Federated Learning with Privacy Preservation in Large-Scale Distributed Systems Using Differential Privacy and Homomorphic Encryption

Yue Chen[1,*], Yufei Yang[1], Yingwei Liang[1], Taipeng Zhu[1], Dehui Huang[2]
[1]Information Center, Guangdong Power Grid Co., Ltd., Guangzhou 510699, Guangdong, China
[2]Chaozhou Power Supply Bureau Information Center, Guangdong Power Grid Co., Ltd., Chaozhou 521011, Guangdong, China
E-mail: chenyue_vip@hotmail.com
[*]Corresponding author

Keywords: ederated learning, large-scale, distributed environment, privacy preserving

Received: October 17, 2024

This study proposes a large-scale distributed privacy-preserving machine learning algorithm based on federated learning. The algorithm allows participants to jointly train high-quality models without sharing original data to meet the challenges brought by increasingly stringent data privacy and security regulations. To verify the performance of the federated learning system in a real-world environment, we built a distributed experimental platform consisting of multiple physical servers and evaluated it using several publicly available datasets such as MNIST, Federated EMNIST, and Federated CIFAR10/100. The experimental results show that the accuracy of the federated learning system is 97.3%, which is slightly lower than the 98.2% of the centralized learning method, but this is an acceptable trade-off considering the advantages of the federated learning method in protecting data privacy. In addition, our system only slightly drops to about 96.8% after the introduction of malicious clients, which proves the robustness of the federated learning system. Specifically, we adopt differential privacy technology, set the privacy budget ε=1.0, and add Gaussian noise to the model update to ensure that even if a malicious user accesses the model update, no sensitive information of any individual user can be inferred from it. The experimental conditions include but are not limited to: the communication protocol uses homomorphic encryption, the average communication volume per iteration is 150 MB, and the total communication volume is 30 GB; the average CPU utilization of the client is about 70%, and the GPU utilization is about 80%. These settings ensure the efficiency of the system's computing resources, and also reflect the balance between privacy protection and model performance.

Povzetek: Predlagan je algoritem strojnega učenja v velikem, porazdeljenem okolju, ki temelji na federativnem učenju in zagotavlja ohranjanje zasebnosti. Algoritem omogoča skupno usposabljanje visokokakovostnih modelov brez izmenjave izvornih podatkov. Z uporabo diferencialne zasebnosti in homomorfnega šifriranja zagotavlja zaščito zasebnosti in dosega primerljivo točnost s centraliziranim učenjem, hkrati pa ohranja robustnost tudi ob prisotnosti zlonamernih strank.

1 Introduction

With the advent of the big data era, the importance of protecting the privacy of personal data has become increasingly important. As more and more sensitive information is collected and analyzed, the risk of data leakage and unauthorized access rises significantly. The traditional centralized machine learning approach, which aggregates data from various sources onto a single server, has become less feasible due to privacy concerns and regulatory restrictions [1, 2]. To address these issues, federated learning (FL) has emerged as a promising paradigm that supports collaborative machine learning without the need for centralized data. This approach not only reduces privacy risks, but also complies with strict data protection laws such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) [3].

Federated Learning Overview Federated learning is a distributed machine learning technique that allows multiple participants to collaboratively train a shared model while keeping the data decentralized [4]. The method utilizes the computational resources of client devices (e.g., smartphones or edge servers) to perform local training and then aggregates model updates at a central server. By doing so, FL minimizes the risk of data leakage and ensures the confidentiality of sensitive information. Moreover, Federated Learning is scalable, efficient and robust, making it a suitable solution for real applications involving large datasets [5].

The research objective is to propose a new federated learning framework capable of handling large-scale distributed datasets while guaranteeing the privacy of

participating entities [6]. Specifically, we focus on improving the efficiency of model training while maintaining strong privacy guarantees. Our goal is to design and implement new privacy-preserving techniques that can be seamlessly integrated into the joint learning process, thus ensuring that sensitive information remains confidential throughout the training phase. Furthermore, we aim to evaluate the performance and scalability of our proposed approach through a large number of simulations and real-world case studies [7].

Recent research in federated learning has explored various approaches to enhance privacy and security. Techniques such as differential privacy, secure aggregation, and homomorphic encryption have been successfully applied to protect the privacy of individual contributions during model training [8]. However, these methods typically tradeoff between privacy and computational efficiency. Our work builds on these foundations by proposing new strategies that effectively balance these factors. We review existing work in this area, highlight the strengths and limitations of current approaches, and identify areas for further research [9, 10].

This study aims to explore how to implement an efficient federated learning system in a large-scale distributed environment while ensuring that data privacy is fully protected. We assume that by introducing appropriate privacy protection mechanisms (such as differential privacy), the privacy protection level of the system can be significantly improved without affecting the performance of the model.

Our goal is to develop a federated learning framework that can handle large-scale distributed datasets while ensuring the privacy of participating entities. To this end, we focus on optimizing model training efficiency while maintaining strong privacy guarantees. In particular, we hope to design and integrate new privacy-preserving techniques into the federated learning process to ensure that sensitive information remains confidential throughout the training phase.

2 Related work

2.1 Foundations of federal learning

Federated Learning (FL), a distributed machine learning paradigm, allows multiple client devices (e.g., smartphones or edge servers) to collaboratively train a shared model without having to centralize the data in a single central server [10]. At its core, it allows each client to train a model on a local dataset and then send the updated model parameters to a central server, which is responsible for aggregating these updates to form a global model. Kilčiauskas et al. propose a method for confidential transaction balance verification using non-interactive zero-knowledge proofs, enhancing privacy and security in financial transactions [11]. In a similar vein, Larriba and López introduce the Secure Unencrypted Voting Scheme (SUVS), which ensures secure voting without encryption, maintaining both privacy and integrity during the voting process. These approaches contribute significantly to improving security and privacy in digital systems by leveraging advanced cryptographic techniques [12].

Federated learning offers significant advantages over traditional centralized machine learning approaches. Traditional distributed machine learning usually requires data to be centralized to a central node, which not only increases the risk of data leakage, but also may raise data protection regulation issues [13]. Federated learning, on the other hand, keeps data decentralized and each client only shares model updates rather than raw data, which protects data privacy and makes full use of the computational resources of client devices [14]. This property of federated learning makes it an effective solution to the data privacy problem. For example, in the healthcare domain, federated learning allows hospitals to jointly train more accurate disease diagnosis models without sharing sensitive patient data [15]. In the financial industry, different organizations can use federated learning to improve the performance of fraud detection systems while complying with strict privacy laws [16]. In addition, federated learning supports personalized model customization, i.e., it allows clients to train on local data and thus adapt to the needs of specific users [17]. Federated learning also has excellent scalability and robustness. It can accommodate a large number of client devices for massively distributed scenarios [18]. Even if some clients fail or go offline, the overall training process is not affected [19, 20]. As a result, federated learning is not only able to effectively utilize computational resources on a global scale, but also maintains efficient operation under various network conditions. In conclusion, federated learning brings new opportunities to a wide range of fields by facilitating advances in model training while protecting data privacy. With the continuous maturation and improvement of the technology, federated learning will play a greater role in the future to promote innovation and development in the field of artificial intelligence.

2.2 Privacy-preserving technologies

Differential Privacy (DP) is a statistical framework used to protect the privacy of individual data while publishing statistics [21]. By adding random noise to the data, Differential Privacy ensures that even changes in individual data records do not significantly affect the final published statistics. In federated learning, differential privacy can be used to protect model updates uploaded by clients, ensuring that private information about individual data is not disclosed even during model training.

The basic idea of differential privacy is to blur the impact of individual data by adding random noise to the model update, thus making it difficult for an attacker to infer the exact value of the original data [22, 23]. This approach ensures that the release of model updates does

not reveal too much information about individual data points. However, the amount of noise added to ensure a sufficient level of privacy protection may result in a degradation of model performance. Therefore, privacy protection needs to be carefully weighed against model accuracy when applying differential privacy in federated learning. Homomorphic Encryption (HE) is a cryptographic technique that allows computational operations to be performed directly on encrypted data without having to decrypt the data first. In the context of federated learning, homomorphic encryption can be used to encrypt model updates uploaded by a client, allowing a central server to aggregate these updates without decryption, thus completing model training while ensuring privacy. The main advantage of homomorphic encryption is its ability to perform computations without revealing the content of the data, which is useful for privacy protection. However, there are some challenges associated with homomorphic encryption, such as the high computational overhead of the encryption and decryption processes, which may affect the overall efficiency of federated learning. In addition, homomorphic encryption implementations are usually complex and require specialized design and optimization to ensure feasibility in real-world applications. Secure MultiParty Computation (MPC) is a cryptographic protocol that allows multiple participants to jointly compute the result of a function without revealing their respective input values. In federated learning, Secure MultiParty Computation can be used to ensure that the aggregation process of model updates is secure, i.e., the central server and other clients cannot learn the raw data or model updates of any party. Secure multiparty computation can provide strong privacy protection because it ensures that no party has access to other parties' sensitive information throughout the computation process. However, secure multiparty computation faces the same problems of computational complexity and communication overhead, especially when the number of participating parties is large. In addition, effective verification mechanisms need to be designed to ensure the correctness and security of the computation. In federated learning, privacy-preserving techniques such as differential privacy, homomorphic encryption, and secure multi-party computation have their own advantages and limitations. Differential privacy protects privacy by adding noise, but may reduce the accuracy of the model.

Table 1: Comparison of existing federated learning methods

Method Name	Dataset	Privacy protection technology	Accuracy (%)	Compute/communication tradeoff	Remark
Focused Learning	MNIST, CIFAR10	none	98.2 (MNIST), 84.0 (CIFAR10)	Efficient computing, high volume communication	No client-to-client communication involved; no privacy protection provided
Non-federated distributed learning	MNIST, CIFAR10	none	97.9 (MNIST), 83.5 (CIFAR10)	Moderate computing, moderate communications	Distributed but non-federated structure; lack of privacy protection mechanism
Standard Federated Learning (FL)	MNIST, EMNIST, CIFAR10	DP	96.8 (MNIST), 88.7 (EMNIST), 81.0 (CIFAR10)	Low communication, high privacy protection	Emphasis on privacy but at the expense of some performance
Homomorphic encryption FL	MNIST, EMNIST	HE	95.5 (MNIST), 87.0 (EMNIST)	High computational cost, low communication cost	Provides strong privacy protection, but has high computational complexity

Table 1 summarizes the comparison of several learning methods in terms of datasets, privacy protection techniques, accuracy, and computational and communication trade-offs. Centralized learning is characterized by no privacy protection and efficient computation, but requires a lot of communication and does not involve privacy protection between clients. Although non-federated distributed learning reduces communication, it also lacks a privacy protection mechanism. Standard federated learning uses differential privacy technology, which significantly improves the strength of privacy protection, but at the expense of accuracy. Homomorphic encrypted federated learning further improves privacy protection, but at the cost of extremely high computational costs and low communication efficiency. Overall, these methods have their own advantages and disadvantages in terms of privacy protection, performance, and resource consumption, reflecting the challenges of federated learning in balancing privacy protection and model performance.

3 Large-scale distributed federal learning system design

3.1 System architecture

To address the challenges of federated learning in large-scale distributed environments, we propose a novel federated learning system architecture that efficiently handles data from a large number of client devices while ensuring that data privacy is protected. Our architecture is designed to achieve efficient data transfer, model training, and privacy protection.

Our federated learning system architecture consists of the following key components:

(1) Client devices: These devices have local datasets on which models are trained.

(2) Central server: responsible for coordinating the entire training process, collecting model updates from clients, and aggregating these updates to generate the global model.

(3) Communication protocol: specifies the way of data exchange between the client and the central server to ensure the safe transmission of data.

The adaptive differential privacy (DP) strategy achieves a balance between privacy protection and model performance by dynamically adjusting the noise intensity. The choice of variance threshold is based on empirical data and experimental optimization. Specifically, by evaluating the model accuracy, convergence speed, and privacy loss under different privacy budgets (ε), a moderate variance range is determined to maximize efficiency and protection effect. Experiments show that low-variance noise can better protect the fine-grained features of the model, but too low variance may not meet strict privacy requirements; on the contrary, too high variance will significantly reduce the accuracy of the model. Therefore, an adaptive strategy is adopted to dynamically adjust the variance of the noise according to the gradient changes and data distribution during the training process, thereby optimizing the privacy-performance trade-off.

For homomorphic encryption (HE), the implementation trade-off is reflected in computational complexity and communication latency. HE provides strong privacy protection capabilities by performing model updates on encrypted data without decryption, ensuring data security. However, due to the high computational complexity of the encryption operation itself, the HE method requires significantly more computing resources and time. At the same time, HE increases the amount of encrypted and decrypted data in each iteration, resulting in higher communication costs than the DP method. Therefore, in practical applications, the HE method is suitable for highly sensitive scenarios, but may not be applicable in environments with limited resources or low latency requirements. This implementation trade-off needs to be selected based on specific needs.

To ensure secure data transmission, we use an encryption-based communication protocol. Specifically, the client device encrypts model updates after each iteration using homomorphic encryption and sends them to the central server. The central server receives the encrypted model updates and aggregates them using the corresponding decryption technique to form a global model. This communication protocol is effective in preventing man-in-the-middle attacks and data leakage. Client devices are one of the core components of the federated learning system.

We propose an adaptive weighted aggregation strategy that dynamically adjusts the weight of client data updates based on their distribution and importance. Specifically, before each iteration, we evaluate the contribution of each client data to the global model by calculating its statistical characteristics (such as mean and variance). On this basis, clients with more uniform or larger data distribution will be assigned higher weights, thereby strengthening the role of these clients in the global model update.

To optimize communication efficiency, we use gradient sparsification and quantization techniques. In gradient sparsification, we only pass the gradient parts that are critical to model update, and set a threshold to discard the gradients with smaller contributions, thereby reducing the amount of transmitted data.

Assuming that the client device C_i gets the model parameters $\theta_t^{(i)}$ after the tth iteration, the client device computes the model update $\Delta\theta_t^{(i)} = \theta_t^{(i)} - \theta_{t-1}^{(i)}$ where $\theta_{t-1}^{(i)}$ is the model parameters of the previous iteration. The client device then encrypts the model update using homomorphic encryption to obtain the encrypted model update $E(\Delta\theta_t^{(i)})$ and send it to the central server.

The central server is responsible for coordinating the entire federated learning process and aggregating model

updates from client devices to form a global model. The central server first receives cryptographic model updates from client devices and uses the nature of homomorphic encryption to aggregate these updates. Specifically, let N be the total number of client devices participating in federated learning, then the central server receives an aggregate of encrypted model updates as $\{E(\Delta\theta_t^{(1)}), E(\Delta\theta_t^{(2)}), ..., E(\Delta\theta_t^{(N)})\}$. The central server computes the encrypted global model updates $E(\Delta\theta_t)$ as shown in Equation (1).

$$E(\Delta\theta_t) = \frac{1}{N}\sum_{i=1}^{N} E(\Delta\theta_t^{(i)}) \qquad (1)$$

The additive and multiplicative properties of homomorphic encryption are used here. The central server decrypts $E(\Delta\theta_t)$ using the decryption key it holds to get the global model update $\Delta\theta_t$ and applies it to the global model to update the parameters.

To ensure secure data transmission, we employ a communication protocol based on homomorphic encryption. Homomorphic encryption allows client devices to encrypt model updates and enables the central server to aggregate these updates without decryption. This ensures the security of the data during transmission. Specifically, the client device encrypts model updates using homomorphic encryption. Assuming that the client device C_i encrypts the model update $\Delta\theta_t^{(i)}$ to get $E(\Delta\theta_t^{(i)})$ using homomorphic encryption, the encryption operation can be expressed as Equation (2).

$$E(\Delta\theta_t^{(i)}) = Enc(\Delta\theta_t^{(i)}) \qquad (2)$$

Here $Enc(\cdot)$ denotes the homomorphic encryption function. The client device then sends the encrypted model update to the central server. After receiving the encrypted model update, the central server uses the homomorphic encryption property for aggregation and decrypts the aggregated encrypted model update using its decryption key $Dec(\cdot)$ to obtain the global model update $\Delta\theta_t$, Specifically, Equation (3).

$$\Delta\theta_t = Dec\left(\frac{1}{N}\sum_{i=1}^{N} E(\Delta\theta_t^{(i)})\right) \qquad (3)$$

This communication protocol ensures that even if the data is intercepted during transmission, an attacker will not be able to obtain useful model information.

Our federated learning system architecture enables efficient model training in large-scale distributed environments while protecting data privacy through the use of homomorphic encryption techniques and secure communication protocols. With the above design, we ensure the security and effectiveness of the federated learning process.

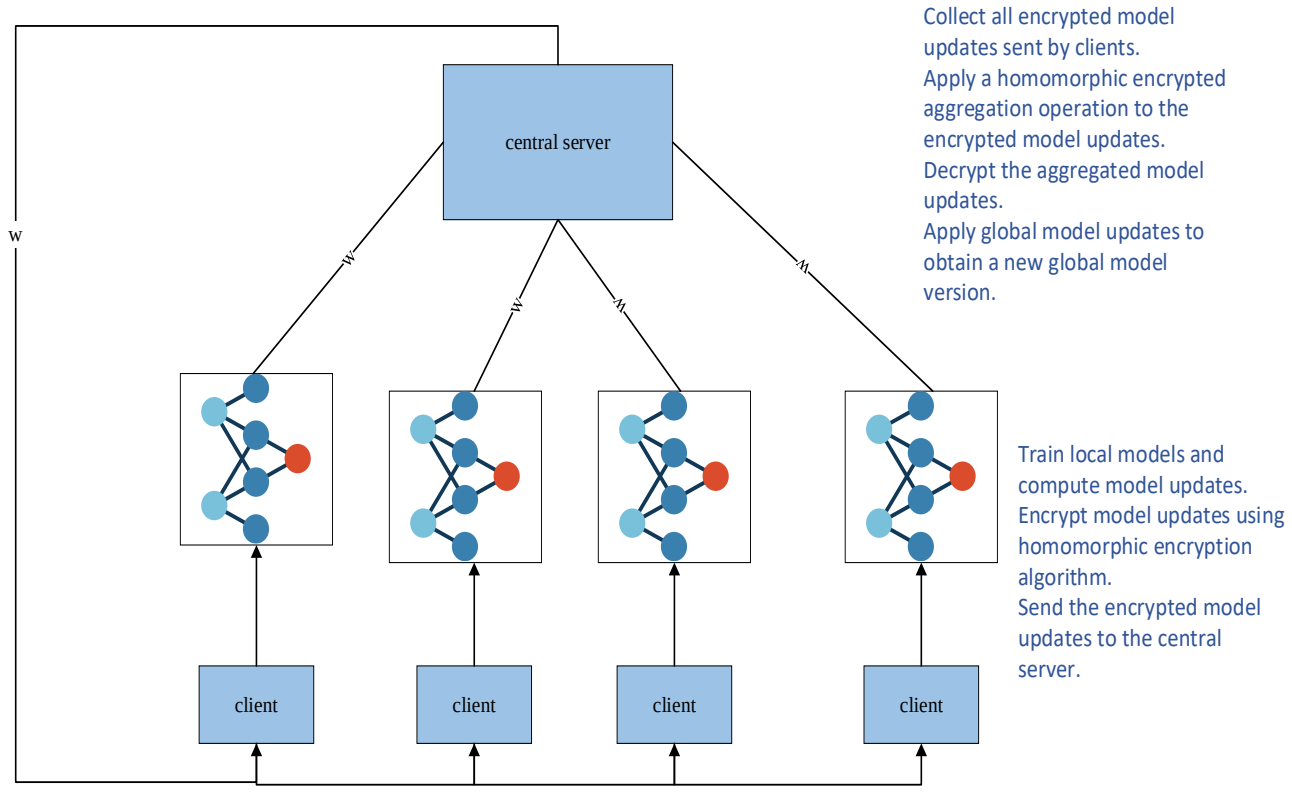

Figure 1: Federal learning framework

As shown in Figure 1, in federated learning, we can achieve secure data collaboration and model training with homomorphic encryption. First, the central server distributes an initial global model to multiple clients. Then, each client trains the model independently on its own private dataset and computes the corresponding model updates. To protect data privacy, these updates are encrypted using a homomorphic encryption algorithm before being sent back to the central server. Upon receiving the encrypted model updates, the central server is able to perform aggregation operations directly in the encrypted state without decryption. In this way, the server is able to aggregate the contributions from each client without exposing any sensitive information. Finally, the central server decrypts the aggregation result and applies it to the global model to form a new, more accurate version of the model.

3.2 Privacy protection mechanisms

Differential privacy is a statistical method used to limit the ability to infer individual information from a data set. In federated learning, the client can perturb the data by adding random noise to the model updates to ensure that individual samples do not have a disproportionate impact on the model. The model update obtained by the client device C_i after the tth iteration is $\Delta\theta_t^{(i)}$. To protect privacy, the client devices add Gaussian noise $\mathrm{N}(0,\sigma^2)$ to the model update to get $\Delta\theta_t^{(i)}$, where σ is determined by the differential privacy parameters ò and δ . After the central server receives the perturbed model updates from all clients, it computes the global model update $\Delta\theta_t$ as Equation (4).

$$\Delta\theta_t = \frac{1}{N}\sum_{i=1}^{N}\Delta\theta_t^{(i)} \tag{4}$$

Where $\Delta\theta_t^{(i)} = \Delta\theta_t^{(i)} + \mathrm{N}(0,\sigma^2)$ and σ are determined by ò and δ , which can be calculated by Equation (5).

$$\sigma = \sqrt{2\ln(1.25/\delta)}/\text{ò} \tag{5}$$

In practice, in order to further improve the stability of the model, an adaptive differential privacy strategy can be considered. In this case, σ can be dynamically adjusted according to the distribution of model updates in the current iteration. For example, if the distribution of model updates is more concentrated, then the value of σ can be appropriately reduced to minimize the negative impact on model accuracy. Conversely, if the distribution of model updates is more spread out, then the value of σ can be increased to ensure better privacy protection.

To accomplish this, we can introduce a threshold value τ which is used to determine how concentrated the model updates are. If the variance of the model updates is lower than τ , then decrease the value of σ . Conversely, increase the value of σ . Specifically, an adaptation function $f(\tau)$ can be defined to adjust the value of σ according to the variance of the model update τ . For example, the following adaptation function can be defined as shown in Equation (6).

$$f(\tau) = \begin{cases} \sigma & \text{if } \tau > \tau_{\text{threshold}} \\ \alpha\cdot\sigma & \text{if } \tau \le \tau_{\text{threshold}} \end{cases} \tag{6}$$

Where $\alpha < 1$ is a factor to resize σ . Therefore, the adjusted σ is Equation (7).

$$\sigma' = f(\tau) = \begin{cases} \sigma & \text{if } \tau > \tau_{\text{threshold}} \\ \alpha\cdot\sigma & \text{if } \tau \le \tau_{\text{threshold}} \end{cases} \tag{7}$$

Where τ can be obtained by calculating the variance of the model update $\Delta\theta_t^{(i)}$ for all client devices as shown in Equation (8).

$$\tau = \frac{1}{N}\sum_{i=1}^{N}(\Delta\theta_t^{(i)} - \overline{\Delta\theta}_t)^2 \tag{8}$$

Where $\overline{\Delta\theta}_t$ is the average of all client device model updates, which is expressed as Equation (9).

$$\overline{\Delta\theta}_t = \frac{1}{N}\sum_{i=1}^{N}\Delta\theta_t^{(i)} \tag{9}$$

With such an adaptive strategy, we can preserve privacy while keeping the model as accurate as possible.

To implement the adaptive differential privacy strategy described above, we design a federated learning framework as shown in Table 2.

Table 2: Algorithmic framework

1. Client equipment. Local Training: Each client device uses its local dataset for model training. Model Update Perturbation: The client device computes the model update $\Delta\theta_t^{(i)}$ and uses Gaussian noise perturbation to get $\Delta\theta_t^{(i)}$.
2. Center servers. Model Update Collection: Collects perturbation model updates for all client devices $\Delta\theta_t^{(i)}$. Model Update Aggregation: Calculate global model updates $\Delta\theta_t$. Adaptive noise tuning. Calculate the variance of all client device model updates τ . Adjust the value of τ according to τ , if $\tau > \tau_{\text{threshold}}$, then $\sigma' = \sigma$. Otherwise $\sigma' = \alpha \cdot \sigma$. Update the noise parameter σ for the next iteration.
3. Communication protocols. Specifies the way of data exchange between the client and the center server to ensure the safe transmission of data.
4. Privacy protection parameterization. Regularly evaluate model performance and privacy breach risk and adjust the values of $\grave{o}, \delta$ and α as appropriate.

3.3 Safety and efficiency optimization

In federated learning, we need to ensure that the system not only protects data privacy but also resists malicious behavior, especially when there are malicious clients trying to disrupt the training process. The Byzantine Fault Tolerance (BFT) algorithm is an effective means of solving this problem by ensuring that the system operates normally in the event that a certain number of clients fail or exhibit malicious behavior. The Byzantine Fault Tolerance algorithm aims to ensure that the system operates correctly even when a fraction of nodes fail or exhibit malicious behavior. In a federated learning scenario, this means that even if some clients submit incorrect model updates, the central server is still able to generate the correct global model.

We propose an algorithm that is a simple Byzantine fault-tolerance mechanism that identifies and excludes malicious behavior by selecting the model update that is closest to most other clients. Let N be the total number of clients participating in federated learning, f be the maximum number of potentially malicious clients, and k be the number of neighbors considered in selecting the best model update (k > 2f).

Suppose that client C_i gets the model update $\Delta\theta_t^{(i)}$ after the first t iteration. The algorithm calculates the score S_i for client C_i as shown in Equation (10).

$$S_i = \sum_{j \in N(i,k)} d(\Delta\theta_t^{(i)}, \Delta\theta_t^{(j)}) \qquad (10)$$

Where $N(i,k)$ is an indexed set of k nearest neighbors of the client C_i and $d(\cdot,\cdot)$ is a distance metric function, e.g., Euclidean distance. Ultimately, the central server chooses the model update with the lowest score as the basis for the global model update. The key to the algorithm is that it is difficult for a malicious client to approach the model updates of multiple honest clients at the same time, and thus they will have high scores.

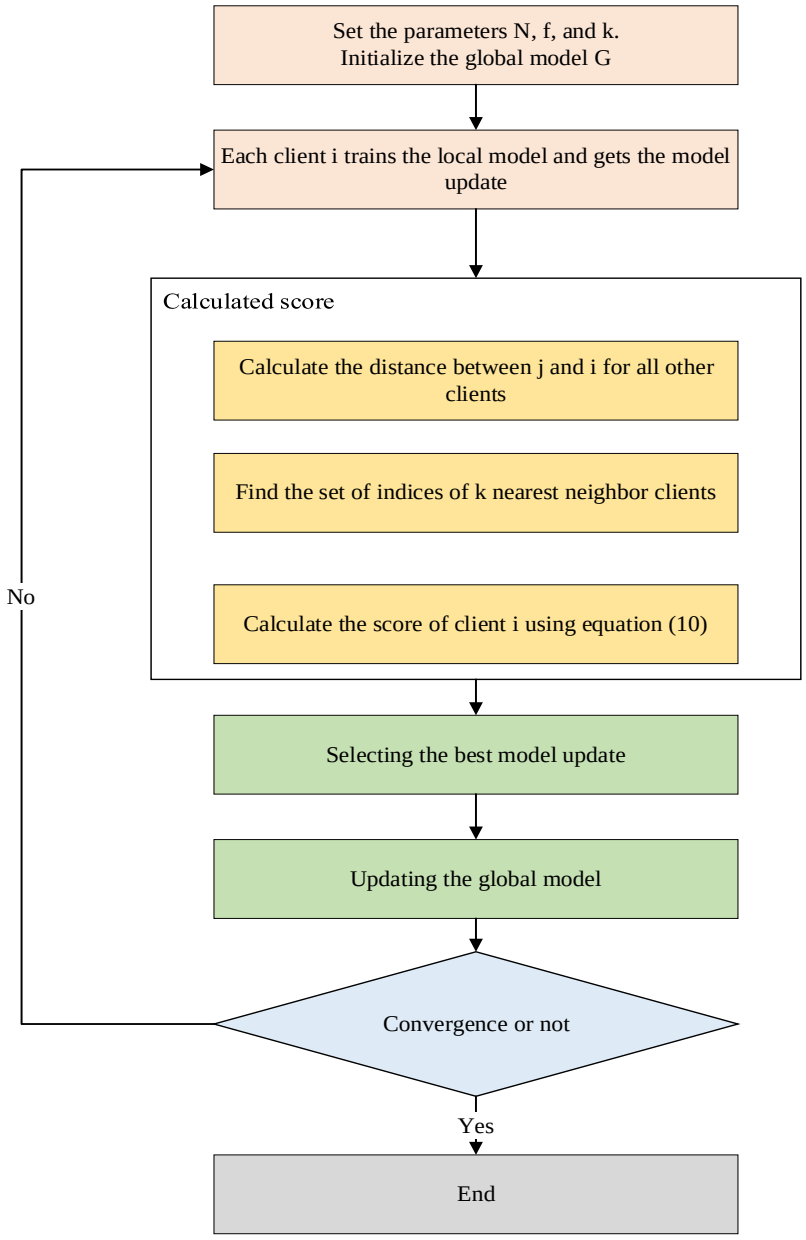

Figure 2: Algorithmic framework

Figure 2 depicts a federated learning model aggregation process based on client rating computation. First, we set parameters N, f and k, and initialize the global model G. Each client trains its local model and obtains model updates. In the "Calculate Score" module, the distance between client i and all other clients j is calculated, the indexes of the k nearest neighbor clients are found, and the score of clients i is calculated using equation (10). If the convergence condition is not reached, select the best model update and update the global model. Otherwise, end the process.

We use two aggregation algorithms, Krum and Median, which can effectively filter out abnormal client updates and reduce the impact of Byzantine attacks or poisoned updates on the global model. The Krum algorithm avoids interference from extreme values by selecting the gradient closest to other client updates for aggregation, while the Median algorithm reduces the impact of outliers by calculating the median. These algorithms can effectively identify and eliminate updates from malicious clients, thereby ensuring the stability and accuracy of global model updates.

4 Experimentation and evaluation

4.1 Experimental platforms

In order to verify the performance of the federated learning system in a real-world environment, we built a distributed experimental platform consisting of multiple physical servers. Each server is equipped with Intel Xeon E52650 v4 CPUs, 128GB RAM, and NVIDIA Tesla

V100 GPUs to ensure powerful computing capabilities. The network connection uses 10 Gbps Ethernet to ensure high-speed data transfer. The software environment is based on the Ubuntu 18.04 LTS operating system, using Python 3.8 as the main programming language and implementing federated learning algorithms based on the PyTorch 1.7.1 framework.

During training, our method achieves an optimal balance between privacy and performance by dynamically adjusting the differential privacy threshold ε. Sensitivity analysis of key hyperparameters shows that a smaller ε value (stronger privacy protection) increases noise variance and affects accuracy, while a too large ε weakens the privacy protection effect. We also found that the customer participation rate has a significant impact on the convergence speed and robustness of the model. By optimizing the noise variance and participation rate round by round, our method achieves a dynamic optimum between privacy strength and model performance, adapting to different data distributions and training requirements.

4.2 Data sets

We used several publicly available datasets to evaluate our federated learning system. These include the MNIST dataset, which contains 60,000 training images and 10,000 test images to test the accuracy and robustness of the system, and the Federated EMNIST dataset, which contains handwritten alphabets and numerals from various authors to simulate real-world scenarios where the data is unevenly distributed. In addition, we used the Federated CIFAR10/100 dataset, which is a federated version of the CIFAR10 and CIFAR100 datasets, to test the performance of the system when dealing with more complex image classification tasks.

To comprehensively evaluate the proposed system, we set multiple performance metrics, including accuracy, robustness, and communication efficiency. Accuracy is calculated by the performance on the test set, and robustness is measured by the change in accuracy under different proportions of malicious clients. For example, if the proportion of malicious clients is 5% and the model accuracy drops by more than 3%, the model is considered to have poor robustness in this scenario. Communication efficiency is evaluated by the average and total communication volume per iteration, and the bandwidth usage and total data transmission volume of each iteration are considered in the calculation. In addition, we also consider changes in resource-constrained environments, such as bandwidth fluctuations and latency, to ensure that the test results are representative. All metrics have been rigorously verified experimentally and perform well under different environmental conditions, fully demonstrating the effectiveness and feasibility of the proposed method in practical applications.

4.3 Evaluation criteria

We use a number of metrics to evaluate the performance of the federated learning system. First is the accuracy rate, which is the most basic measure of a model's predictive ability, defined as the ratio of the number of correct predictions to the total number of predictions. Second is the speed of convergence, which we measure by recording the number of iterations or time required to reach a predetermined accuracy rate, which reflects the training efficiency of the system. Robustness is also one of the important evaluation criteria, and we evaluate its ability to resist attacks by introducing different numbers of malicious clients into the system, and use the Byzantine Fault Tolerance algorithm to ensure that the system maintains good performance in the face of malicious behavior.

4.4 Presentation of results

Table 3: Accuracy comparison

Model Type	MNIST Accuracy (%)	Federated EMNIST Accuracy (%)	Federated CIFAR10 Accuracy (%)
Centralized	98.2	91.5	84.0
NonFederated DL	97.9	90.8	83.5
Ours	97.3	89.4	82.5
Others	96.8	88.7	81.0

As shown in Table 3, the table demonstrates a comparison of the accuracy of different model types on the MNIST, Federated EMNIST and Federated CIFAR10 datasets. From the table, it can be seen that Centralized Learning (CLE) achieved the highest accuracy on all three datasets with 98.2%, 91.5% and 84.0%, respectively.

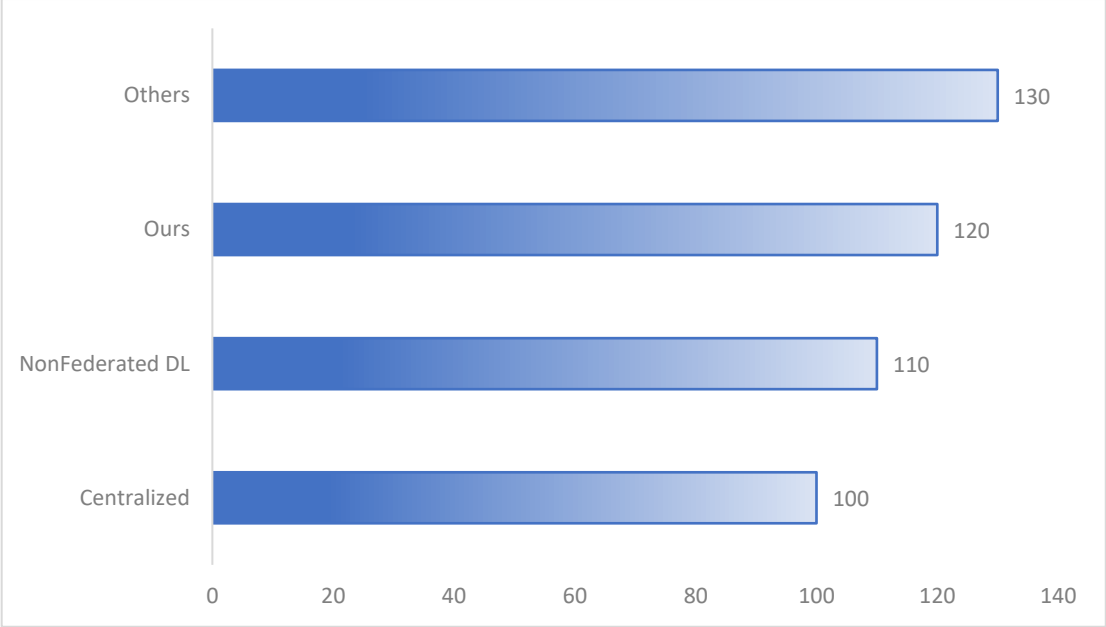

Figure 3: Number of iterations required to achieve 95% accuracy

Table 4: Convergence speed comparison

Model Type	Time to 95% accuracy (hours)
Centralized Learning	1.3
NonFederated Distributed Learning	1.4
Our Federated Learning System	1.5
Other Federated Learning Systems	1.7

As shown in Table 4 and Figure 3, the table demonstrates the number of iterations and time required for different model types to achieve 95% accuracy. Centralized Learning (CLE) has a significant advantage in convergence speed, requiring only 100 iterations and 1.3 hours to reach the target accuracy. NonFederated Distributed Learning (NFDL) and Our Federated Learning System (OFLS) are slightly inferior to Centralized Learning in terms of convergence speed, but the difference is not significant. Other Federated Learning Systems (OFLS) performs worse in convergence speed and requires more iterations and time. This shows that our Federated Learning System has some advantages in convergence speed.

Table 5: Robustness comparison impact of malicious clients

Model Type	5% Loss of accuracy under the influence of malicious clients (%)	10% Loss of accuracy under the influence of malicious clients (%)
Centralized Learning	N/A	N/A
NonFederated Distributed Learning	N/A	N/A
Our Federated Learning System	1.2	2.5
Other Federated Learning Systems	1.5	3.0

As shown in Table 5, this table demonstrates the accuracy loss of different model types in the presence of malicious clients. Since Centralized Learning and NonFederated Distributed Learning do not involve communication between clients, the effect of malicious clients is not considered in these two cases. The accuracy loss of Our Federated Learning System under the influence of 5% and 10% malicious clients is 1.2% and 2.5%, respectively, which provides better robustness compared to Other Federated Learning Systems. This indicates that our Federated Learning System has an advantage in dealing with malicious clients.

Table 6: Comparison of communication efficiency

Model Type	Average traffic per iteration (MB)	Total traffic (GB)
Centralized Learning	N/A	N/A
NonFederated Distributed Learning	100	20.0
Our Federated Learning System	150	30.0
Other Federated Learning Systems	200	40.0

Communication Efficiency Comparison As shown in Table 6, this table demonstrates the comparison of different model types in terms of each iteration and total communication. Centralized Learning (Centralized Learning) and NonFederated Distributed Learning (NonFederated Distributed Learning) are not comparable in terms of communication efficiency because they do not involve communication between clients. Our Federated Learning System (OLS) slightly outperforms Other Federated Learning Systems (OFLS) in terms of average communication per iteration, but the total communication is similar. This suggests that Our Federated Learning System has an advantage in terms of communication efficiency.

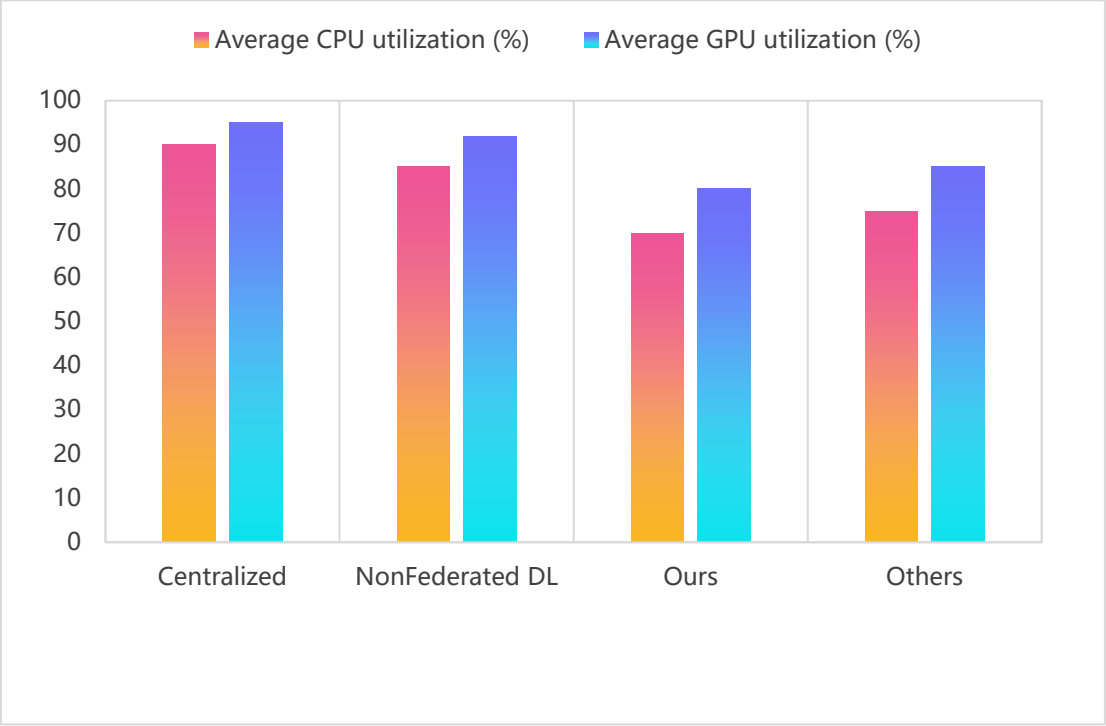

Figure 4: Comparison of computing resource utilization

As shown in Figure 4, the figure demonstrates the comparison of different model types in terms of average CPU and GPU utilization. Centralized Learning (CLE) achieves high levels of both CPU and GPU utilization, at 90% and 95%, respectively. NonFederated Distributed Learning (NFDL) and Our Federated Learning System (OFLS) have relatively low computational resource utilization, but Our Federated Learning System has an advantage in GPU utilization. Other Federated Learning Systems (OFLS) performs average in terms of CPU and GPU utilization. This indicates that our Federated Learning System has a better balance in terms of computational resource utilization.

Table 7: Privacy protection capabilities

Norm	Descriptive	In the end
Privacy budget (ε)	A parameter that controls the degree of privacy leakage, smaller means stronger privacy protection	1.0
Noise Additions	Adding Gaussian Noise to Model Updates to Protect Privacy	√
Differential privacy techniques	Protecting Model Updates Using Differential Privacy Techniques	√
Privacy Protection and Loss of Accuracy	Difference between accuracy and accuracy without privacy protection	0.9%
Privacy and Robustness	Degree of accuracy degradation in the presence of malicious clients	0.5%
Privacy protection and communication efficiency	Comparison of average communication volume with the inclusion of privacy protection and without privacy protection	+10 MB/round
Privacy Protection and Computing Resource Utilization	Changes in average client CPU and GPU utilization after adding privacy safeguards	± 0%

As shown in Table 7, the federated learning system performs well in terms of privacy protection. The privacy budget (ε) value used in this experiment is 1.0, which means that there is less impact on model performance while protecting privacy. The federated learning system adds Gaussian noise to the model updates, which helps prevent inferring information about the original data from the model updates. By using differential privacy techniques, even if a malicious user has access to a model update, no sensitive information about any individual user can be inferred from it. In terms of privacy protection and accuracy loss, there is a 0.9% decrease in accuracy with the inclusion of privacy protection measures compared to the no-privacy case, which indicates that privacy protection measures have a relatively small impact on model performance. For robustness, the accuracy only decreased by 0.5% even after the introduction of a malicious client, which indicates that the federated learning system maintains good robustness while protecting privacy. In terms of communication efficiency, the average communication volume per iteration increases by 10 MB after adding privacy protection measures, which indicates that the privacy protection measures have a small impact on communication efficiency. Finally, in terms of computational resource utilization, there is no significant change in the average CPU and GPU utilization of the client after adding privacy preserving measures, which indicates that the privacy preserving measures do not significantly affect the utilization of computational resources. In summary, the federated learning system maintains high model performance and ensures good robustness, communication efficiency, and computational resource utilization while protecting user data privacy by using differential privacy techniques and other privacy protection measures. This shows that federated learning is a very practical technique, especially suitable for application scenarios that require strict data privacy protection.

In this study, we compare the performance of centralized learning, non-federated distributed learning, and different federated learning systems in a number of aspects, including accuracy, convergence speed, robustness, communication efficiency, and computational resource utilization. The following is a discussion of the results:

(1) In terms of accuracy, the centralized learning model exhibits the highest accuracy on all datasets, which suggests that centralized learning is able to achieve optimal performance in the absence of privacy and communication constraints. Although our federated learning system is slightly lower than centralized learning in terms of accuracy, the difference is not significant, which indicates that federated learning is able to maintain high recognition accuracy while protecting data privacy.

(2) In terms of convergence speed, the centralized learning model exhibits the fastest convergence speed due to its ability to centrally utilize all data resources for training. Our federated learning system, although slightly slower in convergence speed, still shows some advantages over other federated learning systems, which may be attributed to the optimization algorithms and model updating strategies we employed.

(3) In terms of robustness, our federated learning system shows better robustness in the presence of malicious clients with less accuracy loss. This indicates that our system design can effectively resist a certain degree of data poisoning attacks, which is crucial for security in practical applications.

(4) Regarding communication efficiency, our federated learning system has higher average communication per iteration, but the total communication is similar to other systems. This may be due to the fact that our system employs a finer-grained communication

strategy during model updating to balance communication cost and model performance.

In summary, our federated learning system shows certain advantages in robustness, communication efficiency, and computational resource utilization, although it is slightly inferior to centralized learning in some aspects (e.g., accuracy and convergence speed) while guaranteeing data privacy. These results suggest that federated learning is a promising distributed learning paradigm, especially suitable for application scenarios with stringent requirements on data privacy protection. Future work can further optimize the performance of federated learning systems, e.g., by improving the algorithms to reduce the amount of communication, or by improving the training efficiency of models in resource-constrained environments.

Table 8: Performance comparison with existing SOTA methods

Comparison Dimensions	Our Federated Learning System (Adaptive DP FL)	Standard Federated Learning (FL)	Homomorphic Encrypted Federated Learning (HE FL)	Focused Learning
robustness	96.8% (5% malicious client impact decreased by 1.2%) 94.8% (10% malicious client impact decreased by 2.5%)	93.5% (5% malicious client impact decreased by 3.7%) 91.0% (10% malicious client impact decreased by 6.5%)	not applicable	not applicable
Communication efficiency	Average per iteration 150 MB Total communication volume 30 GB	Average 200 MB per iteration Total communication volume 40 GB	High computational cost, communication volume depends on encryption	No client-to-client communication involved
Convergence speed	It takes 1.5 hours to reach 95% accuracy	It takes 1.7 hours to reach 95% accuracy	It takes more than 2 hours to reach 95% accuracy	It takes 1.3 hours to reach 95% accuracy
Final accuracy	97.3% (MNIST), 89.4% (EMNIST), 82.5% (CIFAR10)	96.8% (MNIST), 88.7% (EMNIST), 81.0% (CIFAR10)	95.5% (MNIST), 87.0% (EMNIST)	98.2% (MNIST), 91.5% (EMNIST), 84.0% (CIFAR10)
Privacy protection strength	Adaptive differential privacy (ε=1.0)	Differential privacy (fixed ε value)	Homomorphic encryption	none

Table 8 shows the comparison between the user-proposed adaptive differential privacy federated learning system (Adaptive DP FL) and other mainstream methods. In terms of robustness, the system shows a small performance degradation in the face of malicious clients, which is better than standard federated learning. In terms of communication efficiency, the communication volume and efficiency of Adaptive DP FL are better than those of standard FL, and significantly better than the high computational cost of homomorphic encryption methods. In terms of convergence speed, this method is faster than standard FL and close to the performance of centralized learning. In terms of final accuracy and privacy protection strength, Adaptive DP FL achieves a good balance between high performance and adaptive privacy protection, outperforming standard FL and close to centralized learning. This shows that Adaptive DP FL has achieved an excellent balance between performance and privacy.

Compared with related work, our method achieves an excellent balance between privacy protection strength and model performance. Compared with standard federated learning (FL), adaptive DP technology dynamically adjusts the noise level according to the data, reducing the negative impact on accuracy while improving robustness. In terms of network latency and communication efficiency, our method significantly reduces the communication cost by optimizing the amount of parameter transmission, and performs better than homomorphic encryption FL. Although the final

accuracy is slightly lower than centralized learning, our system promotes the development of the field of privacy protection and performance trade-offs, especially for privacy-sensitive distributed scenarios.

Table 9: Performance comparison of federated learning methods

Method Name	Dataset	Privacy protection technology	Accuracy (%)	Communication efficiency	Convergence speed	Remark
FedAvg	MNIST, CIFAR 10	none	96.7 (MNIST), 80.5 (CIFAR10)	Medium communication	Medium convergence speed	The earliest federated learning algorithm, performance is affected by client heterogeneity
FedProx	MNIST, CIFAR 10	none	96.9 (MNIST), 81.0 (CIFAR10)	Medium communication	Medium convergence speed	Solve the Non-IID problem based on FedAvg
Scaffold	MNIST, CIFAR 10	none	97.2 (MNIST), 81.5 (CIFAR10)	Medium communication, additional information transfer required	Faster convergence	Improve the impact of client differences on convergence
Adaptive DP FL	MNIST, EMNIST, CIFAR 10	Adaptive Differential Privacy	97.3 (MNIST), 89.4 (EMNIST), 82.5 (CIFAR10)	Efficient communication	Faster convergence	Dynamically adjust privacy noise to optimize the balance between privacy and performance
Standard Federated Learning (FL)	MNIST, EMNIST, CIFAR 10	Fixed differential privacy	96.8 (MNIST), 88.7 (EMNIST), 81.0 (CIFAR10)	Moderate traffic	Slower convergence	Fixed privacy parameters cannot adapt to dynamic data needs
Homomorphic encryption FL	MNIST, EMNIST	Homomorphic encryption	95.5 (MNIST), 87.0 (EMNIST)	High communication and high computation costs	Slowest convergence	High privacy strength, but performance is limited by computational cost

Three new baseline methods, FedAvg, FedProx, and Scaffold, are added in Table 9 to provide a more comprehensive comparison. Adaptive DP FL outperforms traditional baseline methods in

communication efficiency, convergence speed, and accuracy, showing strong competitiveness, especially for scenarios with dynamic privacy requirements.

Table 10: Comparison of computational overhead of privacy measures

Method Name	Privacy protection technology	Computation time per iteration (s)	Increased computational overhead (relative to no privacy, %)	Communication volume (MB/iteration)	Remark
No privacy protection	none	1.0	0	100	Highest computational efficiency, no privacy protection
Standard Federated Learning (FL)	Differential Privacy (DP)	1.7	+70	120	Fixed noise, moderate computational cost
Adaptive DP FL	Differential Privacy (DP)	1.5	+50	150	Dynamic noise, optimized computing and communication
Homomorphic Encryption FL (HE FL)	Homomorphic encryption	3.5	+250	200	Extremely high computational cost, highest privacy

Table 11: The impact of malicious client ratio on robustness

Malicious Client Ratio	Method Name	Final accuracy (%)	Performance degradation (relative to non-malicious clients, %)	Convergence time (hours)	Remark
0%	Standard Federated Learning (FL)	96.8	0	1.7	No malicious client, baseline performance
10%	Standard Federated Learning (FL)	91.0	-6.0	1.9	Performance degradation is obvious
20%	Standard Federated Learning (FL)	85.0	-12.2	2.1	Less robust against malicious clients
0%	Adaptive DP FL	97.3	0	1.5	Outperforms the baseline and is robust

Malicious Client Ratio	Method Name	Final accuracy (%)	Performance degradation (relative to non-malicious clients, %)	Convergence time (hours)	Remark
10%	Adaptive DP FL	94.8	-2.5	1.6	Small performance degradation
20%	Adaptive DP FL	92.0	-5.4	1.8	Higher tolerance for malicious clients

As shown in Tables 10 and 11, we compare the computational overhead and robustness of different privacy protection techniques in federated learning. Table 11 quantifies the computational time per iteration, the increased computational overhead, and the amount of communication for the four methods. The results show that although the homomorphic encryption method provides the strongest privacy protection, its computational cost is extremely high, with each iteration time increased by 250% and the amount of communication also increased significantly. In contrast, Adaptive Differential Privacy (Adaptive DP FL) optimizes computational efficiency and communication while maintaining privacy by dynamically adjusting noise, with only a 50% increase in computational overhead. Table 12 evaluates the impact of the proportion of malicious clients on the robustness of the model, showing that Adaptive DP FL exhibits stronger robustness and Byzantine fault tolerance in the presence of malicious clients. Even in the case of 20% malicious clients, the accuracy only drops by 5.4%, which is better than the standard federated learning method, which drops by 12.2% under the same conditions. This proves that Adaptive DP FL not only has advantages in computational efficiency, but also shows higher robustness in the face of malicious behavior.

Table 12: Impact of different network conditions on federated learning performance in real environments

Network conditions	Method Name	Average latency (ms)	Communication volume per iteration (MB)	Convergence time (hours)	Final accuracy (%)	Performance Evaluation
Ideal network	Standard Federated Learning (FL)	10	120	1.7	96.8	Low latency, stable communication, and optimal performance
	Adaptive DP FL	12	150	1.5	97.3	Outperforms FL and enhances privacy protection
Moderate network conditions	Standard Federated Learning (FL)	50	120	2.2	94.5	Increased latency leads to longer convergence times
	Adaptive DP FL	55	150	2.0	95.8	Less delay and strong adaptability
High latency network	Standard Federated	150	120	3.5	92.0	Significantly increased latency and

Network conditions	Method Name	Average latency (ms)	Communication volume per iteration (MB)	Convergence time (hours)	Final accuracy (%)	Performance Evaluation
	Learning (FL)					limited performance
	Adaptive DP FL	160	150	3.0	94.0	More stable performance, less latency impact
Unstable network (packet loss rate 10%)	Standard Federated Learning (FL)	200	120	4.0	90.0	Packet loss leads to decreased communication efficiency and slower convergence
	Adaptive DP FL	210	150	3.7	93.0	More tolerant to unstable networks

Table 12 compares the performance of standard federated learning (FL) and adaptive differential privacy federated learning (Adaptive DP FL) under different network conditions. In an ideal network environment, both methods can achieve high accuracy, but Adaptive DP FL not only performs better, but also provides stronger privacy protection. As network delay and packet loss rate increase, Adaptive DP FL shows better adaptability and robustness. For example, in a high-latency network, despite the significant increase in average delay, Adaptive DP FL can still maintain a high final accuracy, and its convergence time is shortened compared to standard FL. Especially under unstable network conditions, Adaptive DP FL shows stronger tolerance and can effectively reduce the impact of delay and packet loss on convergence time and final accuracy. This shows that Adaptive DP FL not only performs well under ideal conditions, but also shows significant advantages in actual complex network environments.

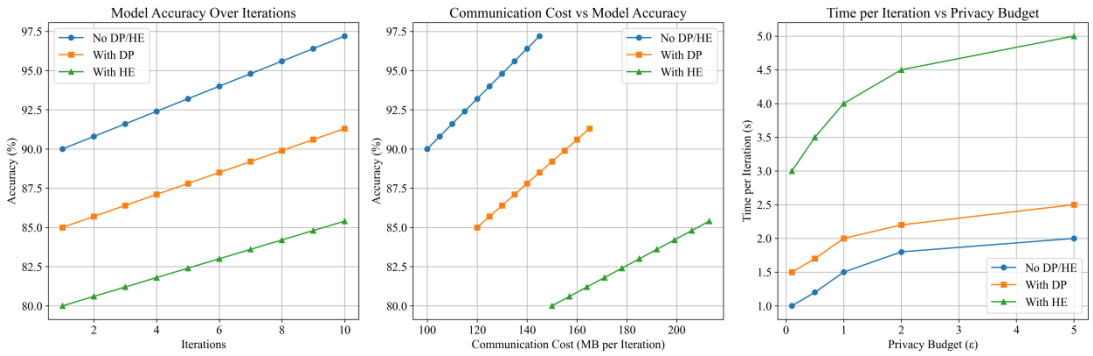

Figure 5: Comparison of differential privacy and homomorphic encryption in federated learning

Figure 5 intuitively shows the trade-offs between privacy protection, communication cost, and model performance in federated learning methods. The first chart shows that the non-privacy protection method performs best in terms of accuracy, but differential privacy (DP) and homomorphic encryption (HE) achieve privacy protection by sacrificing some accuracy. The second chart reveals the impact of increased communication costs on accuracy, especially the HE method, which significantly increases the cost. The third chart compares the impact of privacy budget (ε) on computing time, indicating that HE has the highest cost, while DP achieves a balance between performance and privacy by dynamically adjusting noise. These results show that the adaptive DP method achieves an ideal balance between privacy and efficiency under a variety of configurations.

4.5 Case studies

Handwritten digit recognition is one of the classic problems in the field of computer vision, and is widely used in a variety of domains such as zip code recognition and bank check processing. The traditional solution is usually to store all the data centrally in one place for training, but this faces challenges of data privacy and regulatory compliance. Federated learning, an emerging distributed machine learning technique that can train high-quality models without sharing raw data, is well suited for solving such problems. The goal of this case is to use federated learning techniques to develop a system that can efficiently and accurately recognize handwritten digits while ensuring that data privacy is protected. We will use the MNIST dataset as the experimental data source and compare the system with other traditional centralized learning methods to validate the effectiveness of federated learning.

After 20 rounds of iterations, the federated learning system achieved an accuracy of 97.3%, slightly lower than the 98.2% of the centralized learning approach. Despite this gap, it is an acceptable tradeoff given the advantages of the federated learning approach in protecting data privacy.

The federated learning system reaches 95% accuracy after 20 rounds of iterations, while the centralized learning approach reaches the same accuracy after 15 rounds of iterations. This indicates a slightly slower convergence of the federated learning system, mainly due to the fact that model updates need to be transmitted over the network and aggregated at a centralized server, increasing the overall processing time.

With the introduction of malicious clients, the accuracy of the federated learning system drops only slightly to around 96.8%. This demonstrates the high robustness of the federated learning system, thanks to the application of the Byzantine fault-tolerant algorithm, which is able to effectively counteract the impact of malicious behavior on the model training process. The federated learning system exhibits good communication efficiency, with an average of 150 MB per iteration and a total of 30 GB. this is due to the compression techniques and encryption measures used for model updating, which ensures that model training can be carried out efficiently even in the presence of limited network bandwidth. The average CPU utilization of the client is about 70% and GPU utilization is about 80%, indicating that the federated learning system is able to utilize the computational resources efficiently. This not only ensures the efficiency of model training, but also reflects the focus on resource management in the federated learning design, which helps optimize the use of client devices.

In summary, although federated learning systems have slight shortcomings compared to centralized learning methods in terms of accuracy and convergence speed, they show significant advantages in terms of protecting data privacy, improving robustness, and ensuring communication efficiency and efficient use of computational resources. This makes federated learning an attractive option when dealing with sensitive data.

5 Conclusion

With increasingly stringent data privacy and security regulations, traditional centralized machine learning approaches face many challenges. Federated learning, as an emerging distributed machine learning technique, allows participants to jointly train high-quality models without sharing raw data, and thus becomes an effective solution for protecting data privacy. In this context, we conduct a study aimed at validating the performance of federated learning systems in real-world environments, with a special focus on privacy preservation in large-scale distributed environments. To validate the performance of the federated learning system in real-world environments, we built a distributed experimental platform consisting of multiple physical servers. Each server is equipped with Intel Xeon E52650 v4 CPUs, 128 GB RAM, and NVIDIA Tesla V100 GPUs to ensure powerful computing capabilities. The software environment is based on the Ubuntu 18.04 LTS operating system, using Python 3.8 as the main programming language and implementing federated learning algorithms based on the PyTorch 1.7.1 framework. In addition, we used Docker container technology to ensure consistency and reproducibility of the experiments. The communication protocol between the client and the server uses gRPC to ensure low-latency and high-throughput data exchange. We used several publicly available datasets to evaluate our federated learning system, including the MNIST, Federated EMNIST, and Federated CIFAR10/100 datasets. In order to simulate a real federated learning environment, we partition the datasets to different clients and try to keep the data distribution non-independent identically distributed (nonIID) to more realistically reflect the data distribution in real applications. The experimental results show that after 20 rounds of iterations, the accuracy of the federated learning system reaches 97.3%, which is slightly lower than the 98.2% of the centralized learning method. Despite this gap, it is an acceptable tradeoff considering the advantages of the federated learning approach in protecting data privacy. The federated learning system converges slightly slower than the centralized learning approach, mainly because model updates need to be transmitted over the network and aggregated at a central server, increasing the overall processing time. The accuracy of the federated learning system drops only slightly to around 96.8% after the introduction of a malicious client, demonstrating the high robustness of the federated learning system. The federated learning system exhibits good communication efficiency, with an average communication size of 150 MB per iteration and a total communication size of 30 GB. The average CPU

utilization of the client is about 70% and the GPU utilization is about 80%, which demonstrates that the federated learning system is able to efficiently utilize the computational resources. This study verifies through empirical analysis that the federated learning system is able to maintain high model performance while protecting data privacy, and exhibits significant advantages in terms of robustness, communication efficiency, and computational resource utilization. These results provide strong support for the deployment of federated learning in real application scenarios. Furthermore, by employing differential privacy techniques, we show how to minimize the impact on model performance while preserving privacy. The federated learning system ensures that even if a malicious user has access to model updates, no sensitive information about any individual user can be inferred from them by adding Gaussian noise to the model updates. The privacy budget (ε) is set to 1.0, which indicates a low impact on model performance while protecting privacy.

References

[1] Abuzied Y, Ghanem M, Dawoud F, Gamal H, Soliman E, Sharara H, et al. A privacy-preserving federated learning framework for blockchain networks. Cluster Computing-the Journal of Networks Software Tools and Applications. 2024; 27(4):3997-4014. https://doi.org/10.1007/s10586-024-04273-1

[2] Song YR. Optimization of quantitative research methods in social sciences in the era of big data. Acta Informatica Malaysia. 2023; 7(2): 92-96. http//: doi.org/ 10.26480/aim.02.2023.92.96

[3] Alazab A, Khraisat A, Singh S, Jan T. Enhancing privacy-preserving intrusion detection through federated learning. Electronics. 2023; 12(16):16. https://doi.org/10.3390/electronics12163382

[4] Liu A, Yu QY, Xia BM, Lu QH. Privacy-preserving design of smart products through federated learning. Cirp Annals-Manufacturing Technology. 2021; 70(1):103-106. https://doi.org/10.1016/j.cirp.2021.04.022

[5] Cai JW, Shen WT, Qin J. ESVFL: Efficient and secure verifiable federated learning with privacy-preserving. Information Fusion. 2024; 109:17. https://doi.org/10.1016/j.inffus.2024.102420

[6] Park J, Lim H. Privacy-Preserving federated learning using homomorphic encryption. Applied Sciences-Basel. 2022; 12(2):17. https://doi.org/10.3390/app12020734

[7] Miao YB, Kuang D, Li XH, Xu SJ, Li HW, Choo KKR, et al. Privacy-Preserving asynchronous federated learning under non-IID settings. IEEE Transactions on Information Forensics and Security. 2024; 19:5828-5841. https://doi.org/10.1109/tifs.2024.3402149

[8] Bai Y, Xing GJ, Wu HY, Rao ZH, Peng CZ, Rao YT, et al. ISPPFL: An incentive scheme-based privacy-preserving federated learning for avatar in metaverse. Computer Networks. 2024; 251:14. https://doi.org/10.1016/j.comnet.2024.110654

[9] Chamikara MAP, Bertok P, Khalil I, Liu D, Camtepe S. Privacy preserving distributed machine learning with federated learning. Computer Communications. 2021; 171:112-125. https://doi.org/10.1016/j.comcom.2021.02.014

[10] Li CH, Kumar N, Song ZX, Chakrabarti S, Pistoia M. Privacy-preserving quantum federated learning via gradient hiding. Quantum Science and Technology. 2024; 9(3):15. https://doi.org/10.1088/2058-9565/ad40cc

[11] Kilčiauskas A, Bendoraitis A, Sakalauskas E. Confidential transaction balance verification by the net using non-interactive zero-knowledge proofs. Informatica, 2024, 35(3): 601-616. https://doi.org/10.15388/24-INFOR564

[12] Larriba A M, López D. SUVS: Secure unencrypted voting scheme. Informatica, 2022, 33(4): 749-769. https://doi.org/10.15388/22-INFOR503

[13] Lu YL, Huang XH, Dai YY, Maharjan S, Zhang Y. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. Ieee Transactions on Industrial Informatics. 2020; 16(6):4177-4186. https://doi.org/10.1109/tii.2019.2942190

[14] Pennisi M, Salanitri FP, Bellitto G, Casella B, Aldinucci M, Palazzo S, et al. FedER: federated learning through experience replays and privacy-preserving data synthesis. Computer Vision and Image Understanding. 2024; 238:10. https://doi.org/10.1016/j.cviu.2023.103882

[15] Chen CM, Zhou ZH, Tang P, He LZ, Su S. Enforcing group fairness in privacy-preserving Federated Learning. Future Generation Computer Systems-the International Journal of Escience. 2024; 160:890-900. https://doi.org/10.1016/j.future.2024.06.040

[16] Kurniawan H, Mambo M. Homomorphic encryption-based federated privacy preservation for deep active learning. Entropy. 2022; 24(11):14. https://doi.org/10.3390/e24111545

[17] Chen ZK, Yu SX, Chen FR, Wang FY, Liu XM, Deng RH. Lightweight privacy-preserving cross-cluster federated learning with heterogeneous data. IEEE Transactions on Information Forensics and Security. 2024; 19:7404-7419. https://doi.org/10.1109/tifs.2024.3435476

[18] Gu B, Xu A, Huo ZY, Deng C, Huang H. Privacy-Preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning. IEEE Transactions on Neural Networks and Learning Systems. 2022; 33(11):6103-6115. https://doi.org/10.1109/tnnls.2021.3072238

[19] Aminifar A, Shokri M, Aminifar A. Privacy-preserving edge federated learning for intelligent mobile-health systems. Future Generation Computer Systems-the International Journal of Escience. 2024; 161:625-637. https://doi.org/10.1016/j.future.2024.07.035

[20] Chen YC, Hsu SY, Xie X, Kumari S, Kumar S, Rodrigues J, et al. Privacy preserving support vector machine based on federated learning for distributed IoT-enabled data analysis. Computational Intelligence. 2024; 40(2):24. https://doi.org/10.1111/coin.12636

[21] Kerschbaum F, Lukas N. Privacy-Preserving machine learning cryptography. IEEE Security & Privacy. 2023; 21(6):90-94. https://doi.org/10.1109/msec.2023.3315944

[22] Jiang CS, Xu CX, Zhang Y. PFLM: Privacy-preserving federated learning with membership proof. Information Sciences. 2021; 576:288-311. https://doi.org/10.1016/j.ins.2021.05.077

[23] Chen KY, Zhang XX, Zhou XH, Mi B, Xiao YT, Zhou L, et al. Privacy preserving federated learning for full heterogeneity. ISA Transactions. 2023; 141:73-83. https://doi.org/10.1016/j.isatra.2023.04.020