2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN) | 979-8-3315-2042-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICMLCN64995.2025.11140049

APRA-DP: Differential Privacy based Adaptive Privacy Preserving Robust Aggregation Method for Federated Learning

1st Geming Xia
National University of Defense Technology
Changsha, China
xiageming@163.com

2nd Jian Chen
Huawei Technologies Co.
Changsha, China
chenjianyd@outlook.com

3rd Xinyi Huang
National University of Defense Technology
Changsha, China
huangxinyi@nudt.edu.cn

4th Jiawen Wu
National University of Defense Technology
Changsha, China
2305301027@qq.com

5th Hongwei Huang
National University of Defense Technology
Changsha, China
huanghongwei@nudt.edu.cn

6th Chaodong Yu
National University of Defense Technology
Changsha, China
chaodongyu16@nudt.edu.cn

7th Yuze Zhang
National University of Defense Technology
Changsha, China
zhangyuze2025@163.com

8th Zhiping Cai
National University of Defense Technology
Changsha, China
zpcai@nudt.edu.cn

Abstract—Federated learning is a popular approach in machine learning because it protects participants' data privacy and leverages collective knowledge to enhance model performance. However, it faces security vulnerabilities and privacy risks, such as feature inference attacks and poisoning attacks. Current methodologies often overlook the interdependence of privacy preservation and security fortification. We propose an adaptive differential privacy robust aggregation methodology that safeguards privacy and provides resilience against poisoning attacks. This involves implementing measures on the client side to mitigate the adverse effects of adding noise through differential privacy to locally trained models before transmission. Additionally, Byzantine robust aggregation strategies are applied on the server side to defend against poisoning attacks. Our proposed approach outperforms prevailing state-of-the-art algorithms through extensive evaluation using various real-world datasets.

Index Terms—Federated Learning, Privacy Preservation, Inference Attacks, Poisoning Attacks, Defense

I. Introduction

The quality and quantity of data is crucial for AI model performance, but handling user data, especially sensitive information, requires caution. Many countries have implemented laws to protect user privacy, such as the European Union's General Data Protection Regulation. While these laws safeguard user data, they also create challenges for researchers who need data to train AI models. Competition between different entities can lead to difficulties in sharing data, resulting in the creation of separate data repositories, known as data islands.

Federated learning, a solution to data protection challenges, preserves participants' privacy by transmitting model information instead of raw data. However, models generated through client training are vulnerable to attacks, potentially allowing adversaries to access specific training data. This vulnerability could be exploited to manipulate federated learning servers or participants.

This paper examines poisoning attacks and data privacy protection in federated learning. Privacy preservation and security concerns are closely linked and demand collaborative research. Chase et al. [1] proposed that privacy and security concerns in federated learning are amplified due to the use of shadow models for attribute inference. Thus, practical applications require a thorough examination of privacy and security issues.

Existing solutions to these issues mainly involve cryptographic encryption strategies and differential privacy techniques. Cryptographic encryption strategies include techniques like obfuscation circuits [2], oblivious transmission [3], secret sharing [4], and homomorphic encryption [5]. Differential privacy techniques involve adding noise to inputs [6], outputs [7], and objective functions [8] to protect user privacy. Although widely used, these techniques can affect model accuracy.

This paper presents a dynamic noise addition technique for neural network models, based on existing research on differential privacy. The method allows for the selection and adjustment of relevant parameters, such as the clipping threshold, and results in the generation of higher-quality machine learning models with fixed privacy budgets. Furthermore, the paper utilizes a Byzantine robustness aggregation strategy to improve the robustness of federated learning systems.

The contributions can be summarized as follows:

- We propose an adaptive differential privacy method to protect the privacy of federated learning participants' data and minimize the impact of differential privacy noise on the global model, thereby improving model precision.

- We propose a Byzantine robust aggregation strategy to defend against poisoning attacks and reduce the impact of differential privacy noise on the global model.
- We evaluate our methods using multiple real-world datasets. The results demonstrate the robust defense of our methods against privacy inference and poisoning attacks.

II. Problem Formulation

A. Federated Learning

A federated learning system comprises of a central server and multiple participants where local models or gradients are exchanged during training rather than raw data to protect privacy. Let $m \in \{1, 2, \cdots, M\}$ represent the participants, with M denoting the total number of participants. The local dataset of participant m is composed of a training dataset $\boldsymbol{D}_m$ and a test dataset $\widetilde{\boldsymbol{D}}_m$. The labels of sample $\boldsymbol{x}_{m,n}$ in $\boldsymbol{D}_m$ are denoted by $\boldsymbol{y}_{m,n}$, where $n \in 1, 2, \cdots, N_m$. Here, N_m signifies the overall number of samples within the dataset $\boldsymbol{D}_m$. The federated learning process involves five key steps:

- For each training round in federated learning, the central server employs a random selection process (or follows specific rules [9]) to determine the subset of participants involved.
- The participants who are selected utilize their local datasets to train their individual models. Subsequently, they encrypt their local models using techniques such as homomorphic encryption [10] before transmitting them to the central server.
- The central server then performs secure aggregation, leveraging the received information to update the global model.
- Following the model update, the central server broadcasts the updated global model to all participating individuals.
- Upon receiving the updated global model, each participant incorporates it to update their local model accordingly.

The aforementioned steps are iteratively repeated until the global model converges or reaches the maximum number of iterations, denoted by K. At the start of each training round, the central server randomly selects a subset of participants to engage in federated learning. Let $\boldsymbol{M}_k$ represent the set of participants selected at the k-th iteration, where $k \in \{1, 2, \cdots, K\}$. Subsequently, the selected participants, denoted by m, independently train their local model $\boldsymbol{\omega}_m^k$ using their respective local data. After training, the participants encrypt their trained models and transmit them to the central server.

Upon receiving the local models from all participants, the central server updates the global model by applying an aggregation algorithm. For instance, FedAVG [11] utilizes the following equations:

$$\Delta\boldsymbol{\omega}_m^k = \boldsymbol{\omega}_m^k - \boldsymbol{\omega}_0^k \tag{1}$$

$$\boldsymbol{\omega}_0^{k+1} = \boldsymbol{\omega}_0^k + \frac{\eta}{|\boldsymbol{M}_k|} \sum_{m \in \boldsymbol{M}_k} \Delta\boldsymbol{\omega}_m^k \tag{2}$$

where η denotes the learning rate, $|\boldsymbol{M}_k|$ represents the number of participants in the set $\boldsymbol{M}_k$, and $\boldsymbol{\omega}_0^k$ represents the global model at the k-th iteration.

B. Differential Privacy

Before delving into the definition of differential privacy, it is essential to establish the problem model of local differential privacy in the context of federated learning.

Definition 1 (Local Differential Privacy Problem Model). Let m denote a data owner who possesses a set of data $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$. The objective of data owner m is to provide a random algorithm $\mathcal{M}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$, that reveals specific information about the sample $\boldsymbol{x}$.

In federated learning, the term "data owner m" typically refers to each participant aiming to ensure that both the machine learning algorithm and the gradient $\mathcal{M}(\boldsymbol{x})$, obtained after the addition of noise, do not divulge any sensitive details from the training set.

Following the establishment of the local differential privacy problem model in federated learning, the subsequent step involves introducing the relevant definitions of differential privacy.

Definition 2 (ε-Differential Privacy). Let $\mathcal{M} : 2^{\mathcal{X}} \to \mathcal{Y}$ represent a random algorithm, where $\mathcal{Y}$ represents the domain of $\mathcal{M}$. Consider $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$, and let $\varepsilon > 0$ be a given value. An algorithm $\mathcal{M}$ satisfies ε-differential privacy if and only if, for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and any output subset $y \in \mathcal{Y}$, the following inequality holds:

$$Pr[\mathcal{M}(\boldsymbol{x}_1) \in y] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(\boldsymbol{x}_2) \in y] \tag{3}$$

where ε represents the privacy budget. Smaller values of ε correspond to higher levels of privacy protection but potentially lower data utility.

In practical scenarios, rigid adherence to ε-differential privacy may not be necessary. Instead, a common approach is to employ approximate differential privacy (ADP), which is defined as follows:

Definition 3 ((ε, δ)-Differential Privacy). Let $\mathcal{M} : 2^{\mathcal{X}} \to \mathcal{Y}$ be a random algorithm. Consider $\varepsilon, \delta > 0$. An algorithm $\mathcal{M}$ satisfies (ε, δ)-differential privacy if and only if, for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and any output subset $y \in \mathcal{Y}$, the following inequality holds:

$$Pr[\mathcal{M}(\boldsymbol{x}_1) \in y] \leq e^{\varepsilon} \cdot Pr[\mathcal{M}(\boldsymbol{x}_2) \in y] + \delta \tag{4}$$

where, δ is a non-negative parameter, typically set to a very small value, representing the probability of the algorithm not obeying differential privacy.

From the definitions above, it is evident that randomness plays a crucial role in achieving differential privacy.

Therefore, in general, differential privacy is accomplished by adding random noise. For continuous value domains, Laplace noise and Gaussian noise are commonly employed. In this paper, we primarily use the Gaussian noise.

To incorporate Gaussian noise, L_2 sensitivity is defined as follows:

Definition 4 (L_2 Sensitivity). The L_2 sensitivity of a function $f : 2^{\mathcal{X}} \to \mathbb{R}^k$ is defined as follows:

$$\Delta_2 f = \max_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}} ||f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)||_2 \tag{5}$$

Given a function f, we can achieve (ε, δ)-differential privacy by adding Gaussian noise as follows to obtain a random function $\mathcal{M}$:

$$\mathcal{M} = f(x) + \mathcal{N}(0, \sigma^2) \tag{6}$$

where σ^2 is determined using the sensitivity of f and the desired privacy parameters ε and δ as follows:

$$\sigma^2 = \frac{2\Delta_2 f^2 \ln(1.25/\delta)}{\varepsilon^2} \tag{7}$$

III. Privacy Protection Mechanisms for Federated Learning Based on Differential Privacy

Studies have shown that even in federated learning, privacy can be compromised when clients upload gradients or models. To mitigate this, this paper proposes adding noise to the uploaded gradients using differential privacy techniques, making it difficult for attackers to obtain precise gradient values and thus protecting user privacy.

The use of noise can negatively impact the performance of the global model, but this can be mitigated by using Byzantine robust aggregation algorithms. These algorithms are designed to handle situations where some clients may act maliciously or exhibit faulty behavior, ensuring the integrity of the aggregated model despite potential attacks or noisy gradients.

This paper introduced a privacy protection framework for federated learning, using differential privacy and Byzantine robust aggregation. As shown in Figure 1, the framework has two main components: a differential privacy and noise convergence mechanism on clients, and a Byzantine robust aggregation mechanism on servers.

The client-side component uses a differential privacy algorithm to adjust noise addition based on layer parameters. The locally trained model, with added noise, is evaluated on a local test set. If performance significantly drops, the update direction deviates too much from the original model, so additional noise is introduced to ensure higher privacy.

A noise mitigation algorithm, based on Byzantine robust aggregation, is introduced to counteract the impact of noise added by multiple clients on the global model. This algorithm treats models that significantly deviate from the expected update directions as potentially malicious, simulating poisoning attacks. By combining existing defense algorithms against poisoning attacks, the adverse effects of these models on the global model are mitigated.

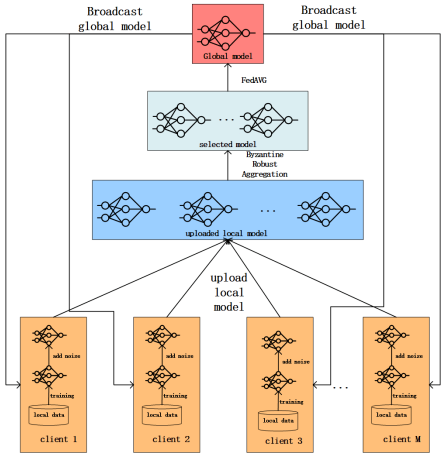


Fig. 1: The framework of ARPA-DP.

The proposed privacy protection method avoids significant increases in computational resource consumption and does not lead to wastage of communication resources due to the absence of an additional communication process.

A. Design of Privacy Protection Algorithms Based on Differential Privacy

Here, Algorithm 1 presents the APRA-DP algorithm, which is the core algorithm designed for privacy protection based on differential privacy.

The algorithm receives inputs such as the training set D_{train}, maximum number of iterations K, number of participants for each round of training n_k, local iteration κ, privacy budget parameters (ε, δ), and learning rate η. It outputs the local gradient $\boldsymbol{g}_m^t$.

In each iteration, n_k participants are selected from the participant set $\boldsymbol{M}_r$ using a Round-robin algorithm, and these participants are added to the set $\boldsymbol{M}_k$. For each participant $m \in \boldsymbol{M}_k$, the local model $\boldsymbol{w}_{m,0}^k$ is initialized as the global model $\boldsymbol{w}_0^k$. Each participant performs local training on its assigned samples in parallel.

During local training, for each sample i belonging to the participant m, the gradient $\boldsymbol{g}_i$ is computed based on the loss function $Loss(\boldsymbol{w}_m^t; \boldsymbol{x}_i)$. Next, noise is calculated according to Algorithm 2 and added to the gradient, yielding the noisy gradient $\tilde{\boldsymbol{g}}_i$. The noisy gradients for all samples are combined by summing them, resulting in the global gradient $\boldsymbol{g}_t$. The local model is then updated using the learning rate η and global gradient $\boldsymbol{g}_t$.After the algorithm completes K iterations, the final output is the global model $\boldsymbol{w}_0^K$.

Overall, the APRA-DP algorithm ensures privacy protection by incorporating differential privacy techniques without significant increases in computational resource consumption or wasteful use of communication resources.

In the context of differential privacy, the addition of noise is a critical factor. However, the introduction of

Algorithm 1 APRA-DP Algorithm

Input: training set$D_{train} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n\}$, max iteration K, participants number n_k, local iteration κ, privacy budgetε, δ,learning rate η
Output: local gradient$\boldsymbol{g}_m^t$

```
for k = 1, 2, ..., K do
    selecting n_k participants from M_r according to Round-robin algorithm
    add selected participants to M_k
    for each participant m ∈ M_k in parallel do
        update local model w_{m,0}^k = w_0^k
        for each sample i in m do
            calculating: g_i = Loss(w_m^t; x_i)
            calculating noise according to 2
            add noise: g̃_i ← g_i + N(0, σ_I^2 C^2 I)
        end for
        g_t ← Σ g̃_i
        update w_{m,i+1}^k = w_{m,i}^k − η g_t
    end for
end for
return global model w_0^K
```

noise may also lead to interference that could impact the normal training of the model. Therefore, striking the right balance between privacy protection and model utility is a crucial consideration when employing differential privacy technology. This section presents the APRA-DP (Adaptive Privacy Protection with Randomized Aggregation and Differential Privacy) algorithm, which is specifically designed to tackle this challenge.

The APRA-DP algorithm consists of two main steps:

1. Computing the Required Noise: The algorithm determines the appropriate amount of noise to be added based on factors such as the model gradient and privacy budget. It calculates the optimal level of noise to safeguard privacy without compromising the model's functionality.

2. Detecting Deviation from Update Direction: Following the addition of noise to the model gradient, the algorithm assesses whether the updated gradient significantly deviates from the original direction. It compares the performance of the model with the added noise to that of the original model on a validation set. If the performance drop exceeds a certain threshold, it indicates significant deviation in the model. In such cases, more noise is added to uphold privacy and usability.

By integrating these two steps, the APRA-DP algorithm aims to strike a balance between privacy protection and model availability. It adapts the noise level based on the model's sensitivity and evaluates its impact on performance. This approach ensures that privacy requirements are met without compromising the model's functionality.

To calculate the range of noise to be added to the gradient, we use the formula specified in Equation 7. Assuming that there are two adjacent datasets D_1 and D_2, and their corresponding updated models are $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, we can calculate the sensitivity s as follows:

$$\begin{aligned} s &= \max_{D_1, D_2 \subseteq \mathcal{X}} ||\boldsymbol{w}_1 - \boldsymbol{w}_2||_2 \\ &= \max_{D_1, D_2 \subseteq \mathcal{X}} ||(\boldsymbol{w}_0 - \eta \boldsymbol{g}_1) - (\boldsymbol{w}_0 - \eta \boldsymbol{g}_2)||_2 \\ &= \max_{D_1, D_2 \subseteq \mathcal{X}} ||\eta(\boldsymbol{g}_2 - \boldsymbol{g}_1)||_2 \end{aligned} \tag{8}$$

Algorithm 2 Dynamic Noise Adjustment

Gradient:$\boldsymbol{g}$, privacy budget:ε, noise term: σ maximum re-addition:rd model:$\boldsymbol{w}$ test set:D learning rate:η threshold:lt
Output: noise:$\boldsymbol{g}_{noise}$

```
Calculate the accuracy of test model pre_acc = eval(w, D)
Add noise to model:g_1 ← g + g_noise
Calculate the accuracy of the model:pos_acc = eval(w − η g_1, D)
Save the noise:g_best ← g_noise
if pre_acc > pos_acc and (pre_acc − pos_acc)/(pre_acc) > lt then
    for r = 1, 2, ..., rd do
        Add noise to the gradient g_1 ← g + g_noise^r
        Calculate the accuracy of the model:pos_acc = eval(w − η g_1, D)
        if pos_acc > pre_acc then
            Update the noise:g_best ← g_noise^r
        end if
    end for
end if
return noise:g_best
```

The dynamic noise adjustment algorithm, described in Algorithm 2, aims to determine the appropriate amount of noise to be added to the gradient.

The algorithm 2 initially evaluates the accuracy (pre_acc) of the model $\boldsymbol{w}$ on the test set D. It then adds noise to the gradients by setting $\boldsymbol{g}_1 = \boldsymbol{g} + \boldsymbol{g}_{noise}$, where $\boldsymbol{g}_{noise}$ represents the noise. The accuracy of the model after adding noise (pos_acc) is then evaluated.

If the accuracy before adding noise is higher than the accuracy after adding noise, and the performance drop exceeds a predefined threshold (lt), the algorithm proceeds to re-add noise within the specified limit (rd) and tests the model's accuracy each time. The noise that results in the highest accuracy improvement ($\boldsymbol{g}_{best}$) is retained.

Finally, the algorithm outputs the noise term $\boldsymbol{g}_{best}$ that yields the highest accuracy improvement. This noise term is then added to the gradient to achieve the desired privacy protection while maintaining model performance.

The algorithm adjusts the amount of noise added to the gradient, balancing privacy protection and model accuracy.

After introducing perturbations, the model may exhibit a deviation from the original direction of gradient update.

In order to mitigate its impact on the global model, the addition of noise can be reconsidered. The specific approach for incorporating noise into the gradient is outlined in Algorithm 2. Prior to noise injection, the accuracy of the updated model $\boldsymbol{w}_m^{t+1}$ is evaluated using the client's local test set, denoted as a_1. Subsequently, the accuracy of the perturbed model $\tilde{\boldsymbol{w}}_m^{t+1}$ is assessed by further testing it using the same test set, and this accuracy value is recorded as a_2. If a_2 is less than a_1 and the relative decrease $\frac{(a_1-a_2)}{a_1}$ is below a predefined threshold lt, it indicates a substantial deviation between the direction of gradient update after incorporating noise and the original direction. In such circumstances, the original gradient will be re-introduced, and the aforementioned steps will be repeated. To prevent unnecessary resource consumption, a maximum loop count rd is set, and ultimately, the gradient exhibiting minimal deviation within the loop count or the gradient with the least deviation is returned.

It should be noted that the methodology proposed in this paper, employing a test set to detect deviations in the perturbed model, is not the sole approach. Various similar techniques, such as cosine similarity calculation, Pearson correlation coefficient analysis, and Support Vector Machines (SVM), can also be utilized to determine whether $\tilde{\boldsymbol{w}}_m^{t+1}$ deviates from the normal direction of update. Furthermore, the aforementioned method of re-adding noise does not guarantee that all gradients provided by clients upon upload conform to the normal update direction. Hence, reinforcing server-side processing for models exhibiting significant deviations is necessary to enhance the robustness of the system.

B. Design of Noise Mitigation Algorithms Based on Byzantine Robust Aggregation

Researchers in federated learning are exploring robust aggregation methods to counter poisoning attacks, especially when using differential privacy. Adding noise to local models can create inconsistency between the original and noised models, mimicking malicious behavior in poisoning attacks. This guides the identification of significantly deviating noised models as malicious.

In light of the above, Algorithm 3 illustrates the process involved in handling these deviant models. In this approach, upon receiving updates from all participants, the server systematically examines the impact of removing each model on the overall performance of the global model. Subsequently, the client model that causes the most substantial decline in global model performance is eliminated from the update process.

Algorithm 3 Byzantine Robust Aggregation Method

Input: local model set $\boldsymbol{W}_{set} = \{\boldsymbol{w}_{m_1}^t, \boldsymbol{w}_{m_2}^t, \cdots, \boldsymbol{w}_{m_{n_k}}^t\}$, participants number n_k, max remove number rm
Output: global model $\boldsymbol{w}_0^{t+1}$

```
for n ∈ {1, 2, ··· , rm} do
    w_avg ← FedAVG(W_set)
    acc_0 ← Eval(w_avg, D_test)
    I = −1
    for i ∈ {1, 2, ··· , n_k} do
        w̃_avg ← FedAVG(W_set − w_i^t)
        acc_1 ← Eval(w̃_avg, D_test)
        if acc_1 > acc_0 then
            acc_0 ← acc_1
            I = i
        end if
    end for
    if I > 0 then
        W_set ← W_set − w_I^t
    end if
end for
w_0^{t+1} ← FedAVG(W_set)
return global model w_0^{t+1}
```

IV. Experimental Evaluation

In this paper, a lightweight federated learning method is proposed after using the differential privacy algorithm with improved dynamic privacy budget. The method has a differential privacy protection algorithm that does not affect training, where each client only trains locally and submits the protected parameter data to the central server. Compared with traditional methods, this algorithm provides some improvements in computational complexity, and since the SGD algorithm executed locally by users is a general step in federated learning, it is not considered in this algorithm.

A. Experiment Data Sets

This paper uses FASHION MNIST [12] dataset, MNIST [13] dataset and FeMNIST [14] dataset for testing.

In this paper, the training samples from the FASHION MNIST and MNIST datasets are equally distributed among all participants, and each participant receives the same number of samples. All participants divide the data set into training sets, validation sets, and test sets according to an 8:1:1 ratio.

B. Experimental Setup

In this paper, we use Specifically, for the MNIST, FASHION MNIST and FeMNIST datasets, the same network structure is used. In addition, Pytorch deep learning framework is used to construct the neural network model, and the optimizer used is SGD.

The experiment was conducted on a server with an Intel i7-13700k CPU and a Nvidia GTX 3080 GPU. The parameters used in this paper are shown in Table I. The privacy budget written in the table represents the total privacy budget used during the training process of a single client, which means that the privacy budget for each iteration is ε/t_m.

TABLE I: Experiments setting

parameters	description	value
M	number of participants	100
η	learning rate	0.01
ε	privacy budget	$[1, 100]$
K	max iteration	200
n_k	participants number of each round	10
δ	privacy term	10^{-5}
rm	max remove number	3
rd	max re-add number	3

C. Baseline

This article conducts a comparative analysis of our proposed method against the following approaches:

1. FedAVG [11]: This method represents federated learning without any defense strategy.

2. DP [15]: This method uses local differential privacy (LDP) to protect the privacy of participants in federated learning.

3. Sparsification [16]: This method reduces information available to potential attackers and improves communication efficiency through gradient compression.

D. Attack Method

For privacy inference attacks, DLG [17] is employed to introduce gradient perturbations to participants in federated learning. These perturbations are used to reconstruct the original training images for assessing the effectiveness of attacks. The evaluation metrics measuring effectiveness include the structural similarity index measure (SSIM) [18] and the peak-signal-to-noise ratio (PSNR) [18], which compare the reconstructed images with the original ones.

In federated learning, two strategies are used to manipulate the global model in poisoning attacks:

1.Label flipping attacks involve an attacker changing the labels of samples from a source class to a target class to deceive a compromised global model into misclassifying them as the target class. This does not impact other tasks.

2.Backdoor attacks involve modifying the top-left four pixels of normal images to specific values, with the goal of tricking a global model into classifying them as a specific target class, thereby introducing a backdoor effect.

E. Experimental Results

We examined the impact of noise on privacy protection by comparing the restored images using the DLG algorithm. We added varying amounts of noise using different strategies (DP and APRA-DP) in the MNIST dataset and compared the similarity between the restored images.

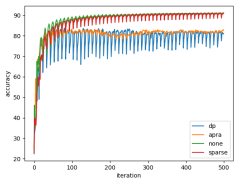
TABLE II: Recover Effectiveness

	DP		APRA-DP	
	ssim	psnr	ssim	psnr
epsilon=10	-0.000526	5.161493	-0.000298	5.016296
epsilon=100	0.006431	5.006114	0.002794	5.006899
epsilon=1000	0.041078	4.998769	0.028730	4.661476
none	0.224729	4.440084	0.248489	6.872755

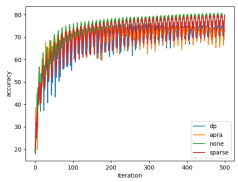
According to Table II, using a dynamic noise addition strategy does not significantly contribute to restoring images to attackers. Therefore, the proposed method in this paper achieves the same level of privacy protection as adding the same differential privacy noise.

We also assessed the effectiveness of baseline methods against poisoning attacks. The variation in model training accuracy in response to the number of iterations for label flipping attacks with baseline methods is depicted in Figure 2. The graph indicates that the model trained using the ARPA-DP method exhibits higher accuracy compared to the one trained using the DP method.

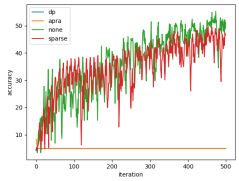
We further analyzed the impact of adding different differential privacy noises to the global model accuracy after training. The paper utilized a rotating method to select clients, with the batch size for each client's training set equal to its local training set size, and each client iterating once per round of training to facilitate the calculation of privacy budget and the reduction coefficient.



(a) MNIST(non-IID)



(b) FASHION MNIST(non-IID)



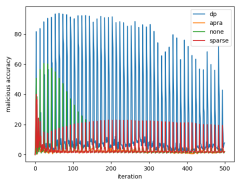
(c) FeMNIST(non-IID)

Fig. 2: Global model accuracy under label flipping attacks. Baseline methods: FedAVG (non-IID), DP (non-IID), Sparsification (non-IID).

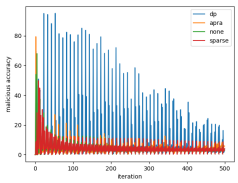
Figure 3 illustrates the accuracy of the global model during the training process on the malicious dataset. It is evident from the graph that the ARPA-DP method demonstrates significantly better defense capabilities against poisoning attacks compared to other methods, and it also exhibits greater stability.

We analyzed the proposed APRA-DP algorithm and tested its efficacy on the MNIST, FASHION MNIST, and FeMNIST datasets. The used comparison method was a differential privacy algorithm. [19]

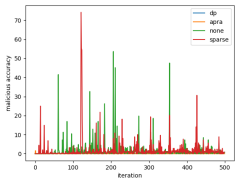
Regarding the model accuracy changes in MNIST, FeMNIST and FASHION datasets, there are many similarities between them. Figure 2 shows that when no noise is added or the amount of noise is very small (in this case, $\varepsilon = 100$), the model accuracy changes under different scenarios,



(a) MNIST(non-IID)



(b) FASHION MNIST(non-IID)



(c) FeMNIST(non-IID)

Fig. 3: Malicious Accuracy in MNIST, FASHION MNIST and FeMNIST. Baseline methods: FedAVG (non-IID), DP (non-IID), Sparsification (non-IID).

including using no additional measures and the APRA-DP strategy proposed in this paper, are largely the same. This indicates that when the noise amount is not large, APRA-DP does not have a negative impact on model accuracy. On the other hand, when too much noise is added, APRA-DP can effectively reduce the impact of noise on the model. Especially when $\varepsilon = 1$, the model cannot converge when no measures are taken, but APRA-DP reduces deviation between local gradient updates, preventing the model from converging too slowly.

V. Conclusion

This article introduces an advanced strategy for implementing dynamic differential privacy in neural network-based federated learning models. The approach proposes a new federated learning model called ARPA-DP, which integrates differential privacy to protect the privacy of clients participating in the federated learning process. Additionally, this methodology encompasses techniques for detecting and defending against malicious models, drawing from research on defending against poison attacks, to mitigate the impact of noise generated by differential privacy on the overall performance. Experimental evaluations carried out on various real-world datasets confirm the effectiveness of the proposed approach in ensuring robust privacy protection and strengthening federated learning models against poison attacks.

References

[1] M. Chase, E. Ghosh, and S. Mahloujifar, "Property inference from poisoning," 2021.

[2] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng, "Privacy-preserving federated deep learning with irregular users," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 2, pp. 1364–1381, 2022.

[3] D. Byrd, V. Mugunthan, A. Polychroniadou, and T. Balch, "Collusion resistant federated learning with oblivious distributed differential privacy," in Proceedings of the Third ACM International Conference on AI in Finance, 2022, pp. 114–122.

[4] H. Zhu, R. S. Mong Goh, and W.-K. Ng, "Privacy-preserving weighted federated learning within the secret sharing framework," IEEE Access, vol. 8, pp. 198 275–198 284, 2020.

[5] A. Guleria, J. Harshan, R. Prasad, and B. N. Bharath, "On homomorphic encryption based strategies for class imbalance in federated learning," CODS COMAD 2024, 2024, accepted for Presentation at ACM Conference on Data Sciences (CODS) and Conference on Machine Learning and Data Mining (COMAD).

[6] F. Fioretto, P. Van Hentenryck, and K. Zhu, "Differential privacy of hierarchical census data: An optimization approach," Artificial Intelligence, vol. 296, p. 103475, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370221000266

[7] Y. Xiao, L. Xiong, and C. Yuan, "Differentially private data release through multidimensional partitioning," in Secure Data Management, W. Jonker and M. Petković, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 150–168.

[8] K. Pan, M. Gong, K. Feng, and K. Wang, "Differentially private regression analysis with dynamic privacy allocation," Knowledge-Based Systems, vol. 217, p. 106795, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121000587

[9] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, "Tifl: A tier-based federated learning system," in Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, 2020, pp. 125–136.

[10] Z. Ma, J. Ma, Y. Miao, X. Liu, K.-K. R. Choo, and R. Deng, "Pocket diagnosis: Secure federated learning against poisoning attack in the cloud," IEEE Transactions on Services Computing, 2021.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Artificial intelligence and statistics. PMLR, 2017, pp. 1273–1282.

[12] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[13] L. Deng, "The mnist database of handwritten digit images for machine learning research," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141–142, 2012.

[14] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," arXiv preprint arXiv:1812.01097, 2018.

[15] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated anomaly analytics for local model poisoning attack," IEEE Journal on Selected Areas in Communications, vol. 40, no. 2, pp. 596–610, 2022.

[16] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," ArXiv, vol. abs/1712.01887, 2017.

[17] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/60a6c4002cc7b29142def8871531281a-Paper.pdf

[18] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in 2010 20th International Conference on Pattern Recognition, 2010, pp. 2366–2369.

[19] W. Yang, Y. Zhou, M. Hu, D. Wu, X. Zheng, J. H. Wang, S. Guo, and C. Li, "Gain without pain: Offsetting dp-injected noises stealthily in cross-device federated learning," IEEE Internet of Things Journal, vol. 9, no. 22, pp. 22 147–22 157, 2022.