T} = 1$. L_{hard} is necessary because the teacher network also has a certain error rate, and using the true label can effectively reduce the possibility of errors being transmitted to the student network.

Step 4: Perform Distillation Training.

The goal of distillation training is to optimize the parameters of the student network by minimizing the loss function of knowledge distillation so that it can learn from the knowledge of the teacher network and improve its performance and generalization ability. Specifically, the training of the student network includes two steps. First, the student network uses the same temperature as the teacher network to obtain soft labels through the Softmax layer and combines them with the soft labels of the teacher network to calculate the distillation loss. Second, the student network sets the temperature parameter to 1 to obtain hard labels through the Softmax layer and combines them with the true labels to calculate the student network loss.

Step 5: Inference and Application.

After the distillation training is completed, the student network that has undergone distillation training (distillation network) can be compared and verified with the teacher network and the original student network. By comparing their loss functions and accuracy in classification training, the effectiveness of the multi-layer perceptron based on knowledge distillation is verified, and the performance and improvement of the distillation network are evaluated.

Thus, in the knowledge distillation module based on multi-layer perceptron, a student model with a simple structure but accurate classification can be obtained. In federated learning, exchanging such student models can effectively reduce communication costs. Furthermore, deploying and applying such student models in resource-limited environments will be more convenient. Therefore, the knowledge distillation algorithm not only can improve the performance of the student network but also can bring advantages in model size and communication aspects.

4.4 Non-periodic Aggregation in Federated Learning

Federated learning aggregation algorithm enables data sharing while protecting user data privacy. Choosing the right algorithm improves model training efficiency and reduces communication costs. This section derives the mathematical relationship between the aggregation time series and loss function to find an optimized aggregation time series, speed up federated learning aggregation, reduce model exchanges, and lower communication costs. Figure 4-3 shows the construction process: modeling and loss function construction, mathematical boundary derivation, and optimized aggregation time series construction. This leads to a more effective algorithm and efficient federated learning aggregation process.

Constructing an Aggregated Loss Function
Deriving Bounds for the Loss Function
Constructing Optimized Aggregated Time Series

Figure 4-3 Derivation of Aggregation Time Series

4.4.1 Loss Function of Aggregation

In the entire lifecycle of federated learning, communication overhead mainly comes from the exchange of model parameters between the server and the client. Reducing the number of communication rounds can lower the overall communication overhead. By adopting appropriate aggregation algorithms, federated learning can achieve low-loss and high-accuracy training results in fewer communication rounds, thereby reducing communication overhead.

FedAvg and FedProx are two classic federated learning aggregation algorithms. FedAvg is a simple periodic algorithm that aggregates model parameters through weighted averaging. FedProx handles model heterogeneity problems by introducing a proximal term in the loss function. However, both are periodic and may not fully utilize client's local data and resources, slowing down model convergence. Thus, they may not be optimal.

Intuitively, for non-independent and identically distributed data, if too many local iterations are performed in federated learning, it may lead to an increase in the differences between client models, which in turn may negatively affect the performance of the aggregated model. Therefore, in the early stages of training, since the initial model is inaccurate, aggregation may not be meaningful and may even introduce errors. However, in the later stages of federated training, the model enters the fine-tuning stage, and the server should increase the frequency of aggregation to reduce the differences between local models. By increasing the frequency of aggregation, these differences can be merged more quickly, thereby improving the performance of the overall model.

Therefore, to reduce communication overhead, a non-periodic aggregation strategy can be adopted. In the early stages of training, the frequency of aggregation can be reduced, allowing clients to perform more local iterations to reduce differences between local models. In the later stages of training, the frequency of aggregation can be increased to achieve the optimal aggregated model more quickly.

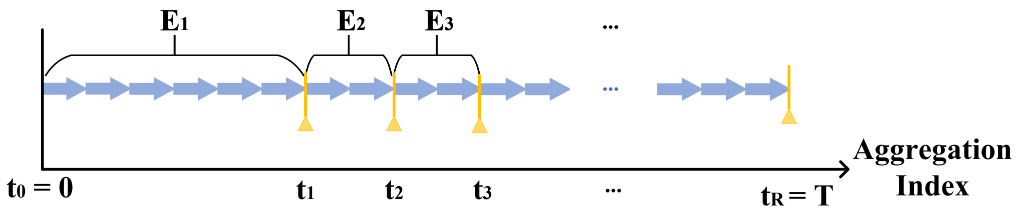


Figure 4-4 Illustration of Aggregation Time Series in Federated Learning

The federated learning aggregation time series is shown in Figure 4-4. Let E_1, E_2, ..., E_R be the number of local iterations performed by the client for each model upload to the server, where R is the total number of model exchanges. Let $\mathbf{T}$ be the total number of iterations performed by the client, then $\mathbf{T} = E_1 + E_2 + ... + E_R$. The time of the i-th model upload, i.e., the time of the i-th aggregation, is $t_i = E_1 + E_2 + ... + E_i$, where $t_0 = 0$, $i > 0$, and i is an integer. In other words, at time t_{i-1}, the client begins to prepare for the i-th aggregation, and after E_i local iterations, completes the training and uploading of the local model at time t_i, which is also the time of aggregation on the server. Define $\mathbf{I_T}$ as the aggregation index and $\mathbf{D_T}$ as the set of the number of local iterations performed by the client between two aggregations. Then, $\mathbf{I_T} = \{t_0, t_1, \cdots, t_R\}$, $\mathbf{D_T} = \{E_1, E_2, \cdots, E_R\}$. Since it is difficult to find the minimum number of communication rounds, the problem of reducing communication costs can be transformed into finding the loss function with the fastest convergence rate while maintaining fraud detection accuracy, thereby achieving the goal of reducing communication rounds and lowering communication costs. The constructed model is as follows:

$$\begin{aligned} &\arg\min \quad f(x) \\ &\mathbf{I_T} = \{t_0, t_1, \cdots, t_R\} \\ &\text{s.t.} \quad 0 = t_0 < t_1 \cdots < t_R = T, \end{aligned} \tag{4-6}$$

where f(x) is the loss function.

4.4.2 Boundary Derivation of Loss Function

This section will derive the mathematical relationship between the loss function and the aggregation time series $\mathbf{I_T}$. The derivation is based on the following assumption:

Assumption 1 (L-smoothness): For any x and y, the loss function $f(x)$ satisfies the definition of Lipschitz continuity:

$$f(\mathrm{y}) \leq f(\mathrm{x}) + \langle \nabla f(\mathrm{x}), \mathrm{y} - \mathrm{x} \rangle + \frac{L}{2} \|\mathrm{y} - \mathrm{x}\|^2. \tag{4-7}$$

The notation ∇f represents the gradient. The L-smoothness assumption defines the upper bound of a second-order function and the maximum rate of change of the gradient. Changing the direction of the inequality obtained in L-smoothness and replacing L with μ leads to the property of strongly convex functions.

Assumption 2 (μ-strong convexity): For any x and y, the loss function $f(x)$ satisfies the following inequality:

$$f(\mathrm{y}) \geq f(\mathrm{x}) + \langle \nabla f(\mathrm{x}), \mathrm{y} - \mathrm{x} \rangle + \frac{\mu}{2} \|\mathrm{y} - \mathrm{x}\|^2. \tag{4-8}$$

The μ-strongly convex assumption defines the lower bound of a second-order function and the minimum rate of change of the gradient. The above two assumptions impose some requirements on the basic properties of the loss function $f(x)$: the rate of change of the function is neither too fast (L-smooth) nor too slow (μ-strongly convex).

Assumption 3 (Bounded Variance of Stochastic Gradients): When each client performs stochastic gradient descent, the variance of the stochastic gradient with uniformly sampled samples is bounded by σ^2:

$$\mathbb{E} \|\nabla f_i (x; \xi_i) - \nabla f_i(x)\|^2 \leq \sigma^2, \tag{4-9}$$

Where ξ_i represents a randomly selected sample from the local data.

Assumption 4 (Bounded Stochastic Gradients): The stochastic gradients of each client are also bounded by $\mathbf{G}^2$:

$$\mathbb{E} \|\nabla f_i (x; \xi_i)\|^2 \leq G^2. \tag{4-10}$$

This chapter requires Lemma 1 [80] and Lemma 2 [80] to clarify the direction of change in stochastic gradient descent. Lemma 1 and Lemma 2 are as follows:

Lemma 1: Based on Assumption 1 and Assumption 2, if $\eta \leq \frac{1}{4L}$, then:

$$\begin{aligned} \mathbb{E} \left\|\overline{\boldsymbol{x}}^{\mathrm{t}+1} - \boldsymbol{x}^*\right\|^2 \leq (1 - \eta\mu) \mathbb{E} \left\|\overline{\boldsymbol{x}}^{\mathrm{t}} - \boldsymbol{x}^*\right\|^2 + \eta^2 \sum_{i=1}^{N} p_i^2 \sigma_i^2 + 6\,\mathrm{L}\eta^2 \Gamma \\ + 2\mathbb{E} \left[\sum_{i=1}^{N} p_i \left\|\overline{\boldsymbol{x}}^{\mathrm{t}} - \boldsymbol{x}_i^{\mathrm{t}}\right\|^2 \right], \end{aligned} \tag{4-11}$$

Where x^{t} represents the client parameters at time t, $\overline{\boldsymbol{x}}^{\mathrm{t}}$ represents the average parameters at time t, $\overline{\boldsymbol{x}}^{\mathrm{t}+1}$ represents the average parameters after one round of SGD update, $\boldsymbol{x}^*$ represents the value of x when the loss function $f(x)$ takes the optimal result, η represents the learning rate, μ represents the parameter of the strongly convex function, $\sum_{i=1}^{N} p_i^2\sigma_i^2$ represents the variation of the gradient[80], $\Gamma = f^* - \sum_{i=1}^{N} p_i f_i^* \geqslant 0$, f^* represents the optimal loss function, and p_i represents the weight of each client in federated learning.

Lemma 2: Based on assumption 4, for any $t = t_i + \varepsilon$, where $t_i \in \mathbf{I_T} = \{t_0, t_1, \cdots, t_R\}$, we have:

$$\mathbb{E}\left[\sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{p_i} \left\|\overline{\boldsymbol{x}}^{\mathrm{t}} - \boldsymbol{x}_{\mathrm{i}}^{\mathrm{t}}\right\|^2\right] \leq 4\eta^2\varepsilon^2G^2, \tag{4-12}$$

Where p_i represents the weight of each client in federated learning, x^{t} represents the client parameters at time t, $\overline{\boldsymbol{x}}^{\mathrm{t}}$ represents the average parameters at time t, η represents the learning rate, and G^2 is the boundary value of the stochastic gradient.

Through Lemma 1 and Lemma 2, this chapter aims to predict the distance between the client average parameters $\overline{\boldsymbol{x}}^{\mathrm{t+1}}$ after one random gradient descent and $\boldsymbol{x}^*$ at the optimal loss function based on all information at time t. The information at time t includes:

(1) The distance between the client average parameters $\overline{\boldsymbol{x}}^{\mathrm{t}}$ and $\boldsymbol{x}^*$ before random gradient descent;

(2) The variance of the update direction of one random gradient descent;

(3) The heterogeneity Γ caused by the client dataset and local model differences;

(4) The variance of each client parameter at time t.

In addition, if the aggregation at each moment in the server uses the FedAvg algorithm, then the change of the global model parameters is actually the process of $\overline{\boldsymbol{x}}^{\mathrm{t}}$ evolving along the gradient direction to $\overline{\boldsymbol{x}}^{\mathrm{t+1}}$. Although Lemma 1 was proved under strong convexity conditions in the original paper, it can be easily proved that the lemma still holds under weak convexity conditions.

Based on assumptions 1 to 4, Lemma 1 and Lemma 2, Theorem 1 can be obtained.

Theorem 1: If assumptions 1 to 4 hold, and $0 < \eta < \min[\frac{1}{\mu}, \frac{1}{4\mathbb{L}}]$, then after R rounds of aggregation in the server and T iterations in the client, the upper bound of the loss function $f(x)$ satisfies the following inequality:

$$\begin{aligned} \mathbb{E}\left[f\left(\bar{x}^{T+1}\right)\right] \leq & f(x^*) + \frac{L}{2}[(1-\eta\mu)^T\, \mathbb{E}\left\|\overline{\boldsymbol{x}}^0 - \boldsymbol{x}^*\right\|^2 \\ & + 8\eta^2G^2 \sum_{i=0}^{R}\sum_{\varepsilon=0}^{E_{i+1}} (1-\eta\mu)^{(T-t_i-\varepsilon)}\varepsilon^2 \\ & + \frac{\eta\sum_{i=1}^{N} p_i^2\sigma_i^2 + 6\,\mathrm{L}\eta\Gamma}{\mu}], \end{aligned} \tag{4-13}$$

where $\overline{\boldsymbol{x}}^{\mathrm{T+1}}$ represents the average parameter at time $T+1$, $\boldsymbol{x}^*$ represents the value of x when the loss function $f(x)$ takes the optimal result, η represents the learning rate,

μ represents the parameter of the strongly convex function, $\overline{\boldsymbol{x}}^0$ represents the average parameter at time $t = 0$, G^2 is the boundary value of the stochastic gradient, R is the current number of aggregation rounds, $E_i \in \mathbf{D_T} = \{E_1, E_2, \cdots, E_R\}$, $t_i \in \mathbf{I_T} = \{t_0, t_1, \cdots, t_R\}$, $\Gamma = f^* - \sum_{i=1}^{N} p_i f_i^* \geqslant 0$, f^* represents the optimal loss function, and p_i represents the weight of each client in federated learning.

Proof: According to Lemma 1, assuming $\Delta\_t = \mathbb{E}\left\|\overline{\boldsymbol{x}}^{\mathrm{t}} - \boldsymbol{x}^*\right\|^2$, we have:

$$\begin{aligned} \Delta_{t+1} &\leq (1 - \eta\mu)\,\Delta_t + \mathrm{A}_t + \mathrm{B} \\ \mathrm{A} &= 2\mathbb{E}\left[\sum_{i=1}^{N} p_i \left\|\overline{\boldsymbol{x}}^{\mathrm{t}} - \boldsymbol{x}_i^{\mathrm{t}}\right\|^2\right] \\ \mathrm{B} &= \eta^2 \sum_{i=1}^{N} p_i^2 \sigma_i^2 + 6\,\mathrm{L}\eta^2\Gamma. \end{aligned} \tag{4-14}$$

After recursion, the following formula is obtained:

$$\begin{aligned} \Delta_{t+1} \leq (1 - \eta\mu)^{t+1}\,\Delta_0 &+ \sum_{i=0}^{t} (1 - \eta\mu)^{(t-i)} \mathrm{A}_i \\ &+ \mathrm{B} \sum_{i=0}^{t} (1 - \eta\mu)^{(i-1)}. \end{aligned} \tag{4-15}$$

If $\eta \leq \frac{1}{\mu}$, then:

$$\Delta_{t+1} \leq (1 - \eta\mu)^{t+1}\,\Delta_0 + \sum_{i=0}^{t} (1 - \eta\mu)^{(t-i)} \mathrm{A}_i + \frac{\mathrm{B}}{\eta\mu}. \tag{4-16}$$

Based on assumption 1 and Jensen's inequality: $\mathbb{E}(f(x)) \geq f(\mathbb{E}(x))$, it can be derived that:

$$\begin{aligned} \mathbb{E}\left[f\left(\bar{x}^{t+1}\right)\right] &\leq f(x^*) + \frac{L}{2}\Delta_{t+1} \\ &\leq f(x^*) + \frac{L}{2}[(1 - \eta\mu)^{t+1}\,\Delta_0 \\ &+ \sum_{i=0}^{t} (1 - \eta\mu)^{(t-i)} \mathbf{A}_i + \frac{\mathbf{B}}{\eta\mu}]. \end{aligned} \tag{4-17}$$

Combining inequality 4-14, it can be found that the parameter $\mathbf{A}_i$ is related to the time series $\mathbf{I_T}$, while the parameter $\mathbf{B}$ is a constant value. Therefore, it is necessary to derive the relationship between the loss function and $\mathbf{I_T}$ using inequalities 4-12 and 4-17. However, t has different meanings in these two inequalities, so it is necessary to unify the parameter meanings of the two inequalities. In inequality 4-17, when $t_z \in \mathbf{I_T} = \{t_0, t_1, \cdots, t_R\}$, for any $t = t_z$, $i = t_j + \varepsilon$, we have:

$$\begin{aligned}
&\sum_{i=0}^{t}(1-\eta\mu)^{(t-i)}\mathbf{A}_i \\
=&\sum_{t_j+\varepsilon=0}^{t_z}(1-\eta\mu)^{(t_z-t_j-\varepsilon)}\mathbf{A}_{t_j+\varepsilon} \\
=&\sum_{j=0}^{z}\sum_{\varepsilon=0}^{t_{j+1}-t_j}(1-\eta\mu)^{(t_z-t_j-\varepsilon)}\mathbf{A}_{t_j+\varepsilon} \\
\leq& 8\eta^2G^2\sum_{j=0}^{z}\sum_{\varepsilon=0}^{E_{j+1}}(1-\eta\mu)^{(t_z-t_j-\varepsilon)}\varepsilon^2
\end{aligned}, \tag{4-18}$$

where, according to Figure 4-4, $E_{j+1} = t_{j+1} - t_j$. From this, the mathematical relationship between the loss function and the time series $\mathbf{I_T}$ is derived. Therefore, the model 4-6 can be transformed into:

$$\begin{aligned}
&\arg\min \quad \mathrm{C}(\mathbf{I_T}) = \sum_{i=0}^{R}\sum_{\varepsilon=0}^{E_{i+1}}(1-\eta\mu)^{(T-t_i-\varepsilon)}\varepsilon^2 \\
&\mathbf{I_T} = \{t_0, t_1, \cdots, t_R\} \\
&\text{s.t.} \quad 0 = t_0 < t_1 \cdots < t_R = T.
\end{aligned} \tag{4-19}$$

In order to reduce communication costs and loss, this thesis should choose an appropriate $\mathbf{I_T}$ to obtain a better C.

4.4.3 Construction of Aggregation Time Series

To minimize loss and reduce communication costs, this thesis needs to construct new aggregation time series, so understanding the properties of the time interval E_i between two adjacent aggregations is crucial. This thesis needs to use Lemma 3[81] and Lemma 4[81] to help construct $\mathbf{I_T}$ and $\mathbf{D_T}$. The two lemmas are as follows:

Lemma 3: For $\mathbf{I_T} = \{t_0, t_1, \cdots, t_R\}$ and $\mathbf{D_T} = \{E_0, E_1, \cdots, E_R\}$, when the loss function $f(x)$ takes the optimal solution, the aggregation time series $\mathbf{I_T}$ should satisfy the following inequality:

$$E_1 > E_1 > \cdots > E_R. \tag{4-20}$$

In other words, as the number of aggregation rounds increases, the number of local iterations should gradually decrease. Therefore, periodic aggregation is not optimal.

Lemma 4: For $\mathbf{I_T} = \{t_0, t_1, \cdots, t_R\}$ and $\mathbf{D_T} = \{E_0, E_1, \cdots, E_R\}$, if $\eta\mu$ is sufficiently small such that $(1-\eta\mu)^{-E_i} \approx 1$ and $\Delta_i \ll E_i$, where $\Delta_i = E_i - E_{i-1}$, then:

$$\Delta_i \approx \Delta_j (= \Delta), \tag{4-21}$$

where $i, j \in [1,2,\cdots,R]$. This is a very concise property, and if the optimal loss function is to be obtained, the sequence $\mathbf{D_T}$ should be constructed as an arithmetic sequence with a common difference of $d_D = -\Delta$.

Based on Lemma 3 and Lemma 4, the sequence $\mathbf{D_T}$ and the sequence $\mathbf{I_T}$ can be transformed into:

$$\begin{aligned} \mathbf{D_T} &= \{E_1, E_1 + d_D, \cdots, E_1 + (R-1)d_D\}, \\ \mathbf{I_T} &= \left\{0, E_1, \cdots, E_1 \cdot i + d_D \frac{i(i-1)}{2}, \cdots, T\right\}. \end{aligned} \tag{4-22}$$

Furthermore, based on the properties of arithmetic sequences, it can be obtained that:

$$\frac{E_1 + E_R}{2} = \frac{T}{R}. \tag{4-23}$$

If $\overline{E} = T/R$, then:

$$d_D = \frac{2(\overline{E} - E_1)}{R-1}. \tag{4-24}$$

Therefore, it can be concluded that both the sequence $\mathbf{D_T}$ and the sequence $\mathbf{I_T}$ are only related to E_1.

Theorem 2: If $\overline{E}$ and R are fixed values, to obtain a better loss function, the sequence $\mathbf{I_T}$ needs to take the optimal solution, then E_1 needs to satisfy:

$$\overline{E} < E_1 < \frac{2R-2+2Z}{R-1+2Z} \cdot \overline{E}, \tag{4-25}$$

where $Z \gg 1$。

Proof: According to Lemma 4, if $\Delta \ll E_i$, let $Z \cdot \Delta < minE_i = E_R$, then:

$$\begin{aligned} -Z \cdot d_D &< E_1 + (R-1)d_D \\ -Z \cdot \frac{2(\overline{E} - E_1)}{R-1} &< E_1 + (R-1) \cdot \frac{2(\overline{E} - E_1)}{R-1} \\ E_1 &< \frac{2R-2+2Z}{R-1+2Z} \cdot \overline{E}. \end{aligned} \tag{4-26}$$

Therefore, it can be determined that, in the case where R and T are fixed values, by setting an aggregation time series that satisfies 4-25, a better loss function can be obtained to reduce the number of aggregation rounds and lower communication costs.

4.5 Experiment Design and Analysis

4.5.1 Experiment Setup

The data and code have been uploaded to Github[1] for sharing.

[1]https://github.com/guozhengxin1999/credit-fraud-fd

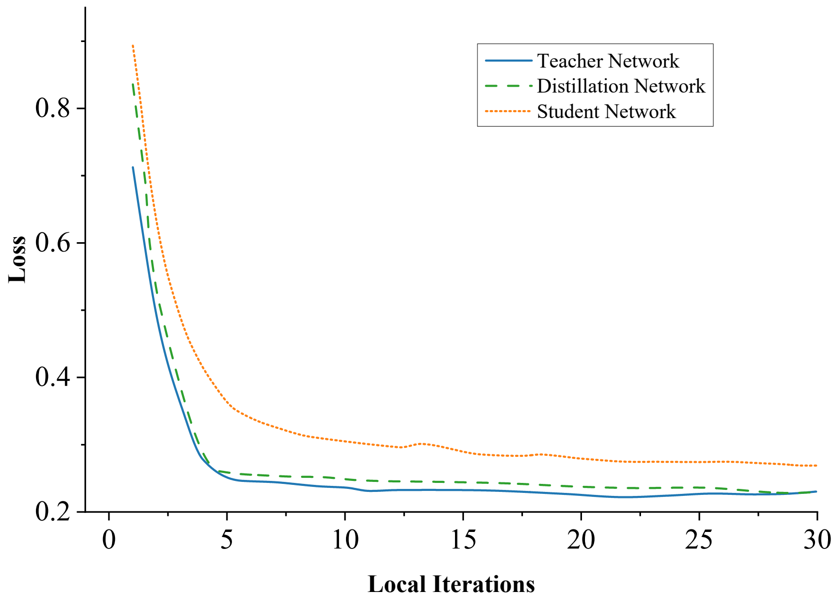
(1) Datasets: As in Section 3.6.1, this chapter uses three datasets to verify the effectiveness of the knowledge distillation algorithm based on multi-layer perceptrons and the feasibility of the non-periodic aggregation algorithm. Among them, the two public datasets are from Kaggle's European credit card dataset and Github's user rating dataset, while the non-public dataset comes from a bank's user dataset. By using these different types of datasets, the performance and effects of the algorithms proposed in this chapter can be comprehensively evaluated in different fields and tasks. At the same time, this chapter also pays attention to ensuring that the use of non-public datasets complies with relevant privacy and security regulations to protect the privacy and confidentiality of user data.

(2) Experimental environment: For the knowledge distillation algorithm based on MLP and non-periodic aggregation algorithm, this chapter builds different experimental environments. To verify the effectiveness of the knowledge distillation algorithm, experiments are conducted on a single client using the complete dataset, include comparing the loss function and F1-score of the teacher, distillation, and student networks, and evaluating the compression effect of the knowledge distillation algorithm, model quantization, and model pruning. To verify the feasibility of the non-periodic algorithm, a federated learning environment is built, include 5 client machines and 1 server machine. The entire dataset is divided into 5 parts, each with the same proportion of normal and fraud data, and allocated to the 5 clients for training. The privacy budget is set to 6 to reduce the impact of noise on accuracy. To verify the effect of the non-periodic aggregation algorithm, params are assumed: R=40, $\overline{E}$=$\frac{T}{R}$=30, E_1=40, d_D=-1, Z=10. The aggregation time sequence $\mathbf{I}$ and local iteration sequence $\mathbf{D_T}$ are constructed. The effect and performance of the non-periodic aggregation algorithm in the federated learning environment can be evaluated through these settings.

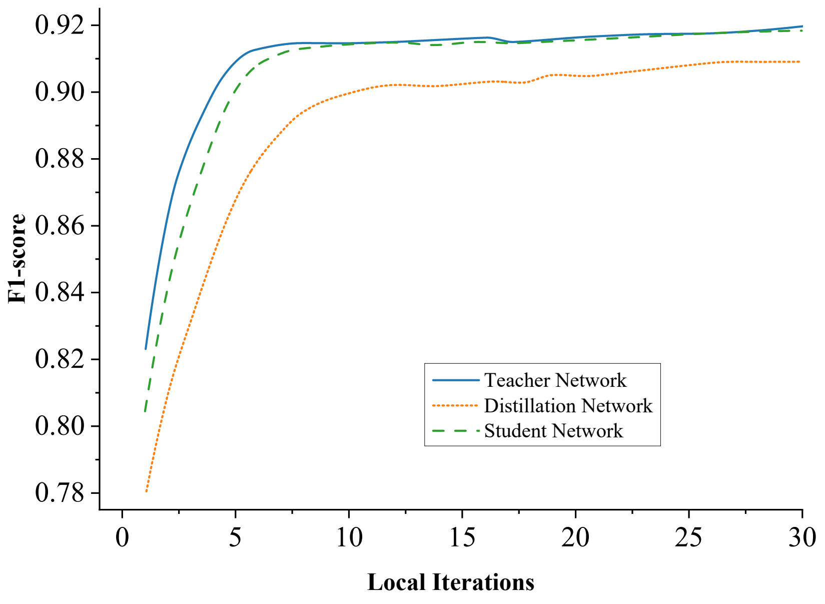
4.5.2 Experiment Result Analysis

4.5.2.1 Feasibility of Knowledge Distillation Algorithm based on MLP

This chapter first verified the feasibility of the knowledge distillation algorithm based on MLP. Experiments were conducted on a single client using credit card datasets, user datasets, and bank datasets. By comparing the loss functions and F1-scores of the teacher network, student network, and distilled network, the compression effect of the knowledge distillation based on MLP was evaluated and analyzed.



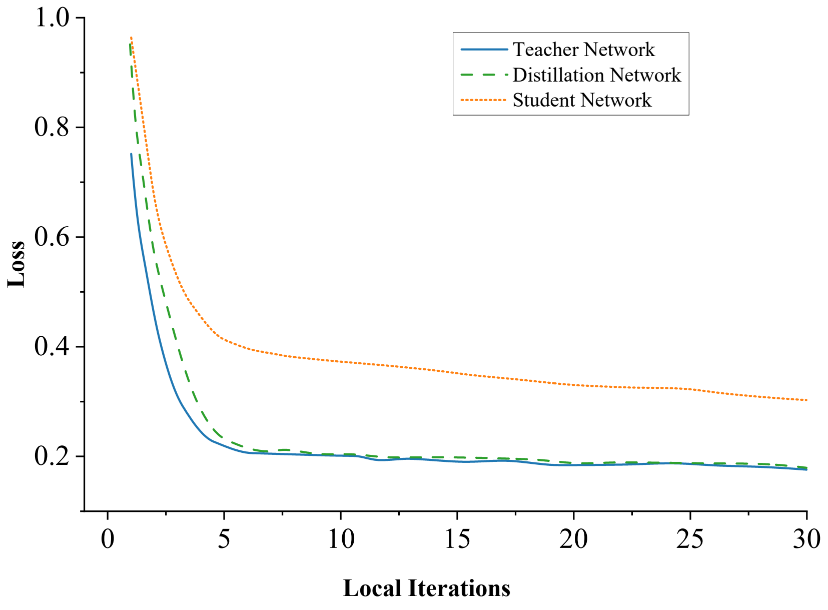
(a) Loss on Credit Card Dataset



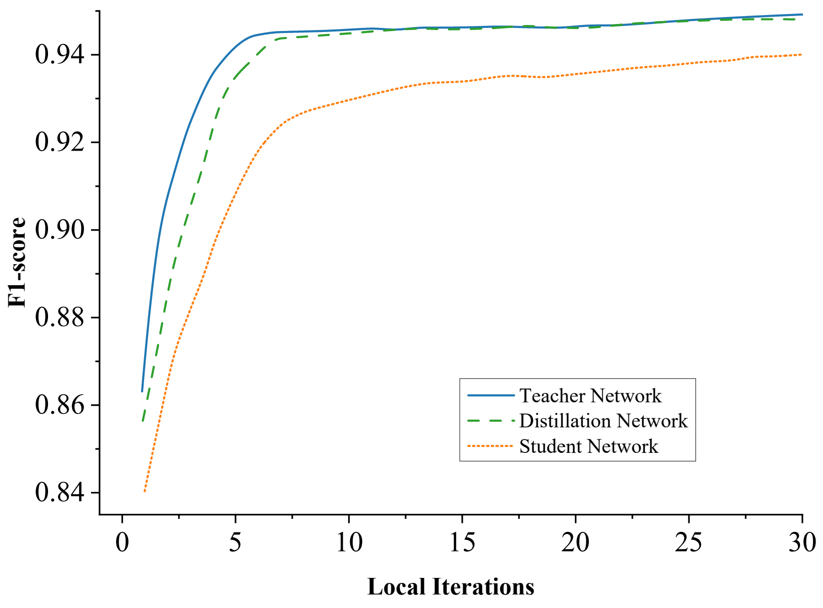
(b) F1-score on Credit Card Dataset

Figure 4-5 Effectiveness of Knowledge Distillation Algorithm based on MLP on Credit Card Dataset

As shown in Figure 4-5, experiments were conducted using the credit card dataset. In the initial stage of training, the loss value and F1-score of the distillation network were between those of the teacher network and the student network. After about 5 iterations of training, the loss and F1-score curves of the teacher network and the distillation network became stable, while the student network required about 8 local iterations to reach a stable state. After 5 local iterations, the training effect of the distillation network significantly approached that of the teacher network and was significantly better than that of the student network. Ultimately, the loss of the teacher network, distillation network, and student network were 0.22, 0.24, and 0.28, respectively, and the F1-score were 0.916, 0.915, and 0.908, respectively.



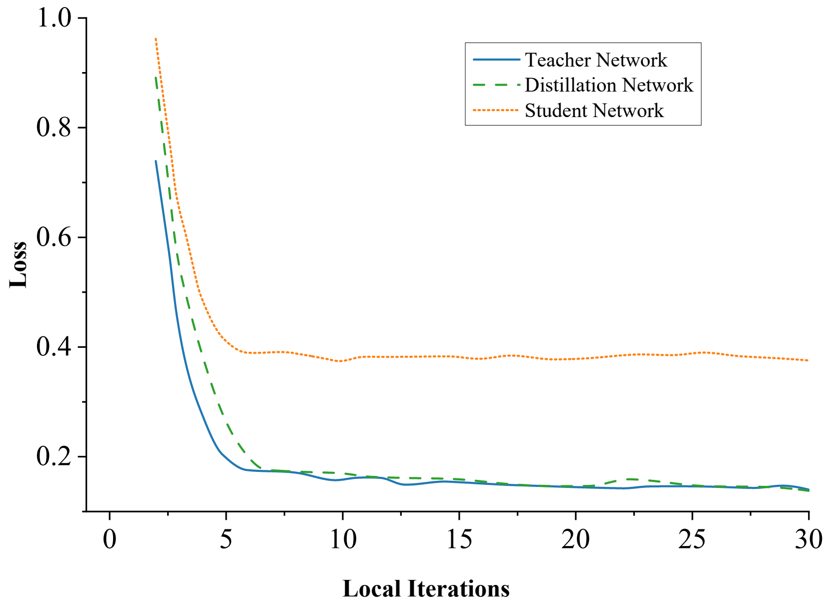
(a) Loss on User Dataset



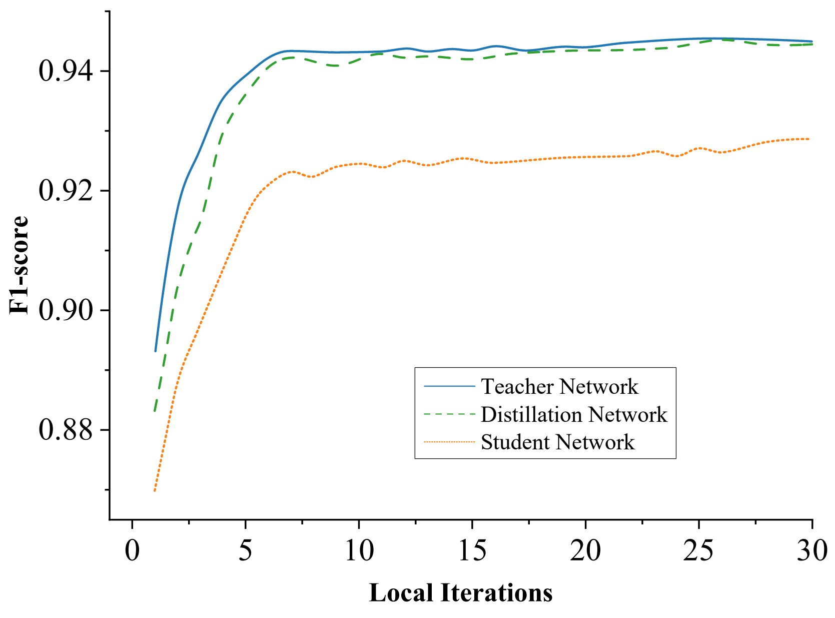
(b) F1-score on User Dataset

Figure 4-6 Effectiveness of Knowledge Distillation Algorithm based on MLP on User Dataset

As shown in Figure 4-6, experiments were conducted using the user dataset. Similar to the credit card dataset, in the initial stage of training, the training effect of the distillation network was between that of the teacher network and the student network. The loss and F1-score curves of the teacher network and the distillation network stabilized faster than those of the student network, with local iteration times of 5, 5, and 7, respectively. After 5 local iterations, the distillation network reached the training effect of the teacher network and was better than that of the student network. Ultimately, the loss of the teacher network, distillation network, and student network were 0.18, 0.20, and 0.30, respectively, and the F1-score were 0.951, 0.950, and 0.938, respectively.



(a) Loss on Bank Dataset



(b) F1-score on Bank Dataset

Figure 4-7 Effectiveness of Knowledge Distillation Algorithm based on MLP on Bank Dataset

As shown in Figure 4-7, experiments were conducted using the bank dataset. The trend of change was consistent with that of the two public datasets, and in the initial stage of training, the training effect of the distillation network was between that of the teacher network and the student network. After about 5 local iterations, the loss and F1-score curves of the distillation network gradually stabilized and approached those of the teacher network, while the student network required about 6 local iterations to stabilize, and its training effect was weaker than that of the teacher network and the distillation network. Ultimately, the loss of the teacher network, distillation network, and student network were 0.15, 0.16, and 0.37, respectively, and the F1-score were 0.946, 0.944, and 0.928, respectively.

The above three experiments verified the feasibility of the knowledge distillation algorithm based on MLP. Using this algorithm, a structurally simple student network can achieve the training effect of a structurally complex teacher network, thus achieving the goal of compressing the model and reducing communication costs.

4.5.2.2 Effectiveness of Knowledge Distillation Algorithm based on MLP

The chapter then compared the performance of three model compression methods: knowledge distillation based on MLP, model quantization, and model pruning. These methods aim to reduce the size of the model and improve its efficiency.

The knowledge distillation algorithm based on MLP transfers the knowledge of the teacher network to the student network to improve its performance and reduce the model size. However, this algorithm depends on the quality of the teacher model, and if the teacher model quality is low, knowledge distillation may not be able to improve the performance of the student network.

Model quantization involves converting model parameters from floating-point numbers to lower-precision representations to reduce the model size. However, model quantization may result in loss of model precision and increased training complexity. Additionally, model quantization typically requires good hardware support.

Model pruning reduces the size of the model by removing redundant parameters, thereby reducing the storage and computational requirements of the model. However, model pruning can also result in some loss of precision, and choosing an appropriate pruning algorithm is a significant challenge.

As shown in Table 4-1, the experiment compared the compression effects of the three algorithms on the credit card dataset, user dataset, and bank dataset. The evaluation metrics included the loss function and F1-score.

Table 4-1 Comparison of Model Compression Methods

Methods	Metrics	Credit Card	User	Bank
Uncompressed	Loss	0.22	0.18	0.15
	F1-score	0.916	0.951	0.946
Knowledge Distillation Based on MLP	Loss	0.24	0.20	0.16
	F1-score	0.915	0.950	0.944
Model Quantization	Loss	0.30	0.22	0.16
	F1-score	0.913	0.945	0.940
Model Pruning	Loss	0.33	0.27	0.25
	F1-score	0.883	0.907	0.898

The comparison results on the credit card dataset show that the loss of the uncompressed model was 0.22 and the F1-score was 0.916. After model compression using the MLP-based knowledge distillation algorithm, the loss slightly increased to 0.24, while the F1-score remained at 0.915. The model quantization compression method further increased the loss to 0.30, while the F1-score was 0.913. Finally, the model pruning compression method had the highest loss of 0.33, while the F1-score decreased to 0.883.

The comparison results on the user rating dataset show that the loss of the uncompressed model was 0.18 and the F1-score was 0.951. After compression using the MLP-based knowledge distillation algorithm, the loss slightly increased to 0.20, while the F1-score was 0.950. The model quantization compression method further increased the loss to 0.22, while the F1-score was 0.945. Finally, the model pruning compression method resulted in a loss of 0.27, which was the highest loss among the three compression methods, while the F1-score decreased to 0.907.

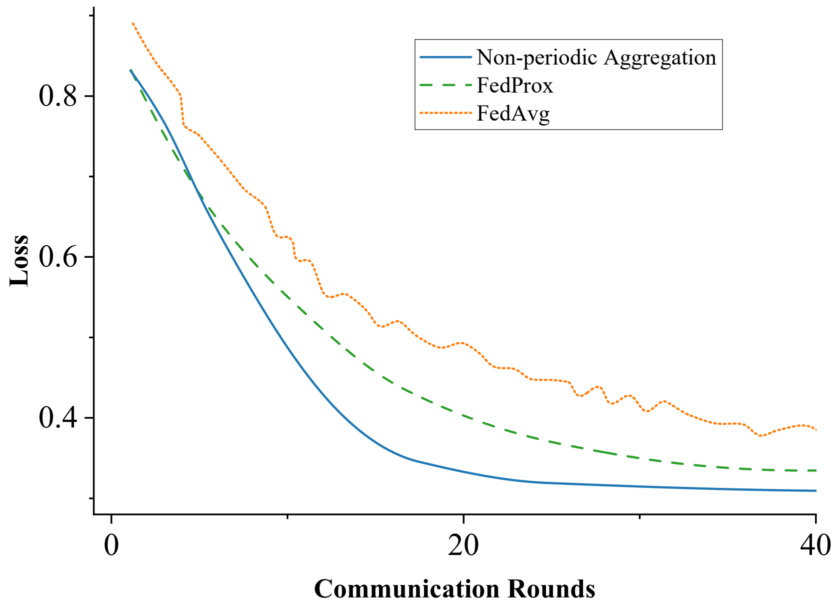
The comparison results on the bank dataset show that the loss of the uncompressed model was 0.15 and the F1-score was 0.946. After compression using the MLP-based knowledge distillation algorithm, the loss slightly increased to 0.16, while the F1-score was 0.944. The model quantization compression method further increased the loss to 0.16, while the F1-score was 0.940. Finally, the model pruning compression method resulted in a loss of 0.25, while the F1-score decreased to 0.898.

According to the results in the table, different compression methods have different effects on model performance. The MLP-based knowledge distillation algorithm and the model quantization method relatively well preserved the model's performance, with small changes in loss and F1-score. When using the knowledge distillation algorithm, although the model loss slightly increased, the F1-score remained at a high level, indicating that the student model successfully obtained effective knowledge from the teacher model. The model quantization method reduced the model size while having a relatively small impact on model performance, with small changes in loss and F1-score. However, the model quantization method has high hardware requirements and requires certain hardware conditions to achieve optimal performance and compression results. In contrast, the model pruning method significantly reduced the model size, but correspondingly weakened the model's performance, with significant decreases in loss and F1-score compared to other methods.

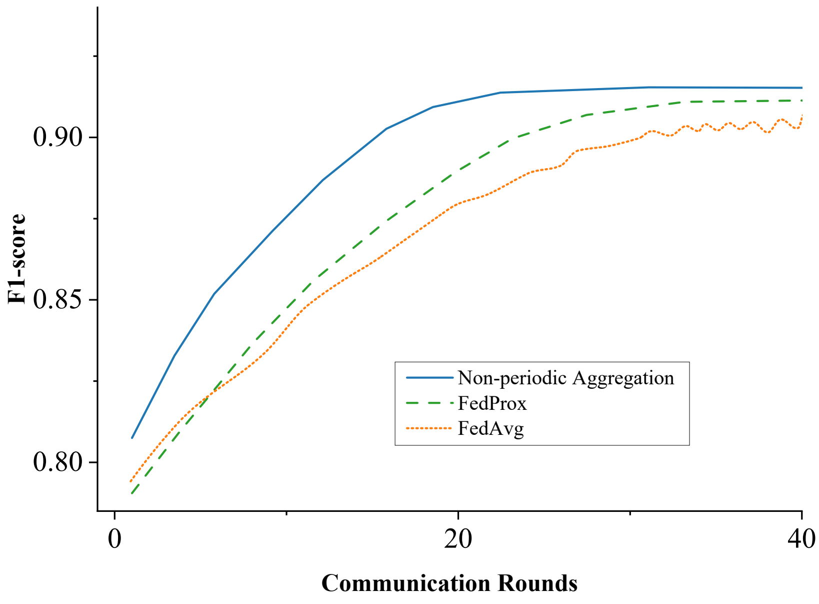
4.5.2.3 Effectiveness of Non-periodic Aggregation Algorithm

Finally, this chapter verified the effectiveness of the non-periodic aggregation algorithm. In the federated learning environment, experiments were conducted using the credit card dataset, user dataset, and bank dataset, respectively. By comparing the aggregation effects of the non-periodic aggregation algorithm, FedProx algorithm, and FedAvg algorithm, the feasibility of the non-periodic aggregation algorithm was evaluated.

The comparison results of the three aggregation algorithms on the credit card dataset are shown in Figure 4-8. The non-periodic aggregation algorithm achieved the minimum loss and maximum F1-score after approximately 20 rounds of communication, followed by the FedProx algorithm, while the FedAvg algorithm had the slowest convergence speed. The three algorithms reached a stable state after approximately 20, 30, and 40 rounds of communication, respectively. Ultimately, the loss of the three aggregation algorithms on the credit card dataset was 0.32, 0.35, and 0.40, respectively, and the F1-score was 0.912, 0.908, and 0.905, respectively. It should be noted that due to the heterogeneity of the credit card dataset, the loss curve and F1-score curve of the FedAvg algorithm had certain volatility. These results indicate that when processing the credit card dataset, the non-periodic aggregation algorithm performed best in achieving lower loss and higher F1-score, followed by the FedProx algorithm, while the FedAvg algorithm performed relatively poorly. Therefore, the non-periodic aggregation algorithm showed better aggregation effects on the credit card dataset.



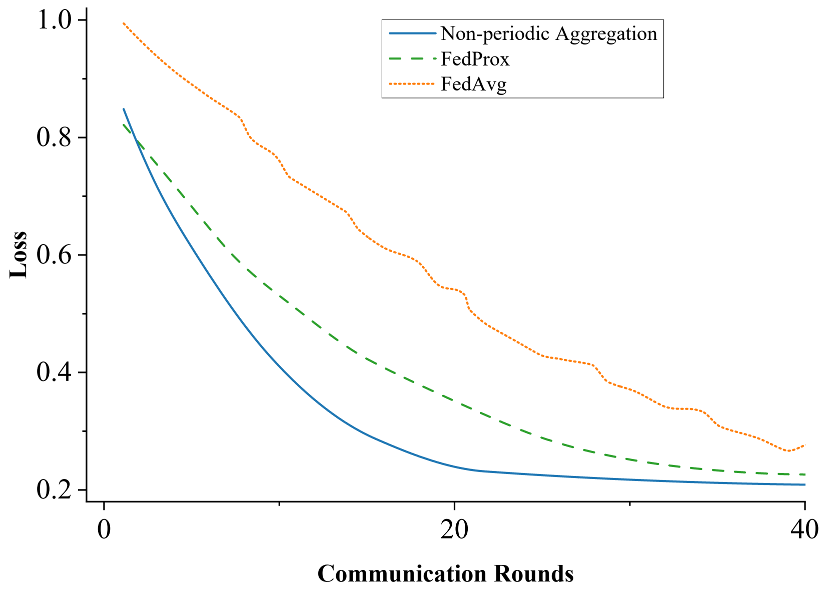
(a) Loss on Credit Card Dataset



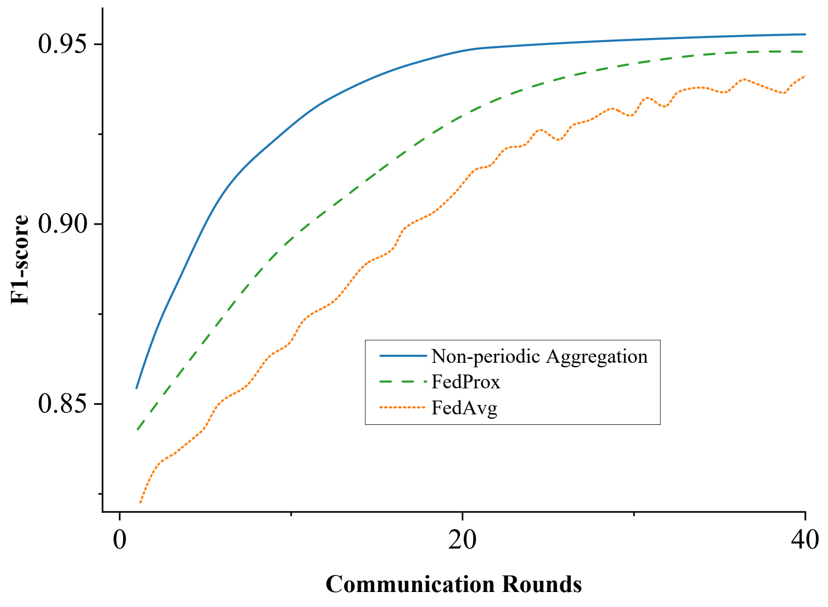
(b) F1-score on Credit Card Dataset

Figure 4-8　　Loss and F1-score of Different Aggregation Algorithms on Credit Card Dataset

The comparison results of the three aggregation algorithms on the user dataset are shown in Figure 4-9. The non-periodic aggregation algorithm showed the fastest convergence speed, followed by the FedProx algorithm, while the FedAvg algorithm had the slowest convergence speed and larger result fluctuations. The number of communication rounds required for the non-periodic aggregation algorithm, FedProx algorithm, and FedAvg algorithm to reach a stable state were approximately 20, 30, and 40 rounds, respectively. Ultimately, the loss of the three aggregation algorithms on the user dataset was 0.22, 0.26, and 0.30, respectively, and the F1-score was 0.950, 0.948, and 0.943, respectively. These results indicate that when processing the user dataset, the non-periodic aggregation algorithm had the fastest convergence speed and the lowest loss, followed by the FedProx algorithm, while the FedAvg algorithm had a relatively slower convergence speed and larger result fluctuations. Therefore, the non-periodic aggregation algorithm showed better aggregation effects on the user dataset.

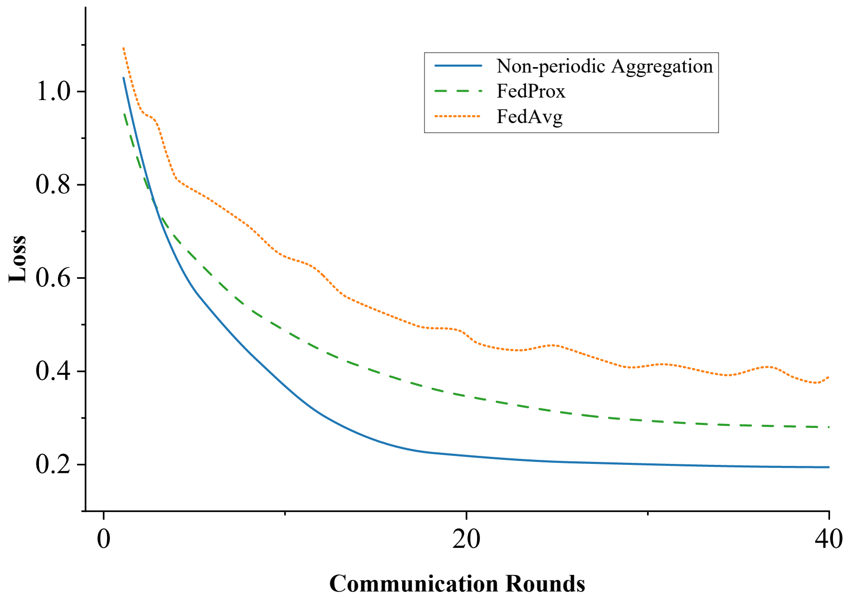


(a) Loss on User Dataset

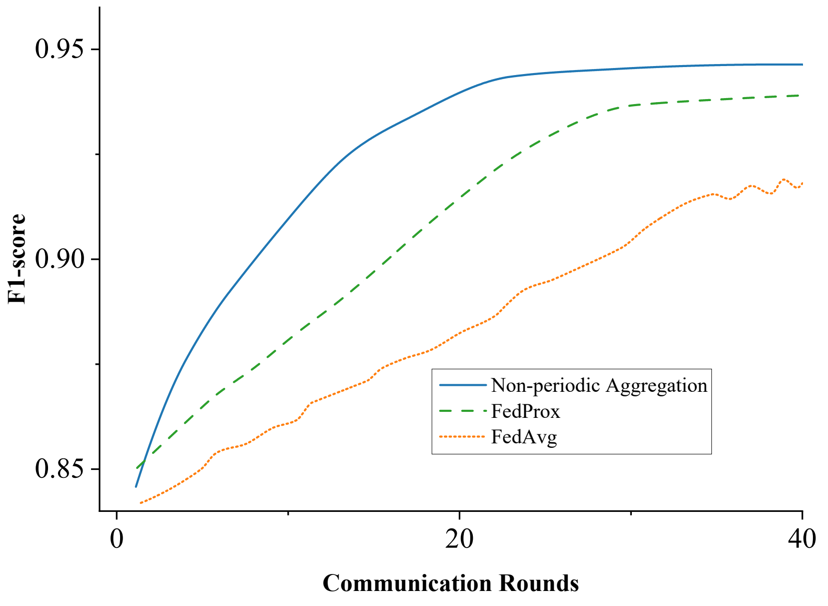


(b) F1-score on User Dataset

Figure 4-9 Loss and F1-score of Different Aggregation Algorithms on User Dataset



(a) Loss on Bank Dataset



(b) F1-score on Bank Dataset

Figure 4-10 Loss and F1-score of Different Aggregation Algorithms on Bank Dataset

The comparison results of the three aggregation algorithms on the bank dataset are shown in Figure 4-10. The non-periodic aggregation algorithm achieved the best results fastest, followed by the FedProx algorithm, while the FedAvg algorithm was the slowest. The non-periodic aggregation algorithm, FedProx algorithm, and FedAvg algorithm reached stability after approximately 20, 30, and 40 rounds of communication, respectively. Ultimately, the loss of the three aggregation algorithms on the bank dataset was

0.20, 0.30, and 0.41, respectively, and the F1-score was 0.942, 0.935, and 0.920, respectively. Similar to the results of the two public datasets, the result curve of the FedAvg algorithm on the bank dataset was also fluctuating. These results indicate that when processing the bank dataset, the non-periodic aggregation algorithm demonstrated the fastest convergence speed and lower loss, followed by the FedProx algorithm, while the FedAvg algorithm had a relatively slower convergence speed and larger result fluctuations. Therefore, the non-periodic aggregation algorithm showed superior aggregation effects on the bank dataset.

The above three experiments have verified the feasibility of the non-periodic aggregation algorithm. In all three datasets, the non-periodic aggregation algorithm demonstrated the fastest convergence speed. In other words, to achieve the same training effect (same loss and same F1-score), the non-periodic aggregation algorithm requires fewer communication rounds, and therefore, the communication cost of the algorithm is lower. Moreover, since FedAvg algorithm does not have an advantage in handling data heterogeneity problems, the curve of FedAvg algorithm inevitably shows fluctuation.

4.6 Summary

This chapter has presented an optimization approach for communication costs based on MLP knowledge distillation algorithm and non-periodic aggregation algorithm. The MLP knowledge distillation algorithm distills knowledge from a complex network into a simpler one, allowing the simpler network to achieve the same training effect as the complex one, thereby compressing the model size. The non-periodic aggregation algorithm optimizes the existing periodic aggregation algorithm to improve the convergence speed of aggregation, using the minimum number of communication rounds to aggregate the optimal global model, thereby reducing communication costs. To verify the effectiveness of the two algorithms, this chapter first compared the training effects of the teacher network, distilled network, and student network on the credit card dataset, user dataset, and bank dataset. The results showed that the MLP knowledge distillation algorithm could compress the teacher network into the student network. Secondly, the chapter evaluated the compression effects of MLP knowledge distillation, model quantization, and model pruning. The results showed that MLP knowledge distillation and model quantization both performed well, but model quantization had higher hardware requirements for the machine. Finally, the aggregation effects of the non-periodic aggregation algorithm, FedProx algorithm, and FedAvg algorithm were compared on the three datasets. The results showed that the non-periodic aggregation algorithm could achieve the same training effect as the other two aggregation algorithms with fewer communication rounds, reducing communication costs, and could also effectively address data heterogeneity issues.

5 Credit Card Fraud Detection Case Analysis

In practical applications, many machine learning methods face challenges in credit card fraud detection, including data distribution discrepancies and model generalization ability. Data distribution discrepancies refer to the differences between the training set and actual data, which may lead to a decrease in model performance. Model generalization ability involves the model's adaptability to unseen data, especially for minority classes and new fraud behaviors. These issues limit the accuracy and efficiency of traditional methods. This chapter deploys the proposed model and algorithm in a real-world credit card fraud detection scenario, compares it with actual detection methods, and analyzes their detection results.

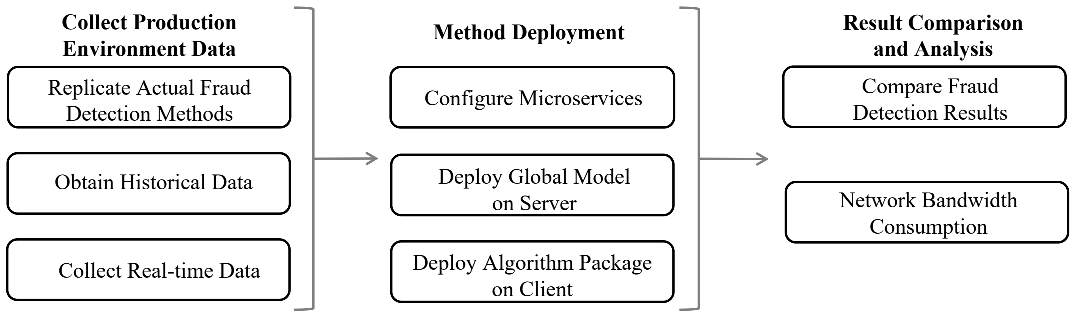

Collect Production Environment Data
Replicate Actual Fraud Detection Methods
Obtain Historical Data
Collect Real-time Data
Method Deployment
Configure Microservices
Deploy Global Model on Server
Deploy Algorithm Package on Client
Result Comparison and Analysis
Compare Fraud Detection Results
Network Bandwidth Consumption

Figure 5-1 Credit Card Fraud Detection Case Analysis Process

As shown in Figure 5-1, this chapter mainly consists of three stages: production environment data collection, method deployment, and result comparison and analysis. In the production environment data collection phase, it is necessary to reproduce the actual fraud detection method, obtain historical data, and collect real-time data. In the method deployment phase, the microservice environment needs to be configured, followed by deploying the global model on the server and testing and validation on the client-side. In the result comparison and analysis phase, the fraud detection results need to be compared and analyzed, and the network bandwidth consumption needs to be evaluated.

5.1 Data Collection and Model Deployment

Production Environment Data Collection

In the deployment process of fraud detection methods in financial institutions, the importance of enterprise security and procedures cannot be ignored. To ensure compliance with the company's privacy regulations and protect the security of customer sensitive information, it is necessary to work closely with the testing department and operations department to collect data in the production environment. The following steps are taken in this phase:

First, to reproduce the actual fraud detection method, the relevant testing environment needs to be configured, and the server and client participating in federated learning need to be deployed. The purpose of this process is to provide a baseline method to compare with the fraud detection method proposed in this thesis and evaluate their advantages and disadvantages in fraud detection. Due to the privacy regulations of the company, the fraud detection method cannot be directly deployed in the production environment. Therefore, it is necessary to work closely with the testing department and operations department to build a testing environment that simulates the production environment. The configuration parameters of this environment include data flow, network configuration, and security measures, among others, to accurately evaluate the performance and reliability of the model while ensuring the security and privacy requirements of the enterprise. Next, the server and client participating in federated learning need to be selected, and the client needs to be registered to the server using the platform's built-in registration function. This registration process includes identity verification and access authorization steps to ensure the credibility of the deployed client and protect the security of client data. The testing department provides a server that registers three clients, each of which simulates different user behavior and transaction scenarios. Finally, the actual fraud detection method in the production environment can be deployed and reproduced for comparative evaluation.

In the second step, to achieve model training and application, data needs to be collected for both training and testing purposes. For the training set, historical data related to credit cards needs to be collected and used for model training. This process requires collaboration with the operations department to obtain a portion of actual historical data that has been desensitized as the training set to complete the training of the federated learning model.

Finally, for the test set, real-time data needs to be collected and used to validate the fraud detection model. First, data collectors provided by the operations department need to be deployed on the client-side to collect test data. In practice, a client may correspond to a bank branch, and the client's data sources include multiple channels, such as data generated by credit cards handled by the branch or related users' online transactions. This data usually includes transaction account, transaction amount, transaction time, etc. Then, by monitoring the client's activities, information related to credit cards can be obtained. For each client, a period of data monitoring needs to be conducted to obtain test data for method validation. Test personnel choose one day, monitoring from 9:00 am to 6:00 pm. This time period is considered a relatively busy and representative time period for transactions, providing sufficient data samples.

During the data collection process, each client generates a different amount of data. Client A generates approximately 70,000 pieces of data, client B generates approximately 80,000 pieces of data, and client C generates approximately 70,000 pieces of data. These data need to be screened and cleaned to remove some invalid or non-compliant data. Finally, the available data for client A is 50,324 pieces, for client B is 59,845 pieces, and for client C is 55,032 pieces. Each client will use its generated valid data as the original data to compare and verify the effectiveness of the fraud detection model.

Method Deployment

To apply the trained model to the simulated production environment built by the testing department, the following method is adopted to deploy the fraud detection method:

First, to integrate the proposed method into the production environment while minimizing the impact on other functions, a microservice environment should be configured. This microservice architecture allows the proposed method to run relatively independently and does not negatively affect the normal operation of other functions, ensuring the stability and reliability of the server and client. Through this architecture, the overall operating efficiency can be effectively guaranteed while meeting the enterprise's requirements for function isolation and maintainability.

Second, execute the Python script on the server to deploy the trained model to the server. The purpose of this script is to load the pre-trained model as a global model onto the server so that the deployed model can be used for federated learning testing in the production environment.

Finally, comprehensive testing and validation should be performed on the client-side to analyze the effect of the proposed method.

5.2 Result Comparison and Experimental Analysis

In this section, the actual fraud detection method and the fraud detection method proposed in this thesis are run on the same dataset and production environment, and their detection results are compared. The focus is on the following two aspects: first, comparing the fraud data detected by the two methods and analyzing them in combination with the characteristics of the dataset to understand their detection preferences and effects. Second, comparing the bandwidth consumption of the two methods during model training to evaluate their communication costs. Through these comparative experiments, it is helpful to deeply understand the performance differences between the method proposed in this thesis and the actual method and provide reference value for subsequent optimization.

Fraud Detection Result Comparison

The actual fraud detection process of a bank is as follows: first, screen out potential fraudulent transaction data using a fraud detection model and threshold segmentation method. This data includes but is not limited to transactions with too high transfer amounts or too frequent transactions between the two parties. Then, the bank will assign specialized personnel to verify and judge this data. If it is determined to be true fraud data, a warning will be issued to the relevant users. The comparative experiment is based on this process. First, the fraud detection method proposed in this thesis (referred to as ”proposed method”) and the actual fraud detection method (referred to as ”actual method”) are used to screen fraud data, and then experts will analyze and verify it.

As shown in Figure 5-2, for the fraud data detected by the model, it is first observed that in client B and client C, the actual method detects more fraud data than proposed method. In client B, proposed method detects 102 fraud data and the actual method detects 150 fraud data, while in client C, proposed method detects 98 fraud data and the actual method detects 117 fraud data. This is because the actual method has a threshold segmentation screening, i.e., the actual method will mark large transfer amounts that conform to user behavior (such as monthly loan repayment transactions) as fraud, while proposed method only screens data that does not conform to user behavior characteristics based on historical data. Therefore, the actual method detects more fraud data. Second, in client A, proposed method detects more fraud data than the actual method, with 120 and 103 fraud data, respectively. After verification, it was found that the test dataset of client A contains many data generated by new users. Due to insufficient training on such data during model generation, proposed method is more likely to judge these data as fraud data. Based on these results, it is found that the actual method tends to be more conservative in detection, marking any large transfer as potential fraud data, while proposed method focuses more on screening fraud data that does not conform to user data characteristics. However, when dealing with data from new users, proposed method has a higher misjudgment rate.

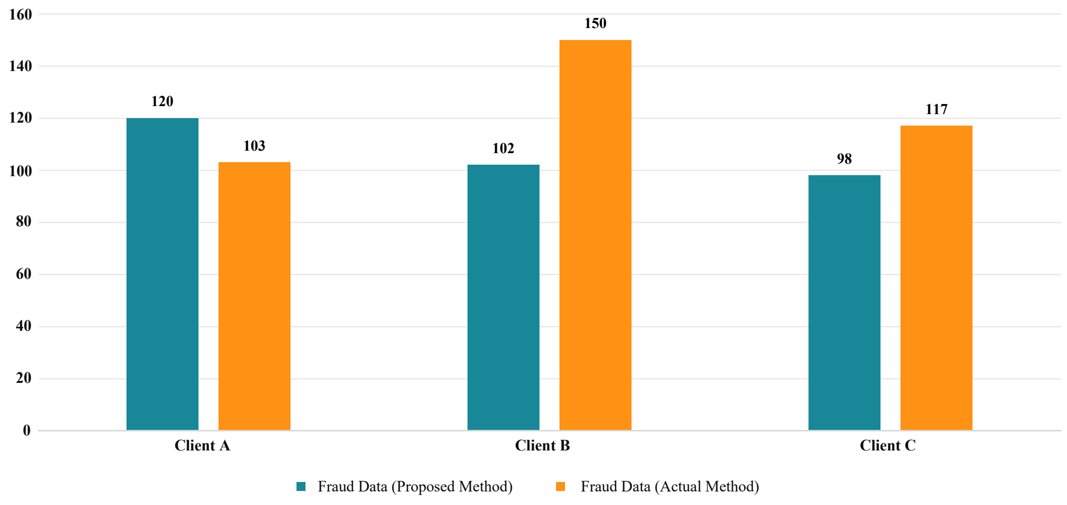


Figure 5-2 Comparison of Fraud Detection Results: Proposed Method vs. Actual Method

Next, business experts manually verify and evaluate the fraud data screened by the methods. During this process, experts found some differences between the proposed method and the actual method. Specifically, except for the scenarios involving new user data, the fraud data detected by the proposed method is completely included in the fraud data detected by the actual method, and the detection scope of the actual method is broader. This difference is mainly because the actual method uses threshold screening, which can better identify and warn of large-scale or large-amount transfer impersonation behaviors. In comparison, the proposed method finds it difficult to effectively detect such behaviors. Therefore, experts recommend that when facing impersonation behaviors, it is still necessary to rely on the capabilities of the actual method to provide early warning and take appropriate measures.

After verifying the proposed method, it is confirmed that the actual method is more comprehensive in fraud detection. The threshold mechanism used by the actual method provides additional assurance for screening fraud data. This provides a clear direction for optimizing the proposed method in the future.

Network Bandwidth Consumption Comparison

To evaluate the communication cost of the proposed method and the actual method, the network bandwidth consumption of the three clients (client A, client B, and client C) during the federated learning training process should be monitored at communication rounds of 10, 20, 30, and 40. By comparing these data, the communication costs of the two methods can be understood.

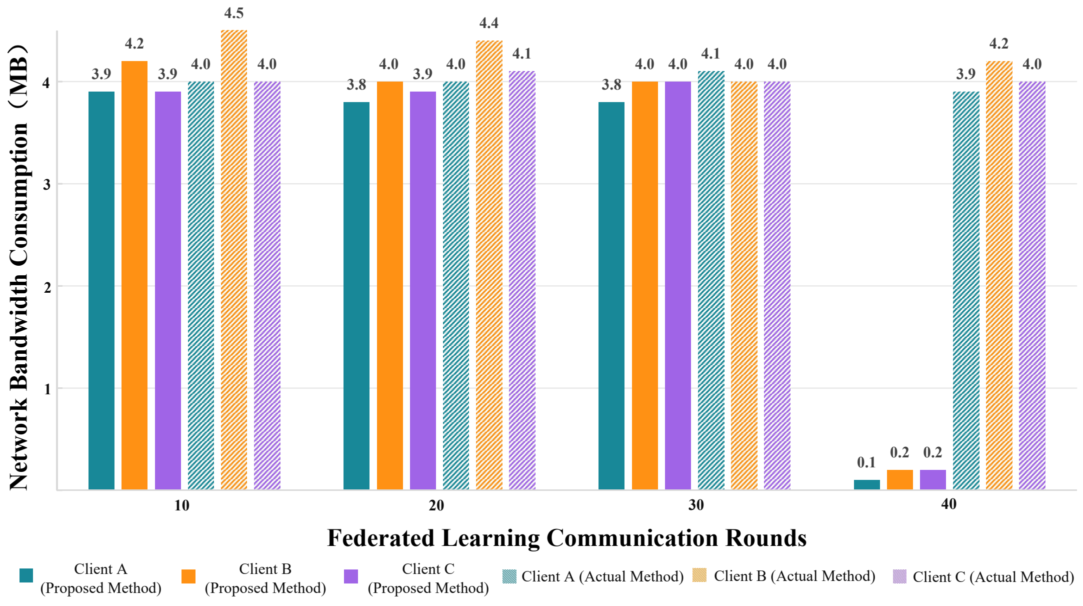


Figure 5-3 Bandwidth Consumption Comparison: Proposed Method vs. Actual Method

As shown in Figure 5-3, by comparing the network bandwidth consumption of the proposed method and the actual method on the client-side, it is observed that the proposed method has lower communication overhead. When the communication round is 10, the network bandwidth consumption of the proposed method and the actual method on client A is 3.9MB and 4.0MB, respectively, on client B is 4.2MB and 4.5MB, respectively, and on client C is 3.9MB and 4.0MB, respectively. When the communication round is 20, the network bandwidth consumption of the proposed method and the actual method on client A is 3.8MB and 4.0MB, respectively, on client B is 4.0MB and 4.4MB, respectively, and on client C is 3.9MB and 4.1MB, respectively. When the communication round is 30, the network bandwidth consumption of the proposed method and the actual method on client A is 3.8MB and 4.1MB, respectively, on client B is 4.0MB and 4.1MB, respectively, and on client C is 4.0MB and 4.0MB, respectively. This result is due to the application of the knowledge distillation algorithm, which reduces the size of the transmitted model.

Particularly, when the communication round is 40, the proposed method and the actual method show significant differences in network bandwidth consumption on different clients. On client A, the network bandwidth consumption of the proposed method is 0.1MB, while that of the actual method is 3.9MB. On client B, the proposed method consumes 0.2MB, while the actual method consumes 4.2MB. On client C, the proposed method consumes 0.2MB, while the actual method consumes 4.0MB. The communication overhead of the proposed method is much lower than that of the actual method, which is due to the application of the non-periodic aggregation algorithm. By the 40th communication round, the proposed method has completed the federated learning training process. Therefore, there is no further model exchange after this round, and only a small amount of network bandwidth consumption exists. This means that the proposed method has higher communication efficiency and can better utilize network resources.

The above experimental results show that the proposed method has smaller communication overhead in actual application. This result further verifies the feasibility and effectiveness of the knowledge distillation algorithm and the non-periodic aggregation algorithm in federated learning.

5.3 Summary

This chapter applies the federated learning fraud detection method with differential privacy protection to actual credit card fraud detection. First, data collection is completed in the production environment, followed by the deployment of the proposed method. Finally, the results of the proposed fraud detection method and the actual fraud detection method are compared. By comparing the detected fraud data and the network bandwidth consumption during the training process, the differences between the results of the two methods are analyzed, and the reasons for these differences are discussed.

6 Conclusion and Outlook

6.1 Work Summary

In the big data era, multiple financial institutions are collaborating to build fraud detection models, and federated learning (FL) has emerged to ensure data security and privacy during collaboration. However, FL faces challenges in security and communication overhead. This thesis proposes an FL fraud detection method with differential privacy protection to improve detection accuracy while ensuring data security and privacy. The main contributions are:

(1) A federated learning fraud detection method with differential privacy protection is proposed. The IDW-SMOTE algorithm is designed to address imbalanced data distribution, and a stacked autoencoder's feature extraction is integrated with a multi-layer perceptron's classification model to build an effective fraud detection model. A differential privacy algorithm is combined with blockchain features to encrypt and trace local models, ensuring data security and privacy during FL training and transmission. The thesis conducts validation experiments on multiple datasets, comparing the detection results of IDW-SMOTE, SMOTE, and original data, as well as the detection accuracy of SAE-MLP, MLP, CNN, KNN, SVM, decision tree, random forest, and logistic regression. The results demonstrate the superior performance of the proposed method on three datasets, and the encryption algorithm effectively ensures the security and privacy of model data.

(2) A federated learning strategy is proposed to reduce communication overhead through model compression and optimizing the aggregation time series. The knowledge distillation algorithm is integrated with the multi-layer perceptron model to compress local models and reduce the data size of transmitted models. The relationship between the loss function and the aggregation time series is revealed, and communication rounds between the client and server are reduced to further lower the communication overhead of federated learning. The thesis conducts validation experiments on multiple datasets, validating the feasibility of the MLP-based knowledge distillation algorithm and comparing its compression ef-

fect with model quantization and model pruning. The non-periodic aggregation algorithm, FedAvg algorithm, and FedProx algorithm are compared. The results demonstrate that the MLP-based knowledge distillation algorithm effectively compresses complex fraud detection models. Additionally, the non-periodic aggregation algorithm reduces communication rounds by 10 rounds compared to FedProx and 20 rounds compared to FedAvg, achieving further efficiency improvements.

Additionally, the proposed fraud detection method was successfully applied in the production environment, achieving bank credit card fraud detection. Through expert analysis and verification, the method demonstrated high accuracy and reliability in identifying fraud transactions. Moreover, it underwent testing and evaluation for network bandwidth consumption. The results indicated its strong performance in real-world scenarios and its capacity to handle large-scale data.

6.2 Problems and Prospects

Although this study has made progress in federated learning-based financial data fraud detection, there are still limitations. Future research should focus on the following areas for further investigation:

(1) Heterogeneity poses a significant challenge in federated learning. While the aggregation algorithm in this study considered data heterogeneity, real-world federated learning training environments still face device heterogeneity and network condition heterogeneity. Therefore, future research will focus on addressing other forms of heterogeneity and designing reliable communication protocols or mechanisms to ensure the security of the federated learning environment and improve the accuracy of trained models.

(2) This study primarily utilizes structured datasets stored in tabular form. However, in the practical application process of financial institutions, users may generate image datasets (e.g., signatures, credit card photos, user face photos), video datasets (e.g., user authentication), and audio datasets (e.g., user recordings). Hence, future research will focus on extending the federated learning training methods from structured datasets to other dataset types, aiming to explore federated learning fraud detection frameworks suitable for a wider range of datasets.

REFERENCES

[1] Kurshan E, Shen H. Graph computing for financial crime and fraud detection: Trends, challenges and outlook [J]. International Journal of Semantic Computing, 2020, 14 (04): 565–589.

[2] Kurshan E, Shen H, Yu H. Financial crime & fraud detection using graph computing: Application considerations & outlook [C]. In 2020 Second International Conference on Transdisciplinary AI (TransAI), 2020: 125–130.

[3] Harris H. Artificial intelligence and policing of financial crime: a legal analysis of the state of the field [J]. Financial Technology and the Law: Combating Financial Crime, 2022, 1: 281–299.

[4] Bhattacharyya S, Jha S, Tharakunnel K, et al. Data mining for credit card fraud: A comparative study [J]. Decision support systems, 2011, 50 (3): 602–613.

[5] Kirkos E, Spathis C, Manolopoulos Y. Data mining techniques for the detection of fraudulent financial statements [J]. Expert systems with applications, 2007, 32 (4): 995–1003.

[6] Webster K, Miranda P A, Murray S, et al. The State of Fraud and Financial Crime in the U.S. [R]. Feature Space, 2023.

[7] West J, Bhattacharya M. Intelligent financial fraud detection: a comprehensive review [J]. Computers & security, 2016, 57: 47–66.

[8] Zhang H, Hong J, Dong F, et al. A privacy-preserving hybrid federated learning framework for financial crime detection [J]. arXiv preprint arXiv:2302.03654, 2023.

[9] Maes S, Tuyls K, Vanschoenwinkel B, et al. Credit card fraud detection using Bayesian and neural networks [C]. In Proceedings of the 1st international naiso congress on neuro fuzzy technologies, 2002: 270–277.

[10] Sundarkumar G G, Ravi V. A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance [J]. Engineering Applications of Artificial Intelligence, 2015, 37: 368–377.

[11] Wang Y, Xu W. Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud [J]. Decision Support Systems, 2018, 105 (1): 87–95.

[12] Fu K, Cheng D, Tu Y, et al. Credit card fraud detection using convolutional neural networks [C]. In International Conference on Neural Information Processing, 2016: 483–490.

[13] Chen J, Shen Y, Ali R. Credit card fraud detection using sparse autoencoder and generative adversarial network [C]. In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2018: 1054–1059.

[14] Charitou C, Garcez A d, Dragicevic S. Semi-supervised GANs for fraud detection [C]. In 2020 International Joint Conference on Neural Networks (IJCNN), 2020: 1–8.

[15] Li T, Sahu A K, Talwalkar A, et al. Federated learning: Challenges, methods, and future directions [J]. IEEE signal processing magazine, 2020, 37 (3): 50–60.

[16] Sorkin A R. Too big to fail: The inside story of how Wall Street and Washington fought to save the financial system–and themselves [M]. London:Penguin, 2010: 130-133.

[17] Sectors V. Critical Infrastructure Protection: Progress Coordinating Government and Private Sector Efforts Varies by Sectors' Characteristics [J]. Government Accountability Office Reports, 2006, 71 (1): 1–5.

[18] Alkhadra R, Abuzaid J, AlShammari M, et al. Solar winds hack: In-depth analysis and countermeasures [C]. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2021: 1–7.

[19] Chatterjee P, Das D, Rawat D B. Use of Federated Learning and Blockchain towards Securing Financial Services [J]. arXiv preprint arXiv:2303.12944, 2023.

[20] Nasereddin M, ALKhamaiseh A, Qasaimeh M, et al. A systematic review of detection and prevention techniques of SQL injection attacks [J]. Information Security Journal: A Global Perspective, 2023, 32 (4): 252–265.

[21] Al-Alawi A I, Al-Bassam M S A. The significance of cybersecurity system in helping managing risk in banking and financial sector [J]. Journal of Xidian University, 2020, 14 (7): 1523–1536.

[22] Lu Y, Huang X, Dai Y, et al. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT [J]. IEEE Transactions on Industrial Informatics, 2019, 16 (6): 4177–4186.

[23] Li Y, Chen C, Liu N, et al. A blockchain-based decentralized federated learning framework with committee consensus [J]. IEEE Network, 2020, 35 (1): 234–241.

[24] Bandara E, Shetty S, Rahman A, et al. Bassa-ml—a blockchain and model card integrated federated learning provenance platform [C]. In 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), 2022: 753–759.

[25] Shayan M, Fung C, Yoon C J M, et al. Biscotti: A Blockchain System for Private and Secure Federated Learning [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32 (7): 1513–1525.

[26] Guo Y, Liang C. Blockchain application and outlook in the banking industry [J]. Financial innovation, 2016, 2: 1–12.

[27] 刘悦. 安全与隐私保护研究综述 [J]. 电子技术与软件工程, 2021 (9): 2–3.

[28] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data [C]. In Artificial intelligence and statistics, 2017: 1273–1282.

[29] Zhang C, Xie Y, Bai H, et al. A survey on federated learning [J]. Knowledge-Based Systems, 2021, 216: 106775–106786.

[30] Bottou L, Curtis F E, Nocedal J. Optimization methods for large-scale machine learning [J]. SIAM review, 2018, 60 (2): 223–311.

[31] Woodworth B, Patel K K, Stich S, et al. Is local SGD better than minibatch SGD? [C]. In International Conference on Machine Learning, 2020: 10334–10343.

[32] Woodworth B E, Patel K K, Srebro N. Minibatch vs local sgd for heterogeneous distributed learning [J]. Advances in Neural Information Processing Systems, 2020, 33: 6281–6292.

[33] Li T, Sahu A K, Zaheer M, et al. Federated optimization in heterogeneous networks [J]. Proceedings of Machine learning and systems, 2020, 2: 429–450.

[34] 高莹, 陈晓峰, 张一余，等. 联邦学习系统攻击与防御技术研究综述 [J]. 计算机学报, 2023, 46 (9): 1781–1805.

[35] 刘艺璇, 陈红, 刘宇涵，等. 联邦学习中的隐私保护技术 [J]. 软件学报, 2022 (003): 33–58.

[36] 肖雄, 唐卓, 肖斌，等. 联邦学习的隐私保护与安全防御研究综述 [J]. 计算机学报, 2023, 46 (5): 1019–1044.

[37] Tolpegin V, Truex S, Gursoy M E, et al. Data poisoning attacks against federated learning systems [C]. In 25th European Symposium on Research in Computer Security(ESORICS), 2020: 480–501.

[38] Cao D, Chang S, Lin Z, et al. Understanding distributed poisoning attack in federated learning [C]. In 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), 2019: 233–239.

[39] Huang W R, Geiping J, Fowl L, et al. Metapoison: Practical general-purpose clean-label data poisoning [J]. Advances in Neural Information Processing Systems, 2020, 33: 12080–12091.

[40] Bhagoji A N, Chakraborty S, Mittal P, et al. Analyzing federated learning through an adversarial lens [C]. In International Conference on Machine Learning, 2019: 634–643.

[41] Cui L, Qu Y, Xie G, et al. Security and privacy-enhanced federated learning for anomaly detection in IoT infrastructures [J]. IEEE Transactions on Industrial Informatics, 2021, 18 (5): 3492–3500.

[42] Shejwalkar V, Houmansadr A. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning [C]. In Proceedings 2021 Network and Distributed System Security Symposium, 2021: 1–19.

[43] Xia Q, Tao Z, Hao Z, et al. FABA: An Algorithm for Fast Aggregation against Byzantine Attacks in Distributed Neural Networks [C]. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019: 4824–4830.

[44] Suresh A T, McMahan B, Kairouz P, et al. Can You Really Backdoor Federated Learning? [C]. In Neural Information Processing Systems(NeurIPS), 2019: 1–10.

[45] Kaviani S, Shamshiri S, Sohn I. A defense method against backdoor attacks on neural networks [J]. Expert Systems with Applications, 2023, 213: 118990–119004.

[46] Wan W, Lu J, Hu S, et al. Shielding federated learning: A new attack approach and its defense [C]. In 2021 IEEE Wireless Communications and Networking Conference (WCNC), 2021: 1–7.

[47] Weng J, Weng J, Zhang J, et al. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive [J]. IEEE Transactions on Dependable and Secure Computing, 2019, 18 (5): 2438–2455.

[48] Yao X, Huang C, Sun L. Two-stream federated learning: Reduce the communication costs [C]. In 2018 IEEE Visual Communications and Image Processing (VCIP), 2018: 1–4.

[49] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network [J]. Stat, 2015, 1050: 9–18.

[50] Momcheva P T G. Sentiment detection with fedmd: Federated learning via model distillation [J]. Information Systems and Grid Technologies, 2020, 1: 1–12.

[51] Lin T, Kong L, Stich S U, et al. Ensemble distillation for robust model fusion in federated learning [J]. Advances in Neural Information Processing Systems, 2020, 33: 2351–2363.

[52] Tramèr F, Zhang F, Juels A, et al. Stealing Machine Learning Models via Prediction APIs [C]. In 25th USENIX security symposium (USENIX Security 16), 2016: 601–618.

[53] Jagielski M, Carlini N, Berthelot D, et al. High accuracy and high fidelity extraction of neural networks [C]. In 29th USENIX security symposium (USENIX Security 20), 2020: 1345–1362.

[54] Li X, Song Z, Yang J. Federated adversarial learning: A framework with convergence analysis [C]. In International Conference on Machine Learning, 2023: 19932–19959.

[55] Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning [C]. In 2019 IEEE symposium on security and privacy (SP), 2019: 739–753.

[56] Acar A, Aksu H, Uluagac A S, et al. A survey on homomorphic encryption schemes: Theory and implementation [J]. ACM Computing Surveys (Csur), 2018, 51 (4): 1–35.

[57] Ma J, Naas S-A, Sigg S, et al. Privacy-preserving federated learning based on multi-key homomorphic encryption [J]. International Journal of Intelligent Systems, 2022, 37 (9): 5880–5901.

[58] Damgård I, Geisler M, Krøigaard M, et al. Asynchronous multiparty computation: Theory and implementation [C]. In International workshop on public key cryptography, 2009: 160–179.

[59] Kaissis G, Ziller A, Passerat-Palmbach J, et al. End-to-end privacy preserving deep learning on multi-institutional medical imaging [J]. Nature Machine Intelligence, 2021, 3 (6): 473–484.

[60] Zhang W, Tople S, Ohrimenko O. Leakage of Dataset Properties in Multi-Party Machine Learning [C]. In 30th USENIX Security Symposium (USENIX Security 21), 2021: 2687–2704.

[61] Dwork C. Differential privacy [C]. In International colloquium on automata, languages, and programming, 2006: 1–12.

[62] Yang G, Wang S, Wang H. Federated learning with personalized local differential privacy [C]. In 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), 2021: 484–489.

[63] 曹世翔, 陈超梦, 唐朋，等. 基于函数机制的差分隐私联邦学习算法 [J]. 计算机学报, 2023, 46 (10): 2178–2195.

[64] Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [J]. Decentralized business review, 2008, 1: 1–9.

[65] Nguyen D C, Ding M, Pham Q V, et al. Federated learning meets blockchain in edge computing: Opportunities and challenges [J]. IEEE Internet of Things Journal, 2021, 8 (16): 12806–12825.

[66] Majeed U, Hong C S. FLchain: Federated learning via MEC-enabled blockchain network [C]. In 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2019: 1–4.

[67] Qi Y, Hossain M S, Nie J, et al. Privacy-preserving blockchain-based federated learning for traffic flow prediction [J]. Future Generation Computer Systems, 2021, 117: 328–337.

[68] Połap D, Srivastava G, Yu K. Agent architecture of an intelligent medical system based on federated learning and blockchain technology [J]. Journal of Information Security and Applications, 2021, 58: 102748–102756.

[69] Chai H, Leng S, Chen Y, et al. A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles [J]. IEEE Transactions on Intelligent Transportation Systems, 2020, 22 (7): 3975–3986.

[70] Short A R, Leligou H C, Papoutsidakis M, et al. Using blockchain technologies to improve security in federated learning systems [C]. In 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), 2020: 1183–1188.

[71] Gao L, Li L, Chen Y, et al. FGFL: A blockchain-based fair incentive governor for Federated Learning [J]. Journal of Parallel and Distributed Computing, 2022, 163: 283–299.

[72] Lu Y, Huang X, Zhang K, et al. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks [J]. IEEE Internet of Things Journal, 2020, 8 (4): 2276–2288.

[73] Kang J, Xiong Z, Niyato D, et al. Reliable federated learning for mobile networks [J]. IEEE Wireless Communications, 2020, 27 (2): 72–80.

[74] Chen J, Xue J, Wang Y, et al. Privacy-Preserving and Traceable Federated Learning for data sharing in industrial IoT applications [J]. Expert Systems with Applications, 2023, 213: 119036–119047.

[75] Xu X, Pautasso C, Zhu L, et al. A pattern collection for blockchain-based applications [C]. In Proceedings of the 23rd European Conference on Pattern Languages of Programs, 2018: 1–20.

[76] Fernández A, Garcia S, Herrera F, et al. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary [J]. Journal of artificial intelligence research, 2018, 61: 863–905.

[77] Vincent P, Larochelle H, Lajoie I, et al. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion [J]. Journal of Machine Learning Research, 2010, 11 (12): 3371–3408.

[78] Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. [J]. Psychological review, 1958, 65 (6): 386–411.

[79] Hinton G E, Salakhutdinov R R. Reducing the Dimensionality of Data with Neural Networks [J]. Science, 2006, 313(5786): 504–507.

[80] Li X, Huang K, Yang W, et al. On the Convergence of FedAvg on Non-IID Data [C]. In International Conference on Learning Representations, 2019: 1–26.

[81] Zhang H, Wu T, Cheng S, et al. Aperiodic local SGD: Beyond local SGD [C]. In Proceedings of the 51st International Conference on Parallel Processing, 2022: 1–10.