

Sparsification Under Siege: Defending Against Poisoning Attacks in Communication-Efficient Federated Learning

Zhiyong Jin

*Beihang University
Beijing, China*

Runhua Xu

*Beihang University
Beijing, China*

Chao Li

*Beijing Jiaotong University
Beijing, China*

Yizhong Liu

*Beihang University
Beijing, China*

Jianxin Li

*Beihang University
Beijing, China*

sy2406106@buaa.edu.cn

runhua@buaa.edu.cn

li.chao@bjtu.edu.cn

liuyizhong@buaa.edu.cn

lijx@act.buaa.edu.cn

Abstract—Federated Learning (FL) enables collaborative model training across distributed clients while preserving data privacy, yet it faces significant challenges in communication efficiency and vulnerability to poisoning attacks. While sparsification techniques mitigate communication overhead by transmitting only critical model parameters, they inadvertently amplify security risks: adversarial clients can exploit sparse updates to evade detection and degrade model performance. Existing defense mechanisms, designed for standard FL communication scenarios, are ineffective in addressing these vulnerabilities within sparsified FL. To bridge this gap, we propose FLARE, a novel federated learning framework that integrates sparse index mask inspection and model update sign similarity analysis to detect and mitigate poisoning attacks in sparsified FL. Extensive experiments across multiple datasets and adversarial scenarios demonstrate that FLARE significantly outperforms existing defense strategies, effectively securing sparsified FL against poisoning attacks while maintaining communication efficiency.

Index Terms—Federated Learning, Poisoning Attacks, Robustness, Sparsification, Communication Efficiency

I. INTRODUCTION

Federated Learning (FL) [1] has emerged as a transformative paradigm for decentralized machine learning, enabling multiple clients to collaboratively train a shared global model while retaining their raw data locally. By maintaining data localized, FL enhances privacy and minimizes the risk of data leakage, making it highly applicable to privacy-sensitive domains such as healthcare, finance, and distributed IoT networks [2]–[4]. Despite these advantages, the iterative nature of FL necessitates frequent transmission of model parameters between clients and the central server, leading to substantial communication overhead [5]. This limitation is especially pronounced in resource-constrained environments, where bandwidth limitations and network instability can severely impact the scalability and practicality of the system.

To address the communication challenges in FL, researchers have proposed various communication-efficient techniques [6]–[8], among which sparsification has emerged as a particularly effective approach. By transmitting only the most significant model updates, sparsification substantially reduces communication costs while maintaining model convergence. Existing sparsification methods, such as Top- k sparsification

[9], have demonstrated considerable success in improving the efficiency of FL. However, these methods predominantly focus on minimizing communication overhead and accelerating convergence, often overlooking the security implications of sparsified communication. Due to the decentralized nature of FL, the system may remain highly susceptible to poisoning attacks, where adversarial clients inject malicious updates to degrade the global model’s performance. This vulnerability is further amplified in sparsified FL, as the reduction in communicated parameters makes adversarial manipulations harder to detect and mitigate. To the best of our knowledge, the security risks introduced by sparsification in communication-efficient FL remain an underexplored area of research.

In contrast to standard FL, where robust aggregation methods such as Median [10], Trimmed Mean [10], and Krum [11] have demonstrated effectiveness in mitigating poisoning attacks, sparsified FL introduces distinct challenges. The selective transmission of updates can obscure malicious modifications, enabling adversarial clients to evade detection while exerting significant influence over the global model. These vulnerabilities highlight an urgent need for the development of defense mechanisms specifically tailored for sparsified FL, ensuring that communication efficiency achieved through sparsification does not compromise the system’s robustness against adversarial threats.

In this work, we systematically investigate the vulnerabilities of FL under poisoning attacks in the context of sparsified communication-efficient FL. Our analysis demonstrates that existing defense mechanisms, originally designed for standard FL, become ineffective when applied to sparsified FL. These findings highlight the pressing need for security-aware sparsification techniques to ensure the robustness of FL systems.

To address this critical challenge, we propose FLARE, a novel federated learning framework specifically designed to achieve robust aggregation while maintaining sparsified communication efficiency. FLARE incorporates sparse index mask inspection combined with model update sign similarity analysis to detect and mitigate poisoning attacks in sparsified communication-efficient FL. By bridging the gap between communication efficiency and security in FL, this work estab-

lishes a foundation for the development of robust sparsification strategies, ensuring that FL remains both communication-efficient and resilient in adversarial environments. Through extensive experiments conducted on diverse datasets and under various attack scenarios, we demonstrate that *FLARE* effectively secures sparsified FL against poisoning attacks while preserving its communication efficiency.

Our key contributions are summarized as follows:

- We systematically analyze the security vulnerabilities of sparsified FL under poisoning attacks and demonstrate that existing defense mechanisms, originally designed for standard FL, are ineffective in sparsified settings.
- We propose *FLARE*, a novel FL framework tailored for achieving robust aggregation against poisoning attacks while maintaining sparsified communication efficiency, which integrates sparse index mask inspection and model update sign similarity analysis to detect and mitigate adversarial behaviors.
- We conduct extensive empirical evaluations on multiple datasets and across diverse adversarial scenarios. The results demonstrate that *FLARE* outperforms existing defense mechanisms, effectively safeguarding sparsified FL against various poisoning attacks while maintaining communication efficiency.

II. RELATED WORK

A. Communication-Efficient FL

Enhancing communication efficiency is critical for improving overall performance of FL systems. The substantial volume of data exchanged between clients and servers imposes significant challenges, including increased latency, computational overhead, and excessive bandwidth consumption, which can hinder the system's real-time capabilities. To address these issues, techniques such as sparsification [12], [13], quantization [6], [14], and asynchronous communication [7], [15], [16] have been proposed. These methods effectively reduce communication overhead, enabling FL systems to operate efficiently in resource-constrained environments while preserving model performance.

Sparsification methods have shown greater promise compared to alternative techniques, as quantization is not suitable for large-scale models or low-bandwidth network environments [17], and the staleness issue introduced by asynchronous aggregation can affect model convergence [16]. Sparsification aims to reduce communication overhead by selecting only a subset of important gradients or model parameters for transmission during each training round, while less significant information is either ignored or compressed. Aji et al. [18] proposed a method that selects the top-k components with the highest absolute values from the gradients and designates the remaining components as residuals. Subsequent studies [5] have demonstrated that Top- k sparsification is less sensitive to the impact of non-IID data in FL. In the context of peer-to-peer federated learning(P2PFL), Wang et al. [13] introduced a momentum-based Top- k sparsification method, termed SparSFA, to further enhance communication efficiency.

B. Model Poisoning Attacks and Defense Strategies in FL

The distributed architecture of FL makes it particularly vulnerable to poisoning attacks, where adversarial clients manipulate local updates to compromise the performance of the global model. Several types of untargeted poisoning attacks have been proposed in the literature. Label flip attack [19] manipulates local training data by flipping class labels, thereby injecting mislabeled samples that degrade the model's generalization. Gaussian noise attack [19] generates model updates by adding random noise, leading to instability and reduced convergence quality. Inner product manipulation [20] strategically crafts adversarial updates to maximize the inner product with benign gradients, thereby amplifying their negative impact on model convergence. Scaling attacks [21] introduce malicious updates that are disproportionately scaled, either by inflating or deflating gradient magnitudes, which disrupts the federated aggregation process and significantly impairs the effectiveness of the global model. These attacks demonstrate the vulnerability of FL to malicious participants, highlighting the need for robust defense mechanisms to mitigate their effects.

To mitigate the impact of poisoning attacks in FL, various defense strategies have been proposed. These strategies can be broadly categorized into robust aggregation rules and anomaly detection techniques. Robust aggregation methods, such as Median [10] and Trimmed Mean [10], aim to improve the resilience of the global model by mitigating the impact of malicious updates during the aggregation process. Median aggregation selects the component-wise median of all client updates, making it robust to outliers and adversarial manipulations. Trimmed Mean excludes a fraction of the largest and smallest update values before computing the mean, effectively mitigating the impact of extreme deviations introduced by adversaries. Anomaly detection techniques take a different approach by identifying and filtering out malicious clients before aggregation. In addition, Krum [11], a widely used anomaly detection-based method, selects a single client update that is closest to its neighbors in Euclidean space, assuming that the majority of clients are honest. By prioritizing updates that exhibit consistency with the majority, Krum effectively reduces the impact of outliers. However, these defense methods were primarily designed for standard FL and have not yet addressed attacker detection in sparse training scenarios.

C. Sparsified Privacy-Preserving FL

Recent studies [12], [22] have explored sparsification techniques to enhance communication efficiency and address privacy concerns in FL. However, these methods primarily integrate sparsification into privacy-preserving FL without addressing adversarial threats like poisoning attacks. For example, FedPHE [12] combines homomorphic encryption with sparsification to reduce communication costs while protecting client privacy by preventing the server from accessing individual updates. Similarly, Ergun et al. [22] propose a method that uses sparsification with Secure Multi-Party Computation

(SMPC) to ensure data privacy by sharing only aggregated updates with the server.

Other studies [13], [23] have investigated sparsification techniques such as sampling the global model to counter adversarial updates or focusing on peer-to-peer frameworks. However, these approaches are not directly applicable to standard communication-efficient FL frameworks. This highlights an urgent need for poisoning-aware secure aggregation methods specifically designed for communication-efficient FL systems.

III. FORMULATION AND SECURITY MODEL

A. Formulation

The overall goal of FL is to collaboratively train a model on a group of clients and minimize the global loss. Assume that there are N clients in the FL environment setting, the input space X and the target space Y represent the input features and output targets of the model respectively. Each client i has a local dataset $D_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{n_i}$, where n_i is the number of samples. The global objective function $f(w)$ on the server is defined as $\min_w f(w) = \sum_{i=1}^N p_i f_i(w)$, where f_i is the objective function of client i and p_i is the aggregate weight of client i . The aggregate weight is typically set $\frac{1}{N}$ or proportional to the size of the local dataset held by the client. Then, the local objective function value of the client i is obtained through local data D_i , formulate as $f_i(w) = \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(F_i(x_{i,j}; w), y_{i,j})$, where ℓ denotes loss function, F_i denotes the model function of client i and $(x_{i,j}, y_{i,j}) \in D_i$ denotes data samples. At the same time, each client applies stochastic gradient descent to update its local model parameters with a fixed mini-batch size m in one or more iterations, formalized as $w_i^{t+1} = w_i^t - \eta \nabla f_i(w)$, where η is the local model learning rate of client i . Subsequently, the local model parameters w_i^{t+1} are communicated back to the server for aggregation.

B. Security Model

In the FLARE framework, we consider a Byzantine threat model in which the proportion of adversarial participating clients is assumed to remain below 50% of the total clients. Each adversarial client is capable of launching poisoning attacks, aiming to manipulate the global model aggregation process by transmitting falsified model parameters during the FL process. This can be achieved either by directly manipulating the local model or by indirectly falsifying the local training samples. Consistent with the threat assumptions of existing poisoning attack studies, we assume that attackers do not have direct access to the local model updates or raw training data of benign participants. Furthermore, in this work, the primary adversarial objective is to conduct untargeted attacks designed to degrade the overall performance of the global model. It is important to note that privacy leakage threats, such as those caused by privacy inference attacks or gradient leakage attacks, are beyond the scope of this paper.

IV. DESIGN DETAILS OF FLARE

A. Overview of FLARE

Existing sparsified communication efficiency approaches have been observed to obscure malicious modifications in poisoning attack scenarios, enabling adversarial clients to evade detection (demonstrated and discussed in Section V-C). Consequently, existing defense mechanisms, originally designed for standard FL, become ineffective when applied to communication-efficient FL frameworks. These findings underscore the urgent need for security-aware sparsification techniques to ensure both robustness and communication efficiency in FL systems. To address this challenge, the goal of the proposed FLARE framework is to enhance robustness against untargeted poisoning attacks in the context of sparsified, communication-efficient FL.

As illustrated in Fig. 1, FLARE is built on a common communication-efficient FL paradigm, where each client communicates with the server by transmitting only the most important model parameters using sparsification techniques. Specifically, each client performs local training on its data, applies top- k sparsification, and uploads the sparse model parameters along with a sparse index mask to the server. The server then aggregates the received sparse model updates to form the global model based on the sparse index mask. This sparsified model aggregation approach significantly reduces communication overhead, as only a fraction of the model parameters are exchanged.

To further enhance the robustness of FLARE against poisoning attacks—whether through injecting incorrect model parameters or manipulating the sparsification process—two key components are designed at the server side: sparse index mask inspection and model update sign similarity analysis. Specifically, the sparse index mask inspection module filters out clients whose model updates are inconsistent with the majority by measuring the overlap in parameter selections across clients using the Jaccard similarity metric. The model update sign similarity analysis module, another critical component of FLARE, detects adversarial behavior patterns by grouping clients with similar update directions (i.e., the signs of parameter differences) using cosine similarity and applying density-based clustering to identify malicious clients. Clients identified as potentially malicious are excluded from the subsequent sparsified communication-efficient aggregation process, ensuring that only benign clients contribute to the final global model.

In the following sections, we first introduce the robustness enhancement mechanism designed in FLARE, which addresses the challenges posed by poisoning attacks in sparsification-based communication within FL scenarios. This mechanism defends against potential poisoning attacks while maintaining communication efficiency. Subsequently, we integrate the proposed defense strategy into the standard sparsified, communication-efficient FL paradigm.

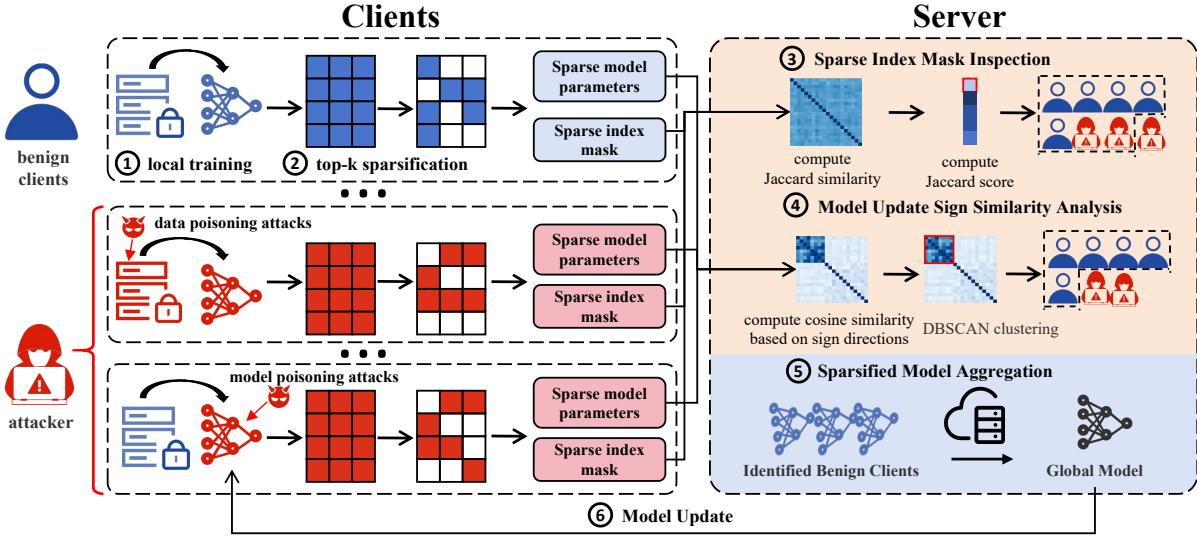


Fig. 1. An illustration of the FLARE training process, incorporating client-side top- k sparsification and server-side robust and sparsified aggregation.

B. Robustness Enhancement Design

To mitigate poisoning attacks in sparse communication scenarios, we propose two key components: *Sparse Index Mask Inspection* and *Model Update Sign Similarity Analysis*. These components are collaboratively designed to identify and filter out potential poisoning clients during the FL training process. The first component verifies whether the index mask of uploaded parameters adheres to expected patterns, ensuring the integrity of sparsified updates. The second component detects adversarial behaviors by analyzing the similarity of model update directions, identifying patterns indicative of potential attacks. We detail these components below.

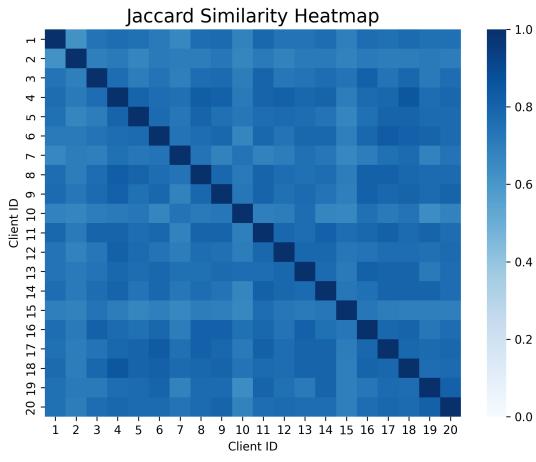


Fig. 2. The heatmap of the Jaccard similarity between the *sparse index masks* of different clients on the CIFAR-10 dataset under the non-IID setting.

1) *Sparse Index Mask Inspection*: Typical sparsified communication-efficient frameworks typically rely on *sparse index masks*. To further investigate their properties, we analyzed the structural characteristics of these masks and examined their consistency across clients. As shown in Fig. 2, our empirical analysis reveals that, even under non-IID training

settings, a significant degree of overlap exists among the masks generated by different clients. Leveraging this observation, we propose a filtering mechanism to eliminate potential adversarial clients whose sparse index masks exhibit low overlap with others or form isolated groups of similarity. Typically, clients with highly distinct sparse index masks introduce a significant degree of uncertainty in detecting poisoning attacks, thereby hindering the establishment of a robust defense. By filtering out these outliers, our approach enhances the reliability of attack detection and ensures a more consistent aggregation process.

Specifically, we compute the Jaccard similarity between the sparse index masks of different clients. Formally, for each client i , let M_i represent its sparse index mask, which contains the indices of the selected model parameters after top- k sparsification. The Jaccard similarity between two clients i and j is defined as:

$$J(M_i, M_j) = \frac{|M_i \cap M_j|}{|M_i \cup M_j|} \quad (1)$$

where $|M_i \cap M_j|$ represents the number of common selected indices between the two clients, and $|M_i \cup M_j|$ represents the total number of unique selected indices.

For each client, we compute its Jaccard score as the average Jaccard similarity with all other clients, defined as:

$$J_{\text{SCORE}}(i) = \frac{1}{m-1} \sum_{j \neq i} J(M_i, M_j) \quad (2)$$

where m is the total number of participating clients. The Jaccard score quantifies the similarity of a client's sparse index mask to those of the other clients in the system.

To identify anomalous clients, we define a threshold for filtering as follows:

$$J_{\text{THRESHOLD}} = \frac{J_{\text{MAX}} + J_{\text{MIN}}}{2} \cdot \beta \quad (3)$$

Here, J_{\max} and J_{\min} represent the highest and lowest Jaccard scores among all clients, respectively, and β is a manually tuned hyperparameter, with a default value of 0.6. Any client with a Jaccard score below this threshold is classified as an outlier and is excluded from the global model aggregation process.

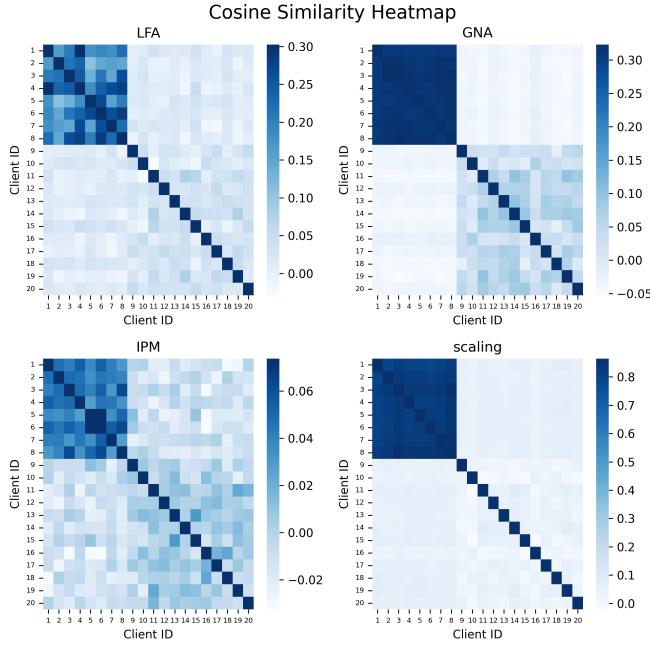


Fig. 3. The Heatmap of sign cosine similarity among clients on the CIFAR-10 dataset under the non-IID setting. The first 8 clients are adversaries, while the remaining 12 are benign clients.

2) *Model Update Sign Similarity Analysis:* Motivated by recent works [24], we also leverage similarity measurements among clients as a fundamental strategy to mitigate poisoning attacks. To evaluate the effectiveness of this approach, we simulate four types of poisoning attacks in a sparsified communication scenario and analyze the similarity of model update signs among clients.

As illustrated in Fig. 3, the first eight clients represent malicious adversaries, while the remaining twelve are benign clients. The results demonstrate that the sign similarity among malicious clients is significantly higher than that among benign clients in the sparsified communication scenario. Furthermore, the similarity between malicious and benign clients remains low, further emphasizing the distinct update patterns introduced by adversarial attacks. It is important to note that, due to the nature of sparsification, the sign similarity between any two clients can only be computed over the overlapping regions of their sparse index masks. In short, our observations reveal that the success of poisoning attacks typically relies on controlling multiple adversarial clients and injecting relatively consistent perturbations into the model updates.

Building upon these insights, we design a *Model Update Sign Similarity Analysis* module to cluster potential malicious clients for detection. The process begins by computing the model parameter differences between the sparse model param-

eters uploaded by each client and the global model from the previous round, as $\Delta w_i = w_i - w_G$. These differences are then converted into their corresponding sign directions, defined as:

$$\text{Sign}(\Delta w_i) = \begin{cases} 1, & \text{if } \Delta w_i > 0, \\ -1, & \text{if } \Delta w_i < 0, \end{cases} \quad (4)$$

where w_i represents the model parameters from client i , and w_G denotes the global model parameters from the previous aggregation round. It is important to note that parameters for which no client participated in the aggregation during the previous round are excluded from the similarity calculation. This exclusion arises because the server lacks the corresponding model parameters for these positions, as they are determined solely by the local models of the clients, which are independently set by each client.

Next, we compute the pairwise cosine similarity of model update signs between clients, considering only the overlapping portions of their sparse index masks. This restriction ensures that the comparison is performed on the same model elements between clients, thereby making the similarity measurement meaningful. The cosine similarity, $\cos(\theta_{i,j})$, is defined as follows:

$$\cos(\theta_{i,j}) = \frac{\langle \text{Sign}(\Delta w_i), \text{Sign}(\Delta w_j) \rangle}{\|\text{Sign}(\Delta w_i)\| \cdot \|\text{Sign}(\Delta w_j)\|}, \quad (5)$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, and $\|\cdot\|$ represents the Euclidean norm.

Subsequently, the cosine similarity is transformed into a distance metric using $1 - \cos(\theta_{i,j})$, which quantifies the distance between the updates of two clients. The resulting distance matrix is then utilized as input to a clustering algorithm. Specifically, we employ the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [25] within our framework. To determine the optimal value of ϵ , which defines the scan radius for core points in DBSCAN, we utilize the K-nearest neighbors (KNN) method. In this approach, ϵ is set as the average distance to the n -th nearest neighbor, where $n = N \cdot \gamma$. Here, N represents the total number of clients, and γ is a hyperparameter that controls the sensitivity of the clustering process. In practice, we set $\gamma = 0.2$ to balance sensitivity to adversarial clients and false positives. Additionally, the minimum number of points required to form a core point in DBSCAN is set to $N \cdot \gamma$, ensuring that each cluster contains a sufficient number of clients. This clustering process enables the effective grouping of clients based on similar attack patterns, while clients exhibiting significantly different update behaviors are identified as outliers. The resulting clusters facilitate the detection and exclusion of malicious clients from the global aggregation process. Consequently, only benign clients are included in the final global model aggregation, thereby enhancing the robustness of the framework.

3) *Algorithm Overview:* Algorithm 1 implements the poisoning mitigation mechanism of FLARE. The process begins with an analysis of the structural consistency of sparse index masks using the Jaccard similarity metric. For each given

Algorithm 1: Poisoning Client Filtering

Input: Sparse model $W = \{W_1, \dots, W_m\}$, sparse index masks $M = \{M_1, \dots, M_m\}$ of m clients C ; filtering threshold β and clustering sensitivity γ .

Output: Identified benign clients C_{BENIGN} .

```

1 function poison_filtering( $C, W, M, \beta, \gamma$ )
2   Initialize  $C_{\text{BENIGN}} \leftarrow C$  ;
3   foreach client  $i$  in  $C$  do
4     |  $J_{\text{SCORE}}(i) \leftarrow \frac{1}{m-1} \sum_{j \neq i} J(M_i, M_j)$ ;
5     Set filtering threshold  $J_{\text{THRESHOLD}} = \frac{J_{\text{MAX}} + J_{\text{MIN}}}{2} * \beta$ ;
6     foreach client  $i$  in  $C$  do
7       | if  $J_{\text{SCORE}}(i) < J_{\text{THRESHOLD}}$  then
8         | | Exclude client  $i$  from  $C_{\text{BENIGN}}$ 
9     foreach remaining client  $i$  in  $C_{\text{BENIGN}}$  do
10    | | Convert  $\Delta w_i$  to sign direction by Equation 4
11    Compute cosine similarity  $\cos(\theta_{i,j})$  by Equation 5;
12    Compute distance matrix  $D = 1 - \cos(\theta_{i,j})$ ;
13    Compute neighbor count  $n = m * \gamma$ ;
14    Set  $\epsilon$  as average distance to  $n$ -th nearest neighbor;
15    Apply DBSCAN clustering on  $D$  with  $n, \epsilon$ ;
16    Exclude grouped clients from  $C_{\text{BENIGN}}$ ;
17    return  $C_{\text{BENIGN}}$ 
```

client i , its Jaccard score, $J_{\text{SCORE}}(i)$, is computed using Equation 2. A threshold $J_{\text{THRESHOLD}}$, determined by Equation 3, is then applied to filter out clients with $J_{\text{SCORE}}(i) < J_{\text{THRESHOLD}}$, effectively removing outliers that exhibit abnormal sparsification patterns. For the remaining clients, parameter updates are transformed into binary sign directions via Equation 4, emphasizing update polarity rather than magnitude. Subsequently, the pairwise directional similarity $\cos(\theta_{i,j})$ is calculated using Equation 5. The resulting distance matrix, $1 - \cos(\theta_{i,j})$, is utilized as input to the DBSCAN clustering algorithm with adaptive parameters. Specifically, ϵ is set as the average distance to the n -th nearest neighbor, where $n = m \cdot \gamma$. Clients grouped into dense clusters by DBSCAN are identified as coordinated attackers and excluded from the set of benign clients.

C. Sparsified Communication-Efficient Robust Aggregation

Model sparsification has emerged as a promising technique for improving communication efficiency in FL by reducing the number of transmitted model parameters during training [5], [26]. In the traditional top- k sparsification setting, as proposed by Aji and Heafield [18], only the most significant model parameters are selected and transmitted at the scalar level, while the remaining parameters are set to zero. Inspired by the recent work of Yan et al. [12], FLARE employs a pack-level top- k sparsification approach. Unlike scalar-level sparsification, this method groups model parameters into small structured packs before applying the sparsification process. By operating at the pack level, this approach preserves the

structured information within each group, providing a more efficient and cohesive representation of the model updates.

In the sparsified, communication-efficient aggregation process of FLARE, each client first flattens its local model parameters and organizes them into structured groups for efficient packing and transmission. Based on the specified sparsification ratio and the characteristics of its local model, the client computes a sparse index mask to identify which parameter packs will be retained. Specifically, the client determines a sparsification threshold and assigns a mask value of 1 to packs whose aggregated values exceed this threshold. The final transmission to the server includes two components: the sparsified model parameters and a sparse index mask, represented with 1-bit per entry, indicating the retained packs. This structured sparsification approach effectively preserves critical information while significantly reducing communication overhead.

Algorithm 2: Sparsified Robust Model Aggregation

Input: Sparse model $W = \{W_1, \dots, W_m\}$, sparse index masks $M = \{M_1, \dots, M_m\}$, and local dataset sizes $\{|D_1|, \dots, |D_m|\}$ of m clients C ; filtering threshold β , clustering sensitivity γ .

Output: Aggregated model W_G .

```

1  $C_{\text{BENIGN}} \leftarrow \text{poisoning\_filtering}(C, W, M, \beta, \gamma)$  ;
2 foreach parameter pack  $P_i$  in global model do
3   |  $S_i \leftarrow \emptyset$  ;
4   | foreach client  $j$  in  $C_{\text{BENIGN}}$  do
5     | | if  $M_j(P_i) = 1$  then
6     | | | Add client  $j$  to  $S_i$  ;
7     | |  $A_{P_i} = \sum_{j \in S_i} \frac{|D_j|}{\sum_{k=1}^m |D_k|}$  ;
8     | Initialize aggregated model parameter  $w_i^{agg} = 0$  ;
9     | foreach client  $j \in S_i$  do
10    | |  $W_j(P_i) \leftarrow W_j(P_i) * \frac{|D_j|}{\sum_{k=1}^m |D_k|}$  ;
11    | | Normalize  $W_j(P_i) \leftarrow W_j(P_i) / A_{P_i}$  ;
12    | |  $w_i^{agg} += W_j(P_i)$  ;
13    | Set  $W_G(P_i) = w_i^{agg}$  ;
14 return  $W_G$ 
```

Algorithm 2 outlines the sparsified aggregation framework designed for communication-efficient FL. When the server receives sparsified model parameters $\{W_1, \dots, W_m\}$ and their sparse index masks $\{M_1, \dots, M_m\}$ from m clients, it first resolves sparse index misalignment by identifying contributing clients $S_i = \{j | M_j(P_i) = 1\}$ for each packet P_i . This alignment ensures structural consistency across clients' heterogeneous sparse representations.

The aggregation weight for client j is dynamically assigned as $\frac{|D_j|}{\sum_{k=1}^m |D_k|}$, proportional to its local dataset size $|D_j|$. For each P_i , the server calculates the total aggregation weight $A_{P_i} = \sum_{j \in S_i} \frac{|D_j|}{\sum_{k=1}^m |D_k|}$ to quantify cumulative contributions. Each model parameter $W_j(P_i)$ undergoes two sequential transformations: first scaled by its client-specific weight $W_j(P_i) \leftarrow W_j(P_i) * \frac{|D_j|}{\sum_{k=1}^m |D_k|}$ then normalized by A_{P_i} via

$W_j(P_i) \leftarrow W_j(P_i)/A_{P_i}$ to eliminate dimensional variance caused by differing contributor counts. The aggregated parameter $w_i^{agg} = \sum_{j \in S_i} W_j(P_i)$ is computed through weighted summation.

It is important to note that for model parameter packets with $S_i = \emptyset$ (unselected by any client), resulting in a value of 0 for those packs in the aggregation. To prevent gradient vanishing, clients subsequently populate these vacant P_i packs using their local parameters before the next training round. This ensures that the global model has non-zero values for all parameters, facilitating smooth gradient propagation and avoiding the problem of inactive parameters during backpropagation.

V. EXPERIMENTAL EVALUATIONS

A. Implementation and Default Setting

Our proposed FLARE¹ is implemented using PFLLib [27], an open-source federated learning (FL) framework.

In the following experiments, unless stated otherwise, the default FL training setup includes 20 clients, all participating in each training round and performing one local training epoch. The attacker ratio is set to 0.4, with poisoning attacks starting at the 10th round and continuing until training ends. The batch size is 64, and the Adam optimizer is used with a learning rate of 0.005.

Furthermore, we evaluate FLARE under two typical data partitioning settings in FL: IID and Non-IID. In the IID partitioning setting, the dataset is evenly distributed among clients, with each client receiving an equal proportion of samples from every class, ensuring a uniform data distribution. In contrast, the Non-IID partitioning setting simulates statistical heterogeneity by employing a Dirichlet distribution-based partitioning method [28]. Specifically, a Dirichlet distribution with a concentration parameter α is used to allocate data to clients, where α controls the degree of data heterogeneity. For our experiments, we set $\alpha = 1$ to achieve a moderate level of data heterogeneity among clients.

B. Datasets and Baselines

To evaluate the effectiveness and performance of FLARE, we conduct several experiments, consistent with the methodologies employed in existing related works. Specifically, we train a ResNet-20 model [29] using FLARE and compare it against related baselines on three public datasets: FashionMNIST [30], which comprises 70,000 grayscale images across 10 fashion categories; CIFAR-10 [31], which contains 60,000 color images from 10 object categories; and CIFAR-100 [31], which includes 60,000 color images distributed across 100 object categories. FLARE is compared with the following related baselines in this paper, as discussed in Section II:

- **FedAVG** [32]: A widely used aggregation method that computes a weighted average of model updates from multiple clients to generate a global model. The weights are based on each client's data size.

¹The code for FLARE will be made publicly available on GitHub upon the final decision of this paper.

- **Multi-KRUM** [11]: Calculates the Euclidean distance between each client update and all others. For each update, the nearest $m - n - 1$ distances are summed to compute a K_r score. The k updates with the smallest K_r scores form the set of honest updates (H), which is then aggregated using FedAVG.
- **Median** [10]: Independently calculates the median value for each model update across all client updates and uses it as the global model update.
- **Trimmed Mean** [10]: Sorts client updates and removes the largest and smallest $k\%$. The mean of the remaining values is used as the global model update. In our experiments, k is set to 10 by default.
- **FedSign** [24]: Identifies malicious clients by measuring cosine similarity between their signature vectors and calculating attack density. Clients with an attack density below average are considered honest, and their updates are aggregated using FedAVG.
- **Robust-DPFL** [33]: Computes a detection score for each client by averaging its model parameters. Detection scores are clustered into two groups using K-means, and only updates from the cluster with higher average detection scores are aggregated.

To evaluate FLARE's resilience to poisoning attacks, we conducted experiments on both IID and non-IID datasets under four common poisoning attack scenarios: Label Flip Attack (LFA) [19], Gaussian Noise Attack (GNA) [19], Inner Product Manipulation (IPM) [20], and Scaling Attack [21].

- **LFA**: The attacker changes the original label l of a training sample to $M - l - 1$, where M is the total number of labels (i.e., categories) in the dataset. The attacker then performs local training on this modified dataset.
- **GNA**: Using the mean and variance of local model, the attacker generates a Gaussian noise model matching the size of the local model and uploads it to the server.
- **IPM**: The attacker manipulates the inner product between true gradients and robust aggregated gradients to be negative, disrupting global model convergence.
- **Scaling**: The attacker amplifies local model updates by a large factor before uploading them to the server.

C. Poisoning Attacks in Sparsified FL

As illustrated in Fig. 4 and Fig. 5, we demonstrate the failure of most existing defense strategies against poisoning attacks in the context of sparsified communication-efficient FL. Different types of poisoning attacks commence at the 10th training epoch (marked by the red dashed line), and their impact on common sparsified aggregation is evident. Specifically, FedAvg fails to maintain stability, particularly under Scaling and Inner Product Manipulation (IPM) attacks, where the test accuracy drops drastically after the attack begins.

It is noteworthy that under adversarial perturbations introduced by various poisoning attacks, existing conventional and state-of-the-art (SOTA) defense mechanisms, such as Multi-KRUM, Median, and Trimmed Mean, exhibit only limited

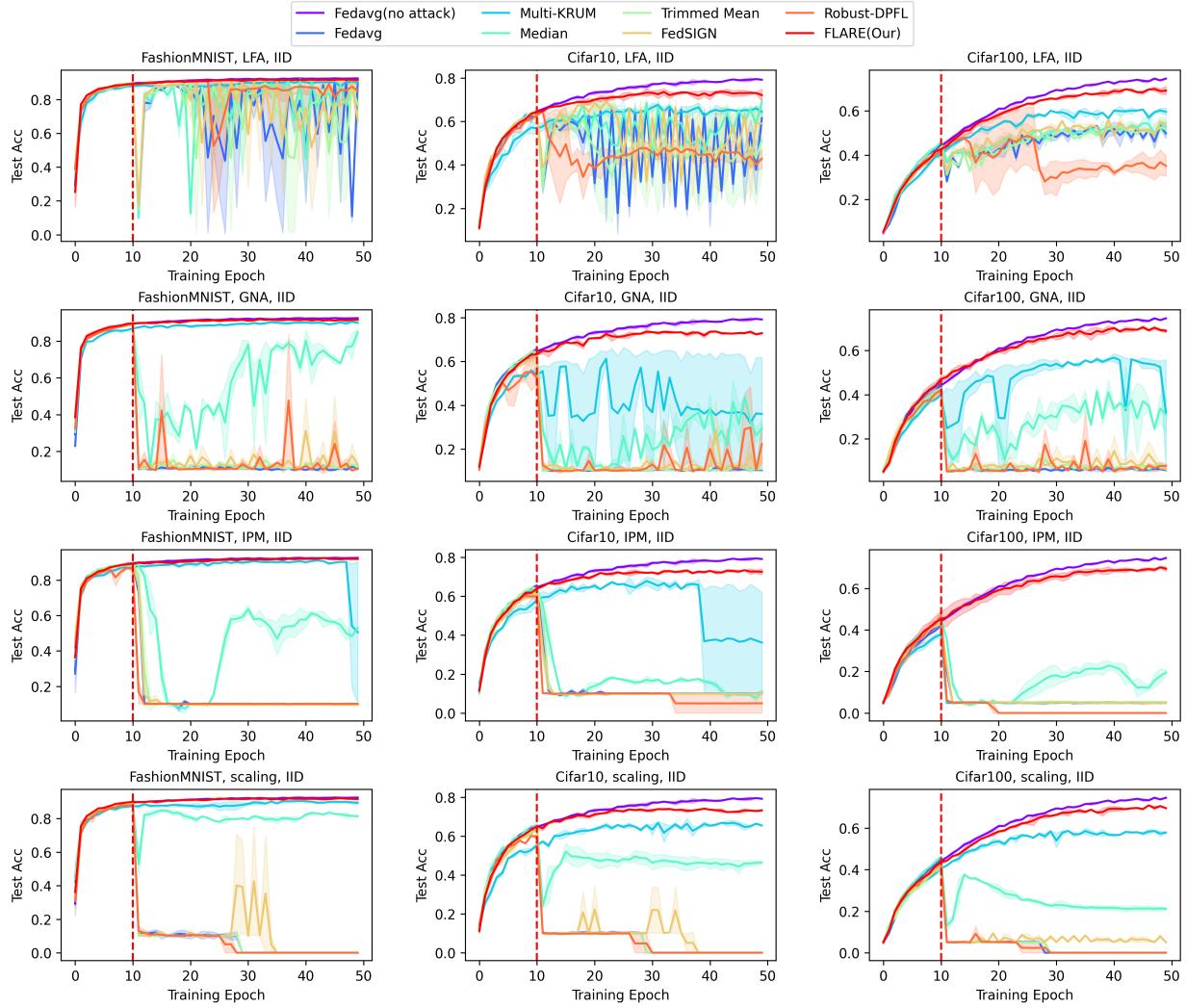


Fig. 4. Comparison of defense effectiveness across various approaches, evaluated on FashionMNIST, CIFAR-10, and CIFAR-100 under Label Flip Attack (LFA), Gaussian Noise Attack (GNA), Inner Product Manipulation (IPM), and Scaling Attack in an **IID SETTING**. The default attacker ratio is fixed at 0.4, and the top- k sparsification rate is set to 0.5. Top-1 accuracy is used as the evaluation metric for FashionMNIST and CIFAR-10, while top-5 accuracy is adopted for CIFAR-100.

resilience. Furthermore, their effectiveness is highly contingent upon the specific dataset and type of attack. For instance, while Multi-KRUM demonstrates robustness against scaling attacks on datasets such as FashionMNIST and CIFAR-10, it fails catastrophically under GNA attack and IPM attack on CIFAR-100. Similarly, Trimmed Mean is effective against LFA attack on CIFAR-10 but suffers substantial performance degradation under IPM and scaling attacks. Even recently proposed defense strategies, such as FedSIGN and Robust-DPFL, exhibit only partial robustness, struggling to withstand high-intensity attacks on more complex datasets like CIFAR-100. These findings underscore the limitations of existing defense mechanisms in addressing sophisticated adversarial strategies.

D. Model Performance and Defense Effectiveness of FLARE

To evaluate our proposed FLARE, we compare the model performance and effectiveness of FLARE with those of baseline methods. As shown in Fig. 4 and Fig. 5, FLARE maintains high test accuracy and stability even after the attack is introduced, outperforming other defenses in multiple adversarial settings. Specifically, Scaling and Inner Product Manipulation (IPM) attacks cause many baseline methods to collapse entirely, while FLARE remains robust, preserving high accuracy. These results indicate the effectiveness of FLARE against various poisoning attacks in adversarial federated learning environments.

E. Impact of Attacker Ratio in FLARE

To further investigate the impact of the attack ratio on FLARE, we evaluated the defense effectiveness of various attack strategies under different attack ratios (0.2, 0.3, and

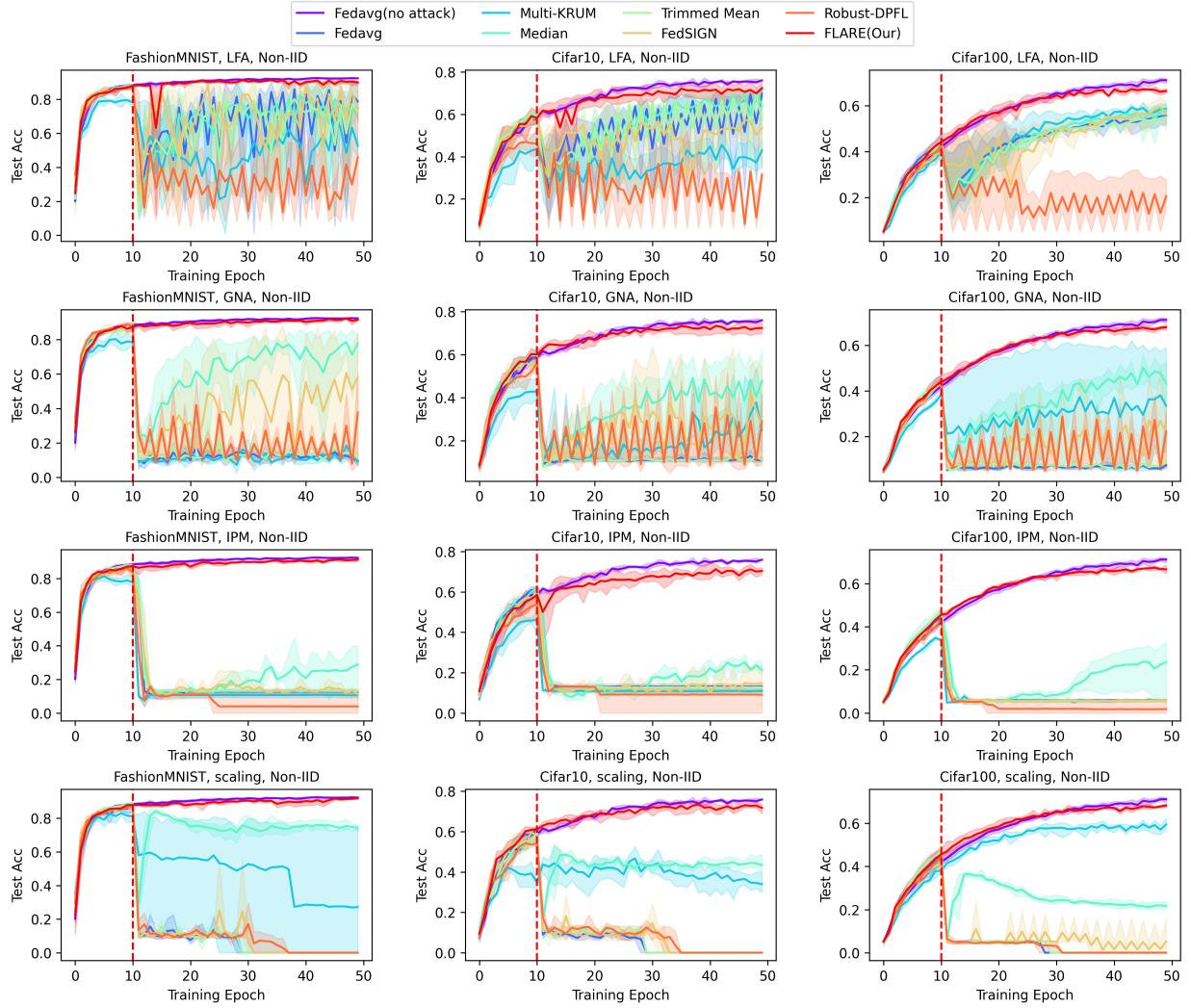


Fig. 5. Comparison of defense effectiveness across various approaches, evaluated on FashionMNIST, CIFAR-10, and CIFAR-100 under Label Flip Attack (LFA), Gaussian Noise Attack (GNA), Inner Product Manipulation (IPM), and Scaling Attack in a **NON-IID SETTING**. The attacker ratio is fixed at 0.4, and the top- k sparsification rate is set to 0.5. Top-1 accuracy is used as the evaluation metric for FashionMNIST and CIFAR-10, while top-5 accuracy is adopted for CIFAR-100.

0.4) on the CIFAR-100 dataset. It is important to note that, in our Threat Assumption, the attacker ratio will not exceed 50%. As shown in Figure 6, the results indicate that the defense methods are effective across all attack ratios, with no significant changes in performance. The test accuracy remains relatively stable regardless of the increase in attack ratio, demonstrating that our proposed defenses can effectively resist model poisoning attacks at various levels.

F. Impact of Top- k Sparsification Ratio in FLARE

To investigate how the top- k sparsification ratio affects FLARE under various attack strategies, we conducted experiments at different sparsification levels on the CIFAR-100 dataset. Specifically, the top- k sparsification ratio was set between 0.4 and 0.6 in our experiments. As shown in Figure 7, the final convergence results indicate that the attacker has no significant impact on our proposed FLARE across all

sparsification ratios. FLARE demonstrates strong resilience against four types of model poisoning attacks. However, lower sparsification ratios may lead to unstable attacker detection.

VI. CONCLUSION

In this work, we investigated the security vulnerabilities of federated learning in sparsification based communication-efficient scenarios under poisoning attacks and demonstrated the limitations of existing defense mechanisms in addressing these threats. To overcome these challenges, we proposed FLARE, a robust defense framework to effectively detect and mitigate adversarial clients. Through extensive experiments, we demonstrated that FLARE significantly enhances the robustness of FL in sparsification scenarios while preserving communication efficiency. These findings underscore the importance of developing security-aware sparsification strategies,

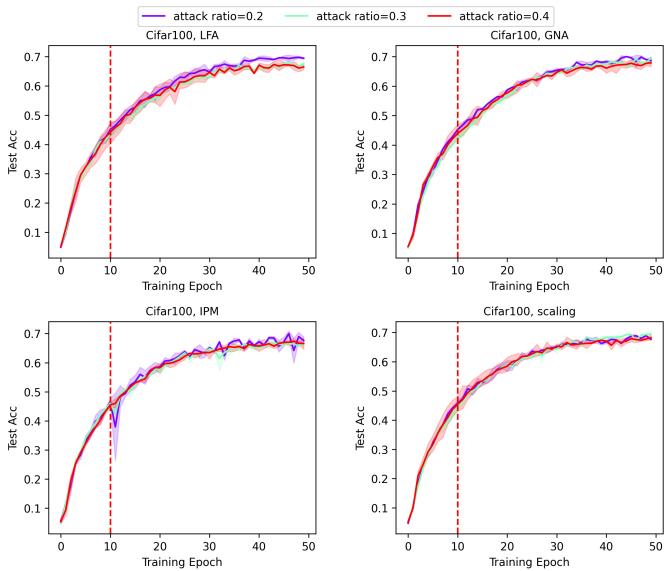


Fig. 6. Comparison of defense effectiveness across different attack ratios, evaluated on CIFAR-100 under LFA, GNA, IPM, and Scaling attacks. The top- k ratio is set to 0.5.

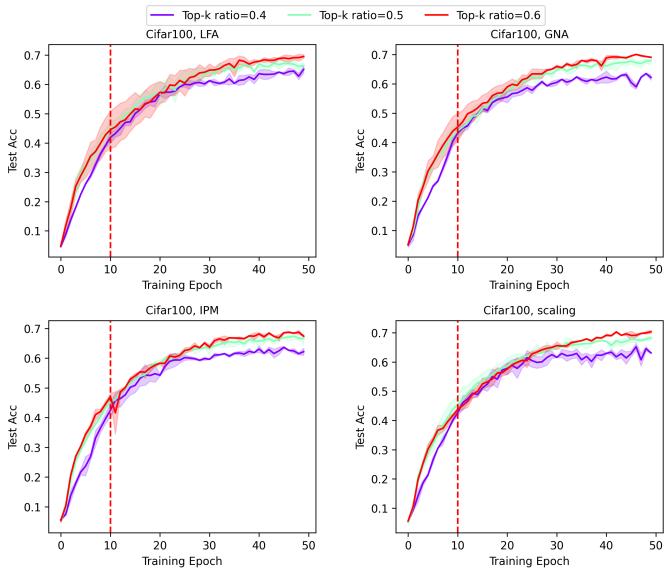


Fig. 7. Comparison of defense effectiveness across different top-k ratios, evaluated on CIFAR-100 under LFA, GNA, IPM attack, and scaling attack. The attacker ratio is set to 0.4.

providing a foundation for future research on strengthening the security of communication-efficient FL systems.

REFERENCES

- [1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Benni, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, “Advances and open problems in federated learning,” *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.
- [2] T. Haibo, L. Maonan, and R. Shuangyin, “Ese: Efficient security enhancement method for the secure aggregation protocol in federated learning,” *Chinese Journal of Electronics*, vol. 32, no. 3, pp. 542–555, 2023.
- [3] Q. Chen, A. Ye, Q. Zhang, and C. Huang, “A new edge perturbation mechanism for privacy-preserving data collection in iot,” *Chinese Journal of Electronics*, vol. 32, no. 3, pp. 603–612, 2023.
- [4] H. Li, T. Hu, J. Chen, X. Wu, and Q. Pei, “Privacy preserving algorithm for spectrum sensing in cognitive vehicle networks,” *Chinese Journal of Electronics*, vol. 33, no. 1, pp. 30–42, 2024.
- [5] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, “Robust and communication-efficient federated learning from non-iid data,” *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.
- [6] Y. Guan, X. Liu, T. Ren, and J. Niu, “Enabling communication-efficient federated learning via distributed compressed sensing,” in *IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.
- [7] X. Cao, T. Ouyang, K. Zhao, Y. Li, and X. Chen, “Efficient multi-task asynchronous federated learning in edge computing,” in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*. IEEE, 2024, pp. 1–10.
- [8] D. Alistarh, D. Grubisic, J. Li, R. Tomioka, and M. Vojnovic, “Qsgd: Communication-efficient sgd via gradient quantization and encoding,” *Advances in neural information processing systems*, vol. 30, 2017.
- [9] P. Han, S. Wang, and K. K. Leung, “Adaptive gradient sparsification for efficient federated learning: An online learning approach,” in *2020 IEEE 40th international conference on distributed computing systems (ICDCS)*. IEEE, 2020, pp. 300–310.
- [10] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *International conference on machine learning*. Pmlr, 2018, pp. 5650–5659.
- [11] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” *Advances in neural information processing systems*, vol. 30, 2017.
- [12] N. Yan, Y. Li, J. Chen, X. Wang, J. Hong, K. He, and W. Wang, “Efficient and straggler-resistant homomorphic encryption for heterogeneous federated learning,” in *IEEE Conference on Computer Communications*. IEEE, 2024, pp. 791–800.
- [13] H. Wang, L. Muñoz-González, M. Z. Hameed, D. Eklund, and S. Raza, “Sparsfa: Towards robust and communication-efficient peer-to-peer federated learning,” *Computers & Security*, vol. 129, p. 103182, 2023.
- [14] R. Höning, Y. Zhao, and R. Mullins, “Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 8852–8866.
- [15] Y. Zang, Z. Xue, S. Ou, L. Chu, J. Du, and Y. Long, “Efficient asynchronous federated learning with prospective momentum aggregation and fine-grained correction,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, 2024, pp. 16 642–16 650.
- [16] Z. Li, P. Chaturvedi, S. He, H. Chen, G. Singh, V. Kindratenko, E. A. Huerta, K. Kim, and R. Madduri, “Fedcompass: efficient cross-silo federated learning on heterogeneous client devices using a computing power aware scheduler,” *arXiv preprint arXiv:2309.14675*, 2023.
- [17] S. Shi, X. Chu, K. C. Cheung, and S. See, “Understanding top-k sparsification in distributed deep learning,” *arXiv preprint arXiv:1911.08772*, 2019.
- [18] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” *arXiv preprint arXiv:1704.05021*, 2017.
- [19] M. Fang, X. Cao, J. Jia, and N. Gong, “Local model poisoning attacks to {Byzantine-Robust} federated learning,” in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1605–1622.
- [20] C. Xie, O. Koyejo, and I. Gupta, “Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation,” in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 261–270.
- [21] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, “How to backdoor federated learning,” in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2938–2948.
- [22] I. Ergun, H. U. Sami, and B. Guler, “Sparsified secure aggregation for privacy-preserving federated learning,” *arXiv preprint arXiv:2112.12872*, 2021.
- [23] A. Panda, S. Mahloujifar, A. N. Bhagoji, S. Chakraborty, and P. Mittal, “Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 7587–7624.
- [24] Z. Guo, L. Xu, and L. Zhu, “Fedsign: A sign-based federated learning framework with privacy and robustness guarantees,” *Computers & Security*, vol. 135, p. 103474, 2023.

- [25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [26] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, “Sparse binary compression: Towards distributed deep learning with minimal communication,” in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.
- [27] J. Zhang, Y. Liu, Y. Hua, H. Wang, T. Song, Z. Xue, R. Ma, and J. Cao, “Pflib: Personalized federated learning algorithm library,” *arXiv preprint arXiv:2312.04992*, 2023.
- [28] T. Minka, “Estimating a dirichlet distribution,” 2000.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [30] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [31] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.
- [32] 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.
- [33] T. Qi, H. Wang, and Y. Huang, “Towards the robustness of differentially private federated learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 18, 2024, pp. 19911–19919.