Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.061377

ARTICLE

Defending Federated Learning System from Poisoning Attacks via Efficient Unlearning

Long Cai, Ke Gu* and Jiaqi Lei

School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha, 410114, China

*Corresponding Author: Ke Gu. Email: gk4572@163.com

Received: 23 November 2024; Accepted: 14 February 2025; Published: 26 March 2025

ABSTRACT: Large-scale neural networks-based federated learning (FL) has gained public recognition for its effective capabilities in distributed training. Nonetheless, the open system architecture inherent to federated learning systems raises concerns regarding their vulnerability to potential attacks. Poisoning attacks turn into a major menace to federated learning on account of their concealed property and potent destructive force. By altering the local model during routine machine learning training, attackers can easily contaminate the global model. Traditional detection and aggregation solutions mitigate certain threats, but they are still insufficient to completely eliminate the influence generated by attackers. Therefore, federated unlearning that can remove unreliable models while maintaining the accuracy of the global model has become a solution. Unfortunately some existing federated unlearning approaches are rather difficult to be applied in large neural network models because of their high computational expenses. Hence, we propose SlideFU, an efficient anti-poisoning attack federated unlearning framework. The primary concept of SlideFU is to employ sliding window to construct the training process, where all operations are confined within the window. We design a malicious detection scheme based on principal component analysis (PCA), which calculates the trust factors between compressed models in a low-cost way to eliminate unreliable models. After confirming that the global model is under attack, the system activates the federated unlearning process, calibrates the gradients based on the updated direction of the calibration gradients. Experiments on two public datasets demonstrate that our scheme can recover a robust model with extremely high efficiency.

KEYWORDS: Federated learning; malicious client detection; model recovery; machine unlearning

1 Introduction

1.1 Background

Machine learning is currently the most popular technology, advancing rapidly and being widely implemented in domains such as natural language processing, smart grid, smart transportation, etc. However, owing to the privacy and security issues brought by traditional centralized machine learning, some institutions are reluctant to share their sensitive data for collaborative training. Federated learning, as a distributed machine learning paradigm compatible with edge computing, addresses these issues [1]. It randomly initializes a training model and broadcasts the model to the participants, then requests them to upload the trained models and aggregate those models. This not only guarantees the privacy of the participants, but also alleviates the computational burden on the central server [2]. Although federated learning has a bright future in distributed learning, there are more intricate and hidden security risks associated with it. Poisoning attacks pose a significant threat to federated learning system due to their strong destructive and covert performance [3–5]. A particularly alarming feature is that attackers can masquerade as legitimate participants, engaging in the global model training process as usual, while submitting tainted models that compromise the reliability of the global model.

Concealment represents a critical characteristic of poisoning attacks. In the context of federated learning, the aggregation server is restricted from directly accessing the private datasets employed for client training, which hinders its ability to identify anomalous training samples. In certain scenarios, attackers may engage in training as legitimate clients, which undermines the effectiveness of conventional traffic monitoring methods. As a result, current works focused on mitigating poisoning attacks primarily emphasize the analysis of client behavioral traits and the implementation of Byzantine robust aggregation [6–10]. The former employs cosine similarity or density-based clustering algorithms to identify outliers, while the latter designs new robust aggregation rules to mitigate the influence of malicious gradients. Nevertheless, these works still fail to address a crucial issue, that is, to recover the global model from poisoning attacks. When a federated learning model is subjected to poisoning attacks, the influence of abnormal data can progressively undermine the model's functionality. Furthermore, attackers might construct backdoors to take control of the global model. Concurrently, the compromised global model may adversely affect the training performance of other clients in subsequent epochs.

Federated unlearning (FU), a recently introduced concept, has surfaced as a viable solution to this challenge [11–13]. In instances where specific clients desire to withdraw from federated learning training, federated unlearning provides a mechanism to mitigate their influence, thereby safeguarding their privacy. When confronted with the menace of poisoning attacks, federated unlearning can play a crucial role in mitigating the influence of such adversarial actions. Regrettably, some existing federated unlearning approaches demand a considerable quantity of computational resources [14], which is unacceptable for federated learning systems that involve numerous resource-constrained devices. Concurrently, the global model recovery process is frail, and a lengthy recovery period offers new opportunities for attackers and might even make the recovered model even worse. Hence, it is necessary to have an efficient and accurate global model recovery strategy for the scenarios of federated learning under poisoning attacks.

1.2 Contribution

In this paper, we propose a framework named SlideFU for defending large federated learning networks from poisoning attacks. SlideFU limits the influence of poisoning attacks within a sliding window. If a notable decline in model accuracy is observed during the training process, the aggregation server calculates the trust factor of each client through PCA compressed local gradients, thereby excluding the attackers. Subsequently, the aggregation server uses the sliding window to roll back the global model and calibrates the local gradients of remaining clients within the window. Through iterative calibration, we obtain a recovered global model that completely eliminates malicious influence. Overall, our contributions are as follows:

- We propose an efficient method for detecting poisoning attackers in large-scale federated learning networks. It utilizes lighter compressed gradient parameters to analyze the trust factors on the client side.
- We propose a federated unlearning framework using sliding window, which limits the global model training and unlearning process to the size of the sliding window. It alleviates the cost and security issues caused by traditional long-term unlearning process.
- We evaluate the accuracy and efficiency of our proposed scheme through simulation experiments in different attack scenarios. By comparing it with several other unlearning methods, we demonstrate that our scheme has significant advantages in unlearning costs while ensuring the accuracy of the global model.

1.3 Organization

The rest structures of this paper are as follows. In Section 2, we introduce recent defense strategies against federated learning poisoning attacks. The preliminary knowledge of this paper is shown in Section 3. In Section 4, we provide the specific design and details of SlideFU, an efficient anti-poisoning attack federated unlearning framework. In Section 5, we conduct performance experiments in different attack scenarios to demonstrate the effectiveness of our scheme. Then we draw our conclusion in Section 6.

2 Related Works

In this section, we introduce the current various defense strategies against poisoning attacks.

2.1 Byzantine Tolerance Schemes against Poisoning Attacks

Poisoning attackers seek to disrupt the global model through the construction of malicious data samples or model parameters [15], with the main affected being the local gradients uploaded by the attackers. Consequently, the majority of contemporary anti-poisoning attack strategies primarily concentrate on the identification of outlier local gradients. Yin et al. [7] changed the aggregation rule and calculated the median of all gradients as the final gradient. However, it is difficult to withstand poisoning attacks targeting the median dimension. Blanchard [3] selected the model with smallest difference from other models among several local models and Multi-Krum further selected more of these models to obtain their mean as the global model. In addition, Cao et al. [16] stored a clean dataset on the server, evaluated each local gradient through cosine similarity, redistributed trust scores and perform aggregation. However, storing a clean dataset on the server is not realistic because of the limit of federated learning. Sattler et al. [17] divided edge nodes into different clusters based on the cosine similarity between the gradients they uploaded, in order to discover the outlier nodes. In [18], they implemented adaptive anomaly model detection by using data validation methods. Erdol et al. [19] proposed a defense strategy that identify harmful attackers by size reduction algorithms. To address the challenge of traditional differential privacy techniques providing cover for poisoned parameters, Huang et al. [20] proposed a verifiable privacy preserving federated learning scheme to protect client privacy. These methods may protect the global model to some extent, but they still face a challenge. That is, the gradient effect of successful poisoning attacks still exists in the global model.

2.2 Machine Unlearning

The emergence of machine unlearning breaks this deadlock. Some studies aim to eliminate the influence of partial samples in machine learning models. Train from scratch [21] is a fundamental method of machine unlearning which involves completely retraining after eliminating targeted samples. This approach is straightforward yet costly, and it is hard for federated learning to bear its expense. So Cao et al. [22] presented an unlearning approach by transforming learning algorithms into a summation form, then made system forget the targeted samples. Ginart et al. [23] proposed two efficient data deletion algorithms based on k-means clustering. However, these methods are not effective in the face of federated learning poisoning attacks. So some scholars extended it to the context of federated learning, Gong et al. [13] studied federated unlearning within a Bayesian framework and proposed an efficient unlearning mechanism. Wu et al. [11] proposed a framework which is empowered by reverse stochastic gradient ascent and elastic weight consolidation. Liu et al. [24] proposed an unlearning calibration method based on historical information. As for the elimination of the influence of poisoning attacks, Cao et al. [25] proposed FedRecover, a method that combines exact retraining and gradients estimation. It ensures to a certain extent the accuracy of the recovery of federated learning model under poisoning attacks and enhances the efficiency of unlearning. However its limitation is that it is difficult to support large neural network models. Zhang et al. [26] erased the influence of clients by removing a weighted sum of gradient residuals from model and ensured the privacy of clients. Yuan et al. [27] proposed an efficient and robust federated unlearning framework, which is highly robust in resisting dynamic attacks. Although the unlearning schemes mentioned above have to some extent eliminated the influence of attackers, they all suffer from reduced global model accuracy or high costs due to multiple inaccurate calculations.

3 Preliminary Knowledge

3.1 Federated Learning

As shown in Fig. 1, federated learning is an advanced paradigm that supports multiple network nodes to participate in machine learning training. It only contains two entities: aggregation server and client. Aggregation server possesses robust computational and storage capabilities, enabling it to collect and aggregate gradients. In our design, the aggregation server is responsible for both detecting malicious attackers and unlearning the global model. Conversely, the clients possess limited computational capabilities, allowing them to train on local data and subsequently upload the gradients. When the federated learning process begins, the aggregation server first initialize a randomly generated global model θ^0. Then in each epoch t, the aggregation server broadcasts θ^0 to all the N clients to locally train the global model θ^t with their private data. Each selected client i uses stochastic gradient descent (SGD) to obtain local gradient g_i^t, which is then sent to the aggregation server. The calculation of local gradient can be formulated as:

$$g_i^t = \frac{\partial \mathscr{L}(\theta^{t-1}, D_i)}{\partial \theta^{t-1}} \tag{1}$$

where η is the learning rate which is pre-set, $\mathscr{L}(\cdot)$ is the loss function and D_i is the private data of a client. After receiving sufficient local gradients, the aggregation sever aggregates the gradients into a new epoch of global model based on a aggregation rule.

$$\theta^t = \theta^{t-1} - \eta \cdot \mathscr{S}(g_1^t, \ldots, g_i^t, \ldots, g_n^t) \tag{2}$$

where $\mathscr{S}(\cdot)$ is the aggregation rule. In traditional federated learning, the aggregation rule is FedAvg, which aggregates gradients based on the number of samples as weights. Our scheme also complies with this aggregation rule.

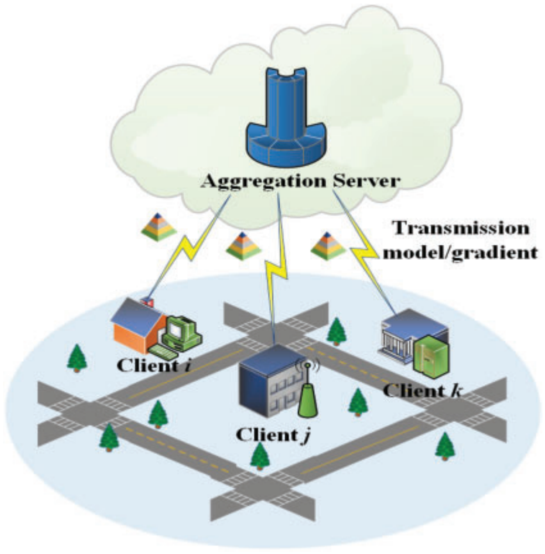

Figure 1: Framework of federated learning

3.2 Federated Unlearning

Federated unlearning is a variant of machine unlearning that extends machine unlearning paradigm to distributed learning. It requires service providers to remove sensitive data from machine learning models while eliminating their features and influence. The goal of federated unlearning is to forget the targeted client's data D_f, that is, to eliminate the influence of D_f from model θ. Formally, give a global model θ trained on datasets $D = \{D_1, \ldots, D_N\}$ with learning algorithm $\mathscr{A}$, federated unlearning algorithm $\mathscr{U}(\mathscr{A}(D), D, D_f)$ is to remove the influence of D_f while ensuring the performance of model θ^- at D/D_f. Compared to machine unlearning, the difficulty of this work lies in the fact that the aggregation server is prohibited from obtaining training data and the FL model extracts features from various data points, deleting data points recklessly may break the dependency relationships, resulting in a significant decrease in model performance (catastrophic forgetting). Train from scratch [21] is a simple solution that only requires recovering the model network structure to its initial state and requiring the remaining clients to retrain. Existing works FedEraser [24] and FedRecover [25] optimize the cost of unlearning based on this, but they still require a significant amount of cost. Nonetheless, an excessively prolonged unlearning process is deemed unacceptable for the majority of resource-constrained federated learning devices.

3.3 Poisoning Attack

Poisoning attacks damage the integrity of artificial intelligence models. According to the attacker's task objectives, poisoning attacks can generally be divided into two categories:

- **untargeted attacks:** In the context of federated learning, the primary objective of this kind of attacks is to diminish the accuracy of the global model. Attackers can offset gradients by incorporating poisoned samples to the local dataset or manipulating the fine-tuning process. Without security measures, the global model may be compromised due to chaotic gradient aggregation.
- **targeted attacks:** Compared to the untargeted attacks, it is more covert and less destructive. The primary objective of targeted attacks is to induce incorrect predictions of sample labels by the global model when it encounters specific samples. Attackers can execute targeted attacks by altering the labels of targeted samples, or implanting backdoors in specific samples to trigger them during the training process. The trained gradients can mislead the global model regarding certain samples, and the implanted backdoors may persistently affect the model's prediction.

For the convenience of understanding, we describe the main symbols used in this paper in Table 1.

Table 1: Symbols and descriptions

Symbols	Descriptions
D	All data involved in training
D_f	All data need to be unlearned involved in training
$\tilde{g}$	Last layer parameters of client local gradient
g'	Compressed client local gradient
$\bar{g}$	Original client local gradient
$\hat{g}$	Unlearned client local gradient
$\tilde{\theta}$	Last layer parameters of global model
θ'	Compressed global model
$\bar{\theta}$	Original global model
$\hat{\theta}$	Unlearned global model

(Continued)

Table 1 (continued)

Symbols	Descriptions
ρ	Trust factors of client
γ	Adjustment factor
η	Learning rate
T	The total epochs of federated learning
T_k	The size of sliding window

4 Efficient Anti-Poisoning Attack Federated Unlearning Framework

In this section, we propose a federated learning anti-poisoning attack framework in large neural networks. Firstly, the conventional federated learning process is incompatible with unlearning, as aggregation server always need to store training gradient information for multiple epochs. Among this information, only a small amount of recent information has significant impacts on the unlearning performance. Therefore, we design a sliding window mechanism which restricts all unlearning operations to a defined temporal scope, thereby alleviating the storage and computational burdens on the aggregation server. Secondly, as shown in Fig. 2, SlideFU executes the federated learning process in two sequential steps. The first one is optimized malicious detection in large-scale neural networks, where each gradient has millions of parameters. We select specific layer parameters and use PCA method to compress them, thereby eliminating potential attackers in the system by calculating malicious factors of clients based on compressed parameters. The second step is the efficient federated unlearning which utilizes the retrained gradients of remaining clients to calibrate the update direction. This process effectively facilitates the elimination of malicious gradient features from the global model.

4.1 Initialization

In this stage, the federated learning system completes the initialization process. The aggregation server first randomly generates a global model θ^0 and broadcasts it to all clients, while generating a sliding window of size T_k. In the first few epochs of federated learning ($t < T_k$), traditional federated learning process is executed, and all the clients upload their trained local gradients g_i^{t+1} for aggregation as follows:

$$\theta^{t+1} = \theta^t - \eta \cdot \sum_{i=1}^{N} \frac{\|D_i\|}{\|D\|} \cdot g_i^{t+1} \tag{3}$$

Note that although there are some Byzantine-resilient aggregation functions, FedAvg can make the model change more significantly after being attacked, making it easier for us to detect and eliminate malicious clients. The start time and frequency of poisoning attacks in real-world scenarios are difficult to determine, but dynamic sliding windows can be applied to any situation. Even in the early stages of federated learning, the identification of malicious clients presents a significant challenge owing to the variability in local data present on the client side, but sliding windows can use several epochs of training to eliminate malicious clients more accurately. Every time an epoch of federated learning training is completed, the pointer within the sliding window moves forward by one unit. When the pointer reaches the top of the sliding window ($t = T_k$), malicious detection and efficient unlearning are initiated.

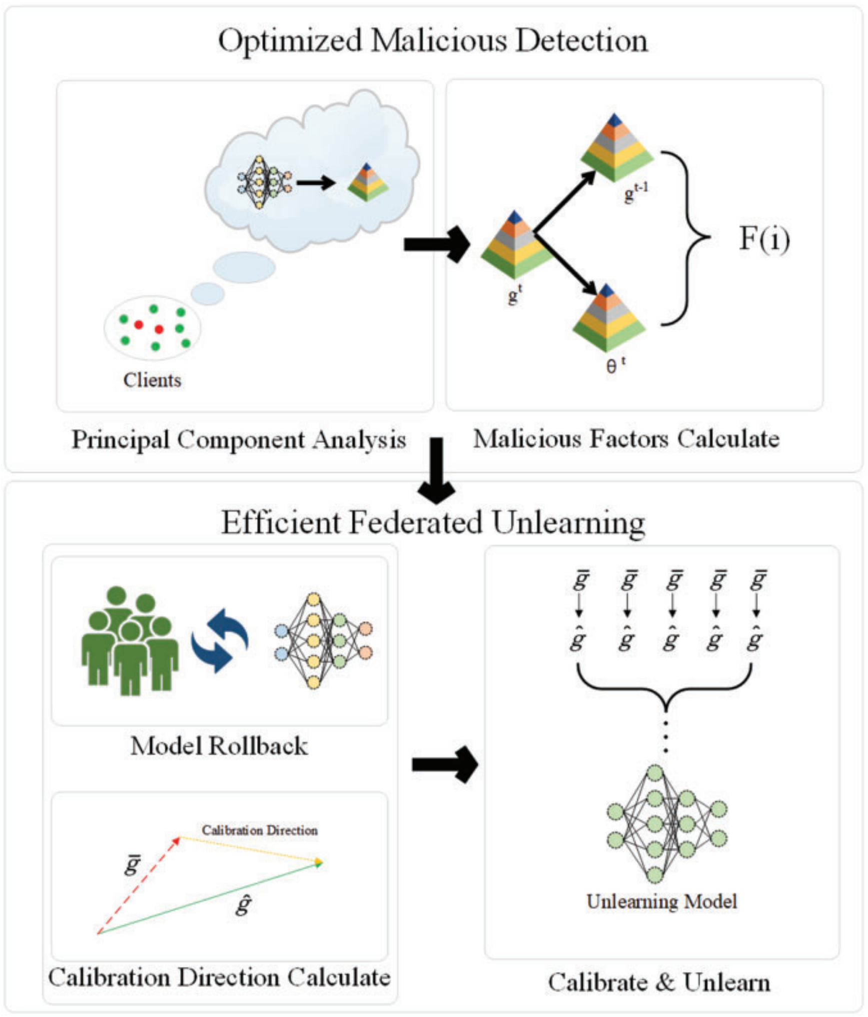
Optimized Malicious Detection
Clients
Principal Component Analysis
g^{t-1}
g^t
θ^t
F(i)
Malicious Factors Calculate
Efficient Federated Unlearning
Model Rollback
Calibration Direction
$\bar{g}$
$\hat{g}$
Calibration Direction Calculate
Unlearning Model
Calibrate & Unlearn

Figure 2: Workflow of SlideFU

4.2 Optimized Malicious Detection Mechanism in Large Neural Networks

In this section, we propose an optimized malicious detection mechanism in large neural networks. The essential aspect of detecting malicious activities in poisoning attacks involves identifying anomalous alterations in gradients and the model. In federated learning based on large-scale neural networks, redundant parameters complicate the assessment of gradient variations. Recent work [28] found that the neurons in the last layer of the deep learning model exhibit valuable features for malicious detection. Regardless of the model used, the weights and biases of neurons in other levels directly and highly influence those of the last layer of neurons through propagation, which means the parameters of the last layer of neurons can reflect the differences in the model to a certain extent. Therefore, in our malicious detection mechanism, only the last layer parameters of gradients $\tilde{g}_i$ and model $\tilde{\theta}$ are extracted as detection targets. In addition, to further save computation costs, we compressed these parameters using principal component analysis (PCA) method. PCA discovers a new group of orthogonal variables (namely, principal components) through linearly combining the original variables to represent the main characteristics of the data in lower dimensions. Each principal component is a linear combination of the original variables and is orthogonal to one another, guaranteeing that the information is not redundant. The calculations of compressed parameters are as follows:

$$g' = \tilde{g} \times V \tag{4}$$

$$\theta' = \tilde{\theta} \times V \tag{5}$$

where V is the projection matrix of corresponding parameter, $\tilde{g}$ and $\tilde{\theta}$ are respectively the last layer parameters of local gradient and global model. Multiplying the high-dimensional gradient or model matrix with the projection matrix can obtain the required low-dimensional information matrix. To ensure that the dimensionality reduction matrix can reflect the information of the original matrix to a certain extent, the calculation of projection matrix is particularly important. Firstly we calculate the mean vector of the original matrix, subtract the corresponding mean from each feature. Then we calculate the covariance matrix, perform eigenvalues decomposition on the covariance matrix to obtain the corresponding eigenvalues λ and eigenvectors v. In the end, we select the eigenvectors corresponding to the first k eigenvalues as principal components to form a projection matrix V (each column is a eigenvector) [29].

In any subsequent federated learning epoch t ($t > T_k$), when there is a significant decrease in the accuracy of global model, the training is paused and the malicious detection mechanism is activated. The aggregation server retrieves the compressed parameters of the current epoch gradients g'^{t}, the previous epoch gradients g'^{t-1} and the global model θ'^{t} stored in the sliding window. Then the aggregation server calculates the trust factor ρ_i for each client as follows:

$$\rho_i = \gamma \cdot \frac{g_i'^{t} \cdot \theta'^{t}}{\|g_i'^{t}\| \cdot \|\theta'^{t}\|} + (1-\gamma) \cdot \frac{g_i'^{t} \cdot g_i'^{t-1}}{\|g_i'^{t}\| \cdot \|g_i'^{t-1}\|} \tag{6}$$

where γ is the adjustment factor, the trust factor of each client is obtained by weighted addition of two cosine similarities, which alleviates the detection inaccuracy issue caused by compressed parameters. The first polynomial in the equation $\frac{g_i'^{t} \cdot \theta'^{t}}{\|g_i'^{t}\| \cdot \|\theta'^{t}\|}$ is called outlier detection, which detects the similarity between the targeted gradient and the aggregated global model. Due to the aggregation rule being FedAvg, the global model can to some extent reflect the centroid of most gradients. The latter polynomial in the equation $\frac{g_i'^{t} \cdot g_i'^{t-1}}{\|g_i'^{t}\| \cdot \|g_i'^{t-1}\|}$ is called misuse detection, which detects the similarity between the current and previous gradients of the targeted client. To some extent it can expose some attack characteristics of attackers, such as high randomness of untargeted attacks leads to low similarity between two epoch gradients. After that, the aggregation server arranges the trust factors of clients in descending order and removes a certain percentage of potential attackers with corresponding lowest trust factor. The whole optimized malicious detection mechanism in large neural networks is described in Algorithm 1.

Algorithm 1: Optimized malicious detection mechanism

```
Procedure:
1: Aggregation server initializes the global model $\theta^0$.
2: for $t < T$ do
3:     Aggregation server broadcasts $\theta^t$ to clients.
4:     for $i$ in clients do
5:         Training local gradient $g_i^{t+1} = \frac{\partial \mathscr{L}(\theta^t, D_i)}{\partial \theta^t}$.
6:         Upload it to aggregation server.
7:     end for
```

(Continued)

Algorithm 1 (continued)

```
8:      Aggregation server aggregate local gradients $\theta^{t+1} = \theta^t - \eta \cdot \sum_{i=1}^{N} \frac{\|D_i\|}{\|D\|} \cdot g_i^{t+1}$.
9:      Calculate and store the compressed parameters $g' = \tilde{g} \times V$ and $\theta' = \tilde{\theta} \times V$.
10:     if $t \geq T_k$ and testing accuracy decrease then
11:         Calculate the trust factor of clients $\rho_i = \gamma \cdot \frac{g_i'^t \cdot \theta'^t}{\|g_i'^t\| \cdot \|\theta'^t\|} + (1-\gamma) \cdot \frac{g_i'^t \cdot g_i'^{t-1}}{\|g_i'^t\| \cdot \|g_i'^{t-1}\|}$.
12:         Remove the clients with lowest trust factors.
13:         Activate the efficient unlearning mechanism.
14:     end if
15: end for
16: Global model converges.
```

4.3 *Efficient Federated Unlearning Mechanism*

In this section, we propose an efficient federated unlearning mechanism. To mitigate the concealed influences and potential backdoors within the global model, it is essential for the model to undergo an unlearning process. In the Train from scratch, the global model rolls back to its initial state, and following numerous epochs of retraining, an optimal model is obtained. However, a significant limitation of this methodology is evident, as the requirement for multiple retraining is impractical. So we utilize a sliding window to limit the number of epochs to unlearn, while ensuring the "freshness" of the unlearning model. As shown in Fig. 3, when the unlearning process begins the global model rolls back to the state at the bottom of the window θ^{t-T_k} and the pointer also moves back to the bottom. Then we still require each remaining client to train an accurate gradient $\hat{g}_i^{t-T_k}$, the aggregation server aggregates them into the first unlearning global model $\hat{\theta}^{t-T_k}$. In the subsequent epochs k in the sliding window, the remaining clients are no longer required to train but only the aggregation server completes the unlearning process. The aggregation server retrieves the historical gradient parameters from the window, calibrates unlearning gradient updating directions, and updates the remaining clients' unlearning gradients accordingly as follows:

$$\hat{g}_i^{k+1} = \|\bar{g}_i^k\| \frac{\hat{g}_i^k}{\|\hat{g}_i^k\|} \tag{7}$$

where $\bar{g}_i^k$ is the original gradient deviated by malicious gradients, $\hat{g}_i^k$ is the retrained gradient or unlearning gradient and $\|\cdot\|$ is the l2-norm of a vector. In the equation, $\frac{\hat{g}_i^k}{\|\hat{g}_i^k\|}$ determines the direction for unlearning to update the corresponding gradient and $\|\bar{g}_i^k\|$ determines the distance for updating in this epoch. Thus, the local gradients of remaining clients are used to calibrate the unlearning gradients, then these unlearning gradients are aggregated into a new epoch of unlearning global model as follows:

$$\hat{\theta}^{k+1} = \hat{\theta}^k - \eta \sum_{i=1}^{m} \frac{\|D_i\|}{\|D/D_f\|} \cdot \hat{g}_i^{k+1} \tag{8}$$

where m is the number of remaining clients and $\|D\|$ is the size of a dataset. After the unlearning global model is aggregated, the pointer in the sliding window moves forward for one epoch. The aggregation server continue the calibration and aggregation steps of Eqs. (7) and (8) until the pointer moves to the start epoch of unlearning to obtain a clean unlearning global model. Then the federated learning system resumes until the global model converges. The specific efficient unlearning algorithm is shown in Algorithm 2.

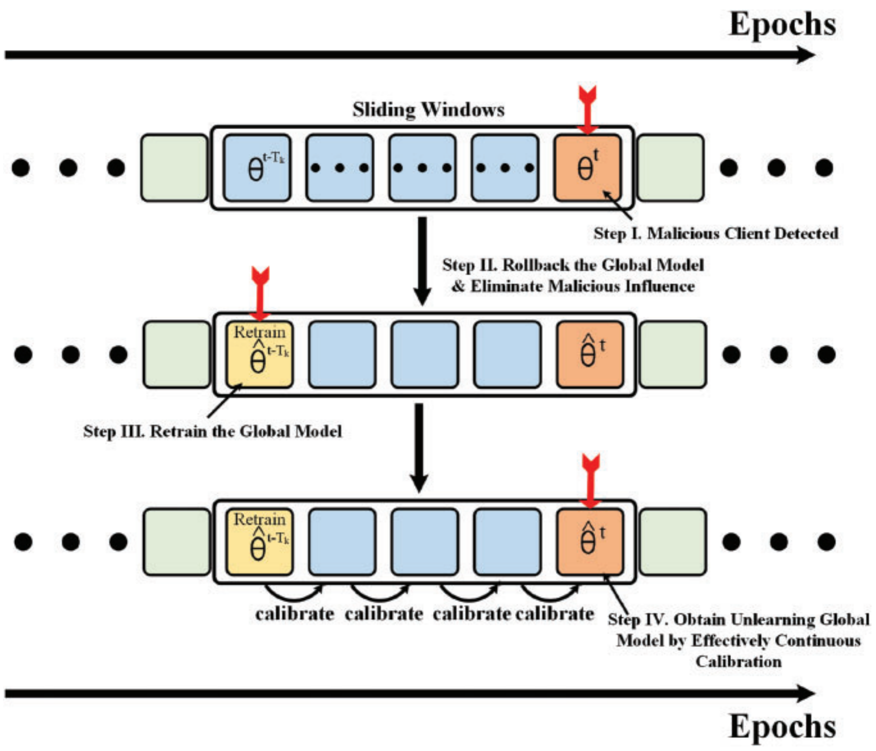
Epochs
Sliding Windows
θ^{t-T_k}
θ^t
Step I. Malicious Client Detected
Step II. Rollback the Global Model & Eliminate Malicious Influence
Retrain $\hat{\theta}^{t-T_k}$
$\hat{\theta}^t$
Step III. Retrain the Global Model
calibrate calibrate calibrate calibrate
Step IV. Obtain Unlearning Global Model by Effectively Continuous Calibration
Epochs

Figure 3: Unlearning in the sliding window

Algorithm 2: Efficient unlearning mechanism

```
Procedure:
 1: Aggregation server rolls back the global model $\theta^k = \theta^{t-T_k}$.
 2: for $k < t$ do
 3:     if $k == t - T_k$ then
 4:         All remaining clients locally train their unlearning gradients $\hat{g}_i^k$.
 5:         Aggregation server aggregates these gradients to unlearning global model $\hat{\theta}^k$
 6:     else
 7:         Aggregation server calibrates local gradients:
            $\hat{g}_i^{k+1} = \|\bar{g}_i^k\| \frac{\hat{g}_i^k}{\|\hat{g}_i^k\|}$.
 8:         Aggregation server aggregates these gradients to unlearning global model:
            $\hat{\theta}^{k+1} = \hat{\theta}^k - \eta \sum_{i=1}^{m} \frac{\|D_i\|}{\|D/D_f\|} \cdot \hat{g}_i^{k+1}$.
 9:     end if
10: end for
11: Obtain a clear unlearning global model $\hat{\theta}^t$.
```

4.4 Theoretical Analysis

In this section, we analyze the performance and effectiveness of the proposed unlearning mechanism. Firstly, a crucial feature of SlideFU is the design of sliding window mechanism, which obviates the necessity for extensive retraining. After the model rolls back, the influence of unlearning gradients is eliminated and the direction of model parameter updates is obtained through a single retraining. The model obtained through multiple epochs of calibration updates has almost no correlation with the unlearning gradients, making the model “forget” the gradients. Meanwhile to avoid excessive calibration deviation, each calibration amplitude is limited to the product of the parameter l2-norm and the learning rate which extremely reduces the computation and storage costs of unlearning. Recalling that Train from scratch requires a complete retraining, with a computational complexity of $O(T)$. In SlideFU, whenever it is necessary to unlearn, only one epoch of retraining is needed. Calibration only requires calculating the l2-norm and tensor product of the gradients. For each client to retrain the local model it needs hundreds of gradient descent and propagation operations on all neurons, while calibrating gradients merely requires matrix multiplication operation. Evidently, retraining demands far more computation and time compared to calibration thus we only focus on the computational complexity brought about by retraining here. So the computation complexity of SlideFU is only related to the unlearning times. At the same time, for poisoning attacks only activate few times malicious detection mechanism can eliminate almost all attackers. In other words, the number of operations that global model need to be unlearned in a complete federated learning is much smaller than that of training epochs$T$, and the computational complexity of SlideFU is also much smaller than $O(T)$. So SlideFU has an overwhelming advantage in costs compared to Train from scratch. The improved solution FedEraser reduces the cost of learning by calibrating historical gradients after retraining, while FedRecover provides an optimized estimation method. However both require the model to be returned to its initial state and retrained multiple times, which gives SlideFU a cost advantage with only a few retraining iterations.

Then, we present the assumption on which our effectiveness analysis is based and prove that the error between the global model recovered by our scheme and the fully retrained model is limited.

Assumption 1. *The error between the calibrated gradient and the true gradient of each client is bounded. Formally:*

$$\forall i, \forall k, \| g_i^k - \frac{\|\bar{g}_i^{k-1}\|}{\|\hat{g}_i^1\|} \hat{g}_i^1 \| \le \sigma \tag{9}$$

where σ is a finite positive value and g_i^1 is the first epoch retrained gradient.

Theorem 1. *Suppose Assumption 1 holds, after one epoch of retraining, the error between the calibrated and fully retrained global model is bounded as follows:*

$$\|\hat{\theta}^{T_k} - \theta^{T_k}\| \le \eta \cdot m(T_k - 1)\sigma \tag{10}$$

Proof of Theorem 1. Assuming the global model at the bottom of the sliding window is θ^0, the final calibrated global model is $\hat{\theta}^{T_k} = \theta^0 - \eta \sum_{i=1}^{m} \frac{\|D_i\|}{\|D/D_f\|} \cdot g_i^1 (1 + \frac{\sum_{j=1}^{Tk-1} \|\bar{g}_i^j\|}{\|g_i^1\|})$. Similarly, if using completely retrain the recovered global model is $\theta^{T_k} = \theta^0 - \eta \sum_{i=1}^{m} \frac{\|D_i\|}{\|D/D_f\|} (\sum_{k=1}^{T_k} g_i^k)$. Then, based on the triangle inequality and Assumption 1, it can be concluded that:

$$\begin{aligned} \|\hat{\theta}^{T_k} - \theta^{T_k}\| &= \|\eta \sum_{i=1}^{m} \frac{\|D_i\|}{\|D/D_f\|} (\sum_{k=2}^{T_k} g_i^k - \frac{\sum_{j=1}^{T_k-1} \|\bar{g}_i^j\|}{\|g_i^1\|} g_i^1)\| \\ &\le \eta \sum_{i=1}^{m} \frac{\|D_i\|}{\|D/D_f\|} (\|g_i^2 - \frac{\|\bar{g}_i^1\|}{\|\hat{g}_i^1\|} \hat{g}_i^1\| + \cdots + \|g_i^{Tk} - \frac{\|\bar{g}_i^{Tk-1}\|}{\|\hat{g}_i^1\|} \hat{g}_i^1\|) \\ &\le \eta \cdot m(T_k - 1)\sigma \end{aligned} \tag{11}$$

where $\eta \cdot m(T_k - 1)\sigma$ is a finite positive value. □

5 Evaluation

In this section, we conduct performance experiments on public datasets to demonstrate the effectiveness of our scheme. All the experiments are conduct on a workstation equipped with NVIDIA GeForce 4060 GPUs and 32 GB of RAM.

5.1 Experimental Setup

5.1.1 Datasets

In our experiments, two public datasets wide-used in federated learning are utilized to verify the performance of our scheme.

- **MNIST** [30]**:** A classic dataset of handwritten digits which is widely used for testing learning models. It contains 70,000 gray images of digital numbers from 0 to 9. Each image has been normalized to display key information at the center of the image.
- **Fashion-MNIST** [31]**:** A modern popular public dataset containing 70,000 gray fashion clothing images. Each sample is 28 × 28 pixels, corresponding to 10 clothing labels such as T-shirt, Dress, etc. Compared to MNIST, it is more complex and versatile.

5.1.2 Federated Learning Settings

In our experiment, we use a large-scale neural network model containing two convolutional layers on two datasets. To simulate federated learning scenarios, we generate an aggregation server and 50 clients, where attackers are proportionally hidden among them. Each client is assigned a local dataset and training at an identical learning rate of 0.0003, with a global training epochs of 50 and a local training epochs of 3. In addition, unless otherwise specified, the experiment defaults to using a sliding window size T_k of 3 and an adjustment factor γ of 0.5.

5.1.3 Attack Settings

In poisoning attack scenarios, attackers perform both untargeted and targeted attacks. To distinguish between minority poisoning attacks and majority poisoning attacks, we set the proportion of attackers on the client to 20% and 40%, respectively. When performing untargeted attack, the attackers introduce corrupted data into local datasets and randomly modify the parameters of gradients. Conversely, when performing targeted attacks, the attackers manipulate the datasets by flipping the labels of specific samples.

5.2 Experimental Result

In this section, we evaluate and analyze the experimental performance of our scheme under different attack scenarios. Two different metrics are used to evaluate the performance of our scheme, namely model accuracy and recovery to training time ratio (RTR), where RTR refers to the ratio of the time spent on learning to the training time spent on federated learning. To strengthen the conclusion, we add the F1 score which reflects the performance of model and conduct multiple statistical significance tests using SlideFU. If the p-value of F1 scores is greater than $\alpha = 0.05$, it proves the stability of our scheme. Meanwhile, we introduce several other unlearning strategies to demonstrate the advantages of our scheme. Train from scratch [21] is a complete retraining during the unlearning process. History-only just uses the historical gradient aggregation of remaining clients. FedEraser [24] alternates between retraining and calibration. In addition, we introduce FedRecover [25] and FedRemover [27], two recent efficient recovery solutions for comparison.

5.2.1 Model Performance under Untargeted Attacks

As shown in Fig. 4, even when encountering a small proportion of untargeted attacks, the accuracy of the federated learning model decreases significantly. The poor performance of using only historical gradients to unlearn the global model (History-only) proves that other benign gradients have also been poisoned by malicious gradients. Under this method, the performance of unlearning model may not even be as good as before. Train-from-scratch maintained the best model accuracy due to complete retraining. Although there are slightly decrease in model accuracy, FedEraser still ensures the high accuracy of global model. FedRemover and FedRecover have better recovery effects and achieve some advantages. SlideFU also maintains a high level of model accuracy, slightly better than FedEraser under small proportion untargeted attacks and almost the same under large proportion untargeted attacks. In addition, the p-values of SlideFU on the two datasets are 0.6107 and 0.7326 respectively indicating that our scheme maintains the stability of the model under untargeted attacks.

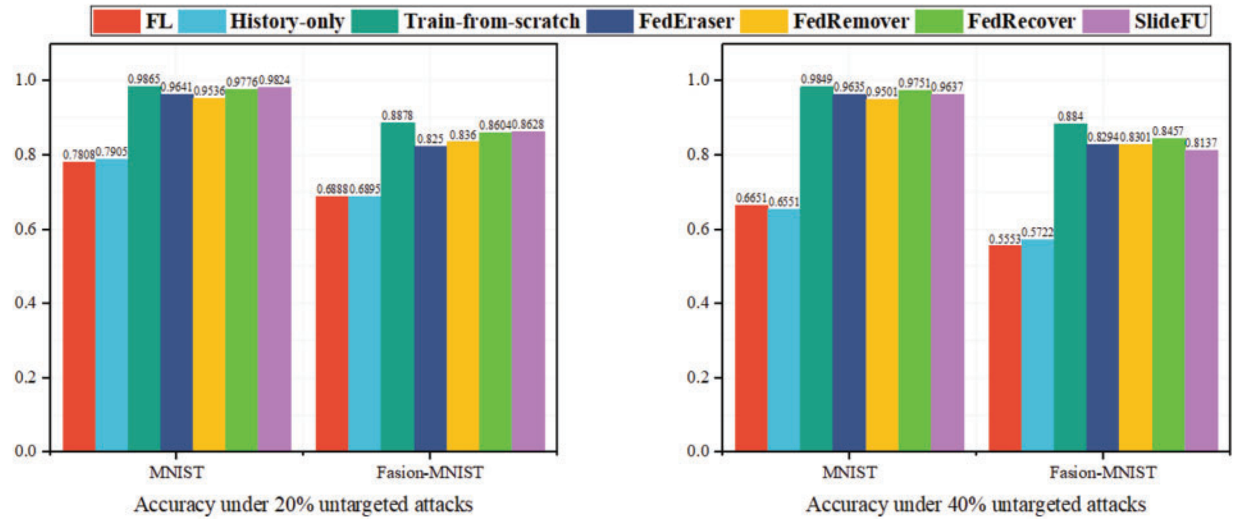

Figure 4: Model accuracy under different ratio untargeted attacks

5.2.2 Time Consumption under Untargeted Attacks

In the federated learning system threatened by poisoning attacks, prolonged unlearning processes not only consume resources but also increase the risk of being attacked. As shown in Fig. 5, the RTR of Train from scratch is the highest, approaching the time required for federated learning training. History-only causes a slight resource consumption, however as shown in the previous experiment, the unlearning model obtained by this way is unstable and unreliable. FedEraser has a significant effect in reducing the cost of unlearning, but as shown in the figure there is still great room for improvement. FedRemover and FedRecover further reduce the cost of recovery. In contrast the RTR of SlideFU is extremely low, only slightly higher than that of History-only, while SlideFU maintains a significant advantage in accuracy compared with History-only. We speculate that this is because the sliding window framework we designed only retraining once during the unlearning process, and the cost of the remaining calibration operations is almost negligible. This proves that our scheme has low unlearning costs in the face of untargeted attacks. To demonstrate the advantages of our scheme from data, Table 2 shows the model accuracy, F1-score and RTR under 40% untargeted attacks.

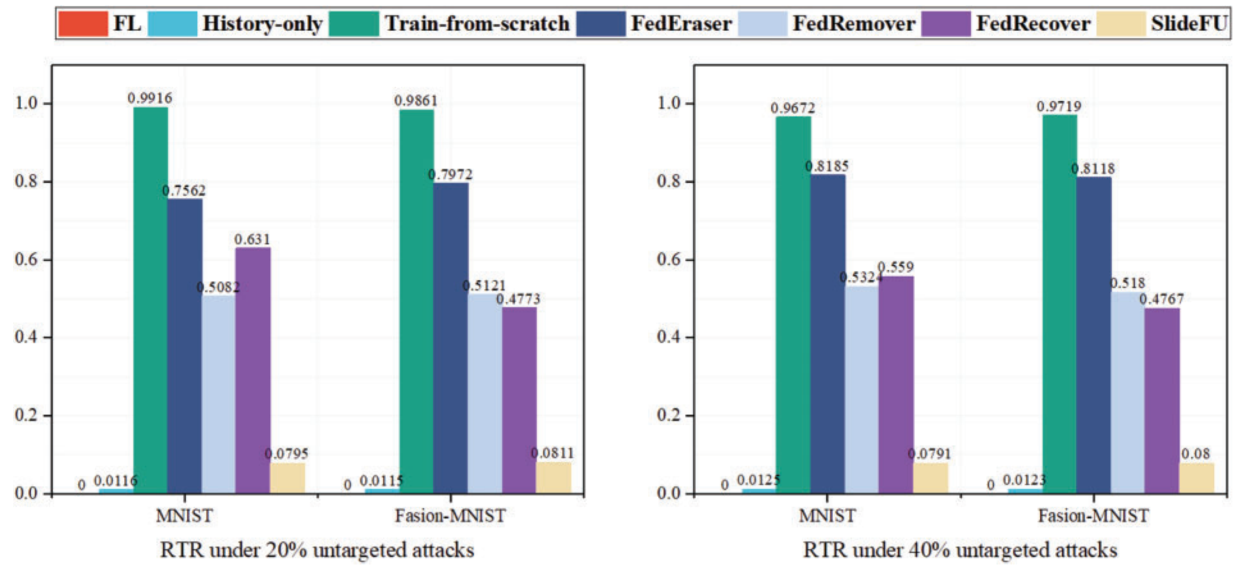

Figure 5: RTR under different ratio untargeted attacks

Table 2: Model accuracy, F1-score and RTR under 40% untargeted attacks on two datasets

Method	**MNIST**			**Fashion-MNIST**		
	Accuracy	**F1-score**	**RTR**	**Accuracy**	**F1-score**	**RTR**
FL	0.6651	0.7311	–	0.5553	0.6579	–
History-only	0.6551	0.7376	0.0125	0.5722	0.6556	0.0123
Train-from-scratch [21]	0.9849	0.9858	0.9672	0.884	0.8859	0.9719
FedEraser [24]	0.9635	0.9646	0.8185	0.8294	0.8409	0.8118
FedRemover [27]	0.9501	0.9432	0.5324	0.8301	0.8346	0.518
FedRecover [25]	0.9751	0.978	0.559	0.8457	0.8595	0.4767
SlideFU	0.9637	0.9666	0.0791	0.8137	0.8203	0.08

5.2.3 Model Performance under Targeted Attacks

As shown in Fig. 6, the impact of targeted attacks on model accuracy is not significant as that of untargeted attacks, and the global model can still maintain high accuracy when encountering small-scale attacks. We speculate that this is due to the relatively small impact of targeted attacks on gradients, which allows federated learning model to maintain good performance under small-scale attacks. However, as the attack rate increases, the accuracy of federated learning model significantly decreases. Among them, Train from scratch shows a slight advantage in accuracy while History-only shows instability. Similarly, FedEraser and SlideFU maintain comparable model accuracy, and in most cases the model unlearned by our scheme has higher accuracy than that of FedEraser. We think is due to earlier elimination of attackers leading to better training performance. In addition, the p-values of SlideFU on the two datasets are 0.5961 and 0.7484 respectively indicating that our scheme maintains the stability of the model under targeted attacks.

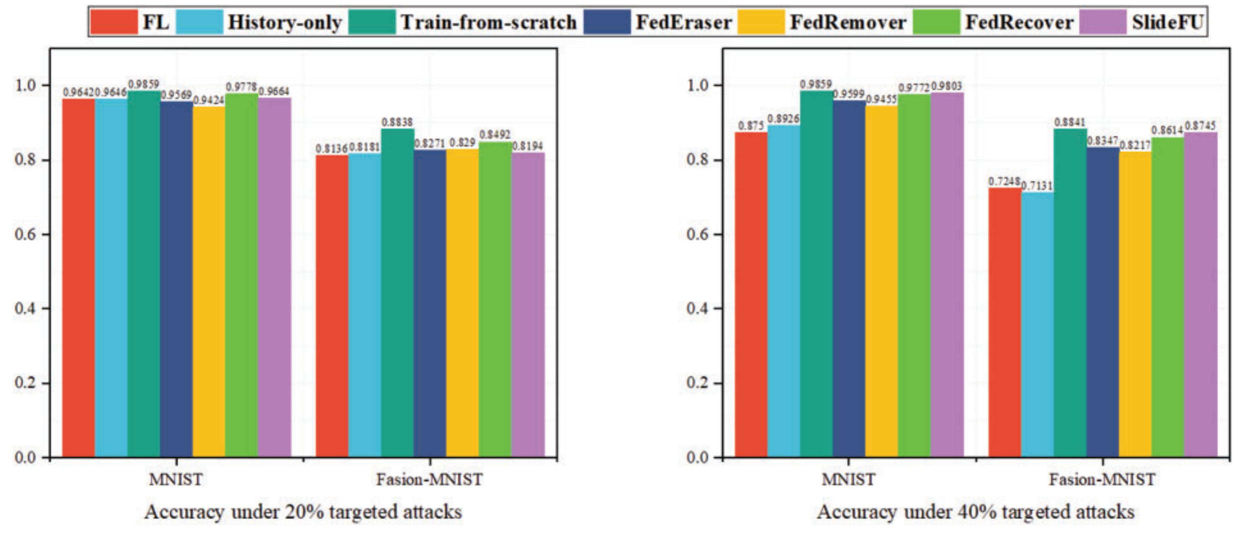

Figure 6: Model accuracy under different ratio targeted attacks

5.2.4 Time Consumption under Targeted Attacks

As shown in Fig. 7, the RTR of each scheme is similar to that of untargeted attacks when encountering targeted attacks. The RTR of SlideFU is stable, although there is some slight fluctuation in that of Train from scratch and FedEraser. We speculate that the unstable RTR is caused by the influence of poisoning attacks during the training and retraining processes, where both Train from scratch and FedEraser require multiple epochs of retraining while SlideFU only needs one epoch retraining per unlearning session. Similarly, FedRemover and FedRecover also introduce more training and estimation costs, making their costs higher. This also indicates that our scheme provides attackers with fewer opportunities for attack and has great advantages in costs under targeted attacks. To demonstrate the advantages of our scheme from data, Table 3 shows the model accuracy, F1-score and RTR under 40% targeted attacks.

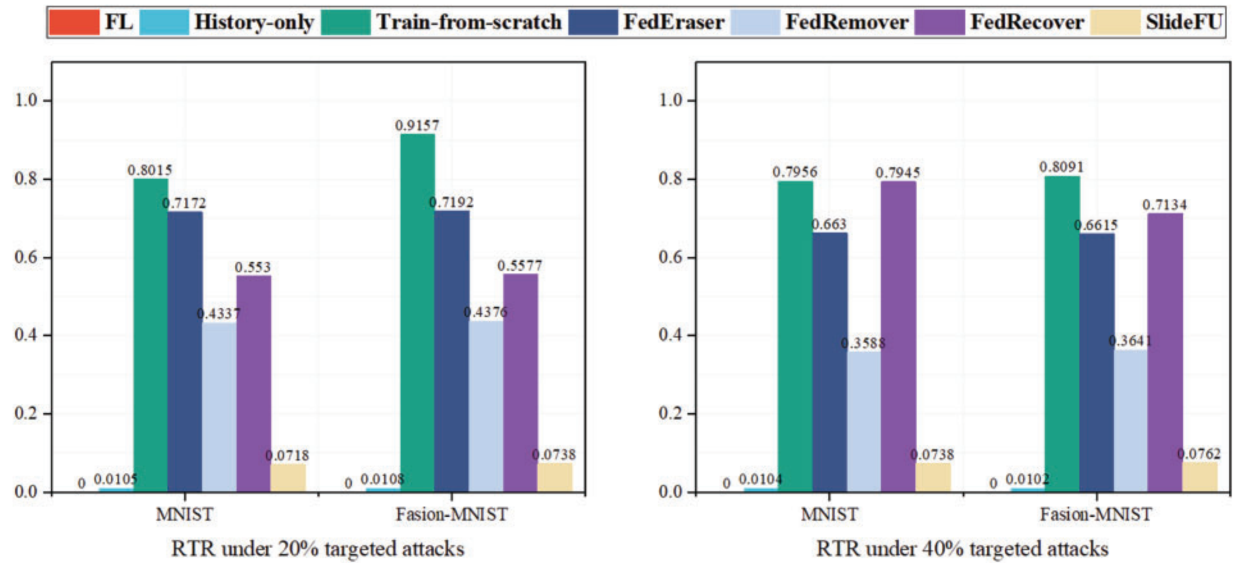

Figure 7: RTR under different ratio targeted attacks

Table 3: Model accuracy, F1-score and RTR under 40% targeted attacks on two datasets

Method	MNIST			Fashion-MNIST		
	Accuracy	F1-score	RTR	Accuracy	F1-score	RTR
FL	0.875	0.8834	–	0.7248	0.736	–
History-only	0.8926	0.8774	0.0105	0.7131	0.7343	0.0102
Train-from-scratch [21]	0.9859	0.9873	0.7956	0.8841	0.8898	0.8091
FedEraser [24]	0.9599	0.9653	0.663	0.8347	0.8425	0.6615
FedRemover [27]	0.9455	0.9439	0.3588	0.8217	0.8382	0.3641
FedRecover [25]	0.9772	0.978	0.7945	0.8614	0.8599	0.7134
SlideFU	0.9803	0.9822	0.0738	0.8745	0.8717	0.0762

5.2.5 The Impacts of Sliding Window Size

In order to test the effect of some hyperparameters in SlideFU on unlearning performance, as shown in Fig. 8, we evaluate and record the accuracy of the unlearning model under different sliding window size. When facing untargeted attacks, the accuracy of the model gradually increases slightly with the increase of window size. Untargeted attacks cause significant gradient deviation, so multiple consecutive gradient calibrations have a better effect. However, when facing targeted attacks, a larger sliding window size does not bring higher accuracy. This is due to the small gradient deviation caused by targeted attacks, which leads to the limited effectiveness of multiple consecutive gradient calibrations. It can be foreseen that too many calibrations may also lead to gradient misalignment. Therefore, in the selection of sliding window size, a moderate size should be a clear choice.

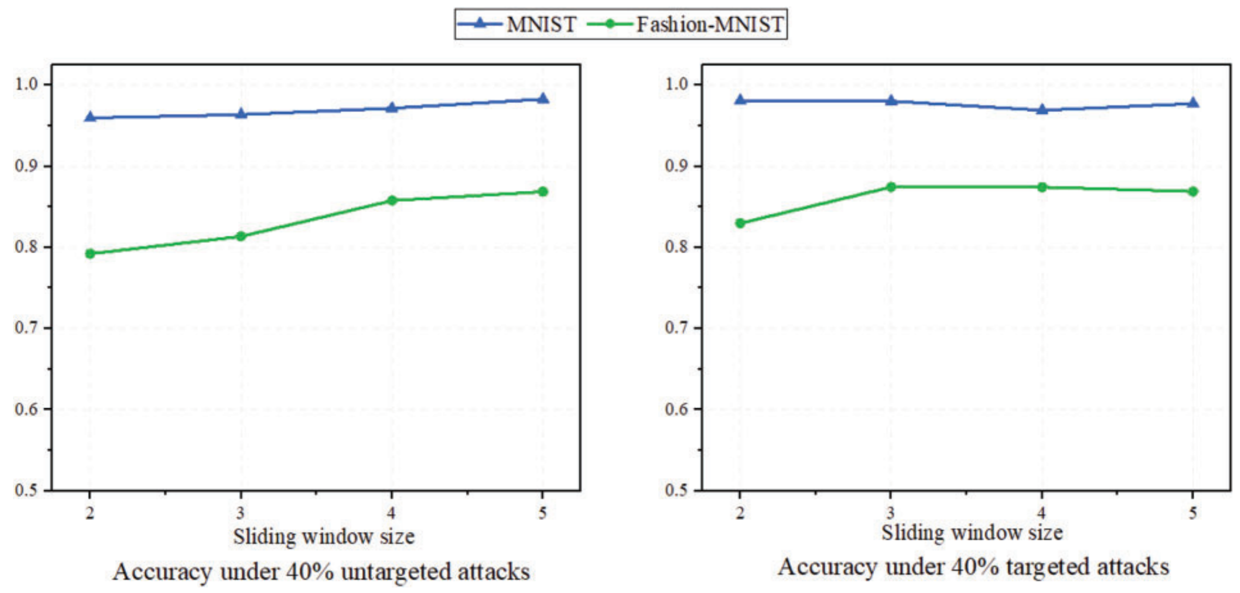

Figure 8: Model accuracy under 40% attacks in different size of sliding window

5.3 The Impacts of Adjustment Factor

To investigate the impact of the weights of two malicious detection methods on the detection performance of attackers, we evaluate the success rate of attacker detection under different adjustment factor γ. As shown in Fig. 9, the accuracy of malicious detection fluctuates slightly due to the imbalanced sample distribution in federated learning scenarios. However, when the adjustment factor is moderate, the detection success rate of each proportion of untargeted or targeted attackers is relatively higher. Then when the adjustment factor is too high or low, the success rate of detection decreases slightly. In general, the moderate adjustment factor can cope with most poisoning attack scenarios.

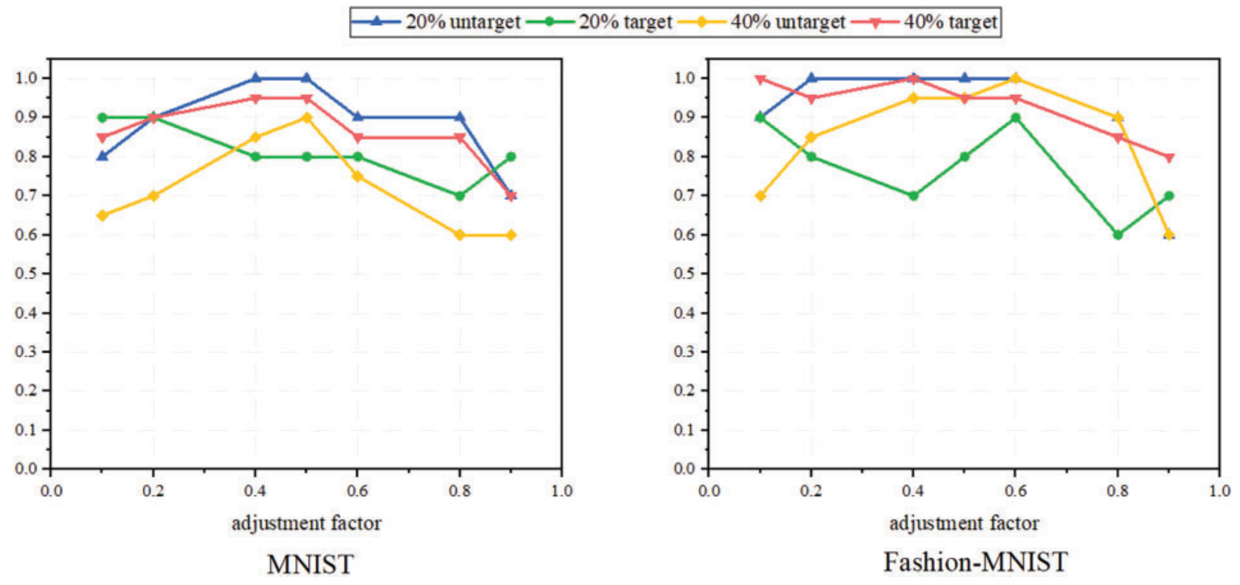

Figure 9: Malicious detection accuracy under different ratio attacks

5.4 Effects on Different Model

To demonstrate the performance of our scheme on different neural network models, we test the model performance on CNN, ResNet, and RNN separately. As shown in Table 4, our scheme can still ensure the recovery effect on different models, even on the RNN model that is most affected by poisoning attacks.

Table 4: Model accuracy under 40% untargeted attacks on different models

Model	CNN		ResNet		RNN	
	MNIST	FMNIST	MNIST	FMNIST	MNIST	FMNIST
FL	0.6651	0.5553	0.6136	0.5752	0.1079	0.1581
Retrain	0.9849	0.884	0.9786	0.8494	0.957	0.8265
SlideFU	0.9637	0.8137	0.9619	0.8127	0.9341	0.7911

6 Conclusion

Poisoning attackers disguised as ordinary clients pose a threat to federated learning model with their stealthiness and destructiveness, while the training gradients of normal clients are also vulnerable to poisoning attacks due to distributed learning characteristics. Therefore, we propose SlideFU, a federated unlearning framework specifically designed to resist poisoning attacks. SlideFU changes the traditional federated unlearning method by designing a sliding window based framework, in which all training and unlearning processes are confined to a single sliding window. In instances where the accuracy of the federated learning model decreases, the trust factors of the clients are calculated based on their compressed gradients, which serves to identify and exclude potential attackers from the system. In addition, to eliminate the influence of poisoning attackers, an efficient calibration method is used to unlearn the global model. The simulation experiments on two public datasets demonstrate that our scheme can achieve excellent model accuracy in both untargeted and targeted attack scenarios with extremely low time costs.

Acknowledgement: The authors wish to express their appreciation to the reviewers and the editor for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: This work was supported in part by the National Social Science Foundation of China under Grant 20BTQ058; in part by the Natural Science Foundation of Hunan Province under Grant 2023JJ50033.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Long Cai, Ke Gu; data collection: Long Cai; analysis and interpretation of results: Long Cai, Ke Gu, Jiaqi Lei; draft manuscript preparation: Long Cai, Ke Gu, Jiaqi Lei. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used in this paper are respectively in references [30,31].

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Singh A, Zhu J, editors. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics; 2017. p. 1273–82.
2. Niknam S, Dhillon HS, Reed JH. Federated learning for wireless communications: motivation, opportunities, and challenges. IEEE Commun Mag. 2020;58(6):46–51. doi:10.1109/MCOM.001.1900461.
3. Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J. Machine learning with adversaries: byzantine tolerant gradient descent. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S et al., editors. Advances in neural information processing systems. Vol. 30. Long Beach, CA, USA: Curran Associates, Inc.; 2017.
4. Nasr M, Shokri R, Houmansadr A. Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE Symposium on Security and Privacy (SP); 2019; San Francisco, CA, USA. p. 739–53.
5. Jiang W, Li H, Liu S, Luo X, Lu R. Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles. IEEE Trans Vehicular Technol. 2020;69(4):4439–49. doi:10.1109/TVT.2020.2977378.
6. Zhu H, Ling Q. Byzantine-robust aggregation with gradient difference compression and stochastic variance reduction for federated learning. In: ICASSP 2022—2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2022; Singapore. p. 4278–82.
7. Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: towards optimal statistical rates. In: Dy J, Krause A, editors. Proceedings of the 35th International Conference on Machine Learning; 2018. p. 5650–9.

8. So J, Güler B, Avestimehr AS. Byzantine-resilient secure federated learning. IEEE J Sel Areas Commun. 2021;39(7):2168–81. doi:10.1109/JSAC.2020.3041404.
9. Lu S, Li R, Chen X, Ma Y. Defense against local model poisoning attacks to byzantine-robust federated learning. Frontiers Comput Sci. 2022;16(6):166337. doi:10.1007/s11704-021-1067-4.
10. Li S, Ngai E, Voigt T. Byzantine-robust aggregation in federated learning empowered industrial IoT. IEEE Trans Ind Inform. 2023;19(2):1165–75. doi:10.1109/TII.2021.3128164.
11. Wu L, Guo S, Wang J, Hong Z, Zhang J, Ding Y. Federated unlearning: guarantee the right of clients to forget. IEEE Network. 2022;36(5):129–35. doi:10.1109/MNET.001.2200198.
12. Wang J, Guo S, Xie X, Qi H. Federated unlearning via class-discriminative pruning. In: Laforest F, Troncy R, Simperl E, Agarwal D, Gionis A, Herman I et al., editors. WWW '22: The ACM Web Conference 2022; 2022 Apr 25–29; Lyon, France: ACM; 2022. p. 622–32. doi:10.1145/3485447.3512222.
13. Gong J, Simeone O, Kang J. Bayesian variational federated learning and unlearning in decentralized networks. In: 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC); 2021; Lucca, Italy. p. 216–20.
14. Xu J, Wu Z, Wang C, Jia X. Machine unlearning: solutions and challenges. IEEE Trans Emerg Top Comput Intell. 2024;8(3):2150–68. doi:10.1109/TETCI.2024.3379240.
15. Cao D, Chang S, Lin Z, Liu G, Sun D. Understanding distributed poisoning attack in federated learning. In: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS); 2019; Tianjin, China. p. 233–9.
16. Cao X, Fang M, Liu J, Gong NZ. FLTrust: byzantine-robust federated learning via trust bootstrapping. In: 28th Annual Network and Distributed System Security Symposium, NDSS 2021; 2021 Feb 21–25; The Internet Society; 2021.
17. Sattler F, Müller KR, Wiegand T, Samek W. On the byzantine robustness of clustered federated learning. In: ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 2020; Barcelona, Spain. p. 8861–5.
18. Zhai K, Ren Q, Wang J, Yan C. Byzantine-robust federated learning via credibility assessment on non-IID data. arXiv:2109.02396. 2021.
19. Erdol ES, Ustubioglu B, Erdol H, Ulutas G. Low dimensional secure federated learning framework against poisoning attacks. Future Gener Comput Syst. 2024;158(13):183–99. doi:10.1016/j.future.2024.04.017.
20. Huang Y, Yang G, Zhou H, Dai H, Yuan D, Yu S. VPPFL: a verifiable privacy-preserving federated learning scheme against poisoning attacks. Computers Security. 2024;136(5):103562. doi:10.1016/j.cose.2023.103562.
21. Hu T. Dense in dense: training segmentation from scratch. In: Jawahar CV, Li H, Mori G, Schindler K, editors. Computer vision–ACCV 2018. Cham: Springer International Publishing; 2019. p. 454–70.
22. Cao Y, Yang J. Towards making systems forget with machine unlearning. In: 2015 IEEE Symposium on Security and Privacy; 2015; San Jose, CA, USA. p. 463–80.
23. Ginart A, Guan MY, Valiant G, Zou J. Making AI forget you: data deletion in machine learning. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R, editors. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019; 2019 Dec 8–14; Vancouver, BC, Canada: Curran Associates; 2019. p. 3513–26.
24. Liu G, Ma X, Yang Y, Wang C, Liu J. FedEraser: enabling efficient client-level data removal from federated learning models. In: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS); 2021; Tokyo, Japan. p. 1–10.
25. Cao X, Jia J, Zhang Z, Gong NZ. FedRecover: recovering from poisoning attacks in federated learning using historical information. In: 2023 IEEE Symposium on Security and Privacy (SP); 2023; San Francisco, CA, USA. p. 1366–83.
26. Zhang L, Zhu T, Zhang H, Xiong P, Zhou W. FedRecovery: differentially private machine unlearning for federated learning frameworks. IEEE Trans Inf Forensics Secur. 2023;18:4732–46. doi:10.1109/TIFS.2023.3297905.
27. Yuan Y, Wang B, Zhang C, Xiong Z, Li C, Zhu L. Toward efficient and robust federated unlearning in IoT networks. IEEE Internet Things J. 2024;11(12):22081–90. doi:10.1109/JIOT.2024.3378329.

28. Jebreel NM, Domingo-Ferrer J. FL-defender: combating targeted attacks in federated learning. Knowl Based Syst. 2023;260(2):110178. doi:10.1016/j.knosys.2022.110178.
29. Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometr Intell Lab Syst. 1987;2(1):37–52. doi:10.1016/0169-7439(87)80084-9.
30. LeCun Y. The MNIST database of handwritten digits; 1998 [cited 2025 Feb 13]. Available from: http://yann.lecun.com/exdb/mnist/.
31. Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747. 2017.