Defending Against Data Poisoning Attack in Federated Learning With Non-IID Data

Chunyong Yin and Qingkui Zeng

Abstract—**Federated learning (FL) is an emerging paradigm that allows participants to collaboratively train deep learning tasks while protecting the privacy of their local data. However, the absence of central server control in distributed environments exposes a vulnerability to data poisoning attacks, where adversaries manipulate the behavior of compromised clients by poisoning local data. In particular, data poisoning attacks against FL can have a drastic impact when the participant's local data is non-independent and identically distributed (non-IID). Most existing defense strategies have demonstrated promising results in mitigating FL poisoning attacks, however, fail to maintain their effectiveness with non-IID data. In this work, we propose an effective defense framework, FL data augmentation (FLDA), which defends against data poisoning attacks through local data mixup on the clients. In addition, to mitigate the non-IID effect by exploiting the limited local data, we propose a gradient detection strategy to reduce the proportion of malicious clients and raise benign clients. Experimental results on datasets show that FLDA can effectively reduce the poisoning success rate and improve the global model training accuracy under poisoning attacks for non-IID data. Furthermore, FLDA can increase the FL accuracy by more than 12% after detecting malicious clients.**

Index Terms—**Data augmentation, data poisoning, federated learning (FL), gradient detection.**

Nomenclature

Notations	Description
ω	FL model parameter.
ω^*	FL optimal model parameter.
ω_0	Initial model parameter.
ω_t	Model parameter at epoch t.
ω_t^i	Client i'S model parameter at epoch t.
T	FL model training epoch.
N	Number of clients.
c	Fraction of selected clients.
m	Number of random selected clients.
p_j	Proportion of data participating in FL.
D	FL local training dataset.
U_i	Client i.
D_i	Client i'S local data.
p_s	Proportion of training samples poisoned.
α	Beta distribution parameter.
λ	Beta distribution value.
Γ	non-IID degree.
ζ	Attack degree.
η	Learning rate.
Δ_i	Poisoning content on the ith sample.
(x_i, y_i)	ith true training sample.
y_i^f	False target class of ith training sample.
$(\tilde{x}, \tilde{y})$	Feature-target vector.

Manuscript received 16 March 2023; revised 24 June 2023; accepted 13 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61772282 and in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX23_1357. *(Corresponding author: Chunyong Yin.)*

The authors are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China (e-mail: yinchunyong@hotmail.com).

Digital Object Identifier 10.1109/TCSS.2023.3296885

I. Introduction

WITH the wide application of the Internet of Things (IoT), applications such as intelligent healthcare and intelligent transportation are flourishing [1]. The data generated by terminal devices are growing exponentially, and these data provide a solid foundation for building an informative world. For example, in the Internet of Medical Things (IoMT), health data of users collected and stored by end devices are jointly trained to an efficient medical deep learning model [2]. However, receiving data from terminal devices to centralized servers may pose threats to users' privacy [3]. To maintain the efficiency of big data handling and protect the privacy of clients, federated learning (FL) [4] has been proposed to mitigate the performance bottlenecks and privacy risks associated with centralized computing. On the one hand, FL aggregates federated global models based on model gradient updates trained locally by the client rather than on raw data held by the user. On the other hand, FL has exposed some privacy and security issues due to its distributed nature.

The private aspects of local training by clients make FL inherently vulnerable to poisoning attacks from clients [5]. Such attacks manipulate an arbitrary percentage of malicious users to disrupt the training performance of the global model by modifying local data labels or model parameters [6]. The attackers launch targeted attacks to cause a specified skew in the training model [7] or launch untargeted attacks to significantly degrade the training results or even make the model infeasible [8]. As shown in Fig. 1, some malicious users manipulate clients' raw data and upload gradients of abnormal models to the server [9]; others modify model parameters during local training or upload incorrect gradient updates to the server [10]. In this article, we focus on the data poisoning attacks. Both poisoning attacks have a severely impact on FL,

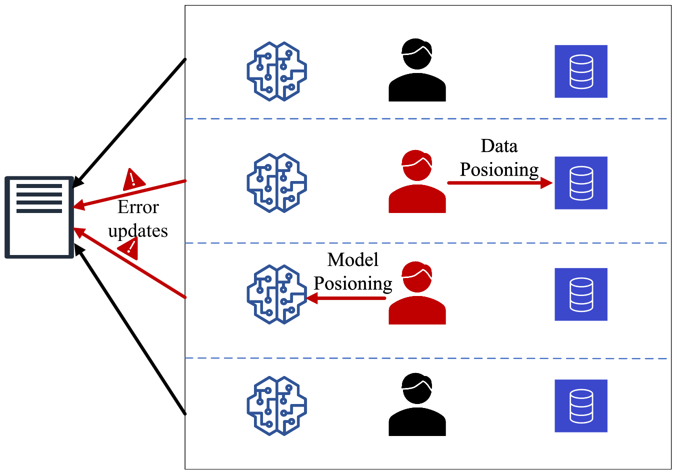

Fig. 1. FL poisoning attack.

especially for data poisoning attacks that are often worse when the customer data is non-independent and identically distributed (non-IID) [11].

In centralized computing, common defenses against poisoning attacks such as data preprocessing [12] and robustness training [13] approaches require access to local training data. In the distributed FL setting, on the other hand, the server only has access to local gradients and these defenses cannot be executed. To defend against data poisoning attacks in FL, several defenses have been explored in existing research, for example, Byzantine robust aggregation [14], [15], [16], cluster-based detection [17], and other approaches. However, Byzantine aggregation works poorly or even invalidly when the number of malicious users exceeds that of benign users; when the client is highly non-IID data locally, the clustering-based defense approach has difficulty distinguishing between malicious and benign users for gradient updates. To summarize, these works have the following limitations.

1) Many defense strategies assume an upper limit on the number of Byzantine users, which is not realistic in poisoning attack scenarios.
2) Some approaches address specific attack defenses, while malicious users have the ability to launch both targeted and untargeted attacks.
3) The existing work has poor defense performance when the FL client's local dataset is non-IID. Therefore, our goal is to bridge this gap, specifically, we expect to successfully defend against targeted or untargeted data poisoning attacks in FL with the non-IID data.

To address the above problems, we propose the FL data augmentation (FLDA) defense framework, which defends against data poisoning attacks in FL with the client's local data being non-IID. Unlike other existing works, the FLDA does not have a rigorous limit on the number of malicious clients. In a reality FL poisoning attack scenario, it is not possible to effectively hypothesize the number of benign users or to avoid the case that the user data are non-IID. Data augmentation facilitates the state-of-the-art performance in a variety of tasks [18]. In defending against poisoning attacks, data augmentation strategies are typically lighter and more effective than other strategies, while not significantly degrading training accuracy [19]. More, existing FL poisoning attack defense strategies for specific attack instances, and can not be applied to a variety of data poisoning attack ways. Although data augmentation has promising performance in defending against poisoning attacks and mitigating non-IID data issues, the free behavior of local clients in FL does not ensure the execution of local data mixup.

Our work contributes to the feasibility of implementing local data augmentation in FL. Specifically, we introduce an incentive mechanism to motivate participants to actively engage in data augmentation techniques. Furthermore, we develop an update clustering algorithm on the server side to detect whether clients have executed data augmentation and whether any malicious activity is involved. By incorporating these advancements, our proposed defense framework, FLDA, proves effective in mitigating the damage caused by various types of poisoning attacks, including targeted attacks such as label flipping and untargeted attacks such as additive noise.

In this article, we first confirm the data poisoning attack will cause more significant damage in the FL with non-IID, which cannot be mitigated by existing defense schemes effectively. Second, we propose the FLDA defense framework to mitigate model accuracy degradation by implementing the mixup [20] scheme in a lightweight capacity on the clients to make the local training model robust. Next, to utilize benign user data and reduce the frequency of malicious user participation, we design a malicious user weight detection scheme to reduce the number of suspicious malicious user participation during server aggregation. Finally, experiments on datasets validate that FLDA can mitigate the impact of data poisoning attacks without significantly degrading the model's accuracy. The main contributions are summarized as follows.

1) We explore how data poisoning attacks against FL can have a worse impact on the global model when the client's local dataset is non-IID data.
2) We propose the FLDA defense framework to mitigate the impact of poisoning attacks without significantly degrading the model training performance.
3) We introduce a gradient-based malicious user detection strategy to increase the participation rate of benign clients and exploit the limited availability of local non-IID data.
4) Through comprehensive experiments, we demonstrate the superior defense performance of FLDA compared to the state-of-the-art approaches in non-IID scenarios.

The rest of this article is organized as follows: Section II is the related work about FL and poisoning attacks; Section III introduces the basic knowledge of this article; Section IV introduces the proposed defend framework; Section V conducts the experiment and analyzes the results; Section VI presents the conclusion.

II. Realted Works

A. Poisoning Attack in FL

In any machine learning or deep learning circumstance, the training phase may suffer from poisoning attacks whenever the curatorship is not in the hands of the learner. The poisoner aims to subvert the performance of the training model by interfering with the training phase. In general, poisoning attacks include two methods: model poisoning and data poisoning.

In the model poisoning attack, the poisoner interacts directly with the model access learning algorithm or modeling process, and the learner cannot control model modification. The model poisoning attacks are launched during the training phase of the model by arbitrarily tampering with some parameters of the local model [21]. The ith parameter of the local model of the poisoned client is modified using random noise [22]. The Krum [23] attacker makes the server choose some elaborate local model as an adaptive attack. The attack exploits the vulnerability of Krum and similar aggregation rules by trying to construct a poisoned model that is closest to other local models but obtains the reverse direction to correct the global gradient. The trim attack [14] is designed for median aggregation and Trim-mean aggregation. It poisons the local model by falsifying values much larger than the maximum or smaller than the minimum benign local model parameters. Based on these values, the global model is shifted in the opposite direction in subsequent training.

In the data poisoning attack, the poisoner can affect the training results by manipulating a small portion of the training data used by the learning algorithm. This happens when the learner employs datasets that are not sanitized or come from an unknown source. The case where the poisoner manipulates the training data labels and a small part of the data content is called dirty-label data poisoning [24]. The case where the poisoner cannot manipulate the training data label and can only modify part of the data content is called clean-label data poisoning. A backdoor attack [25] is a targeted data poisoning attack in which the attacker attacks one or more participants. In FL, most learning models that suffer from backdoor attacks do not significantly impact the main task. However, these Byzantine attackers force the model to make incorrect decisions about data with specific features. Likewise, the label-flipping attack [26] modifies the labels of the training samples to a specified class. However, the features of the data remain unchanged, which can lead to misclassification of the attacked classifier.

B. Defend Strategies in FL

In FL, the detrimental effects of poisoning attacks are not immediate, necessitating the consideration of their subsequent impact on the system [27]. Therefore, defense strategies should aim to prevent attacks in advance and effectively mitigate their consequences. Consequently, the study of defense mechanisms can be divided into two categories: passive defense strategies implemented before an attack and active defense strategies employed in response to an attack. Passive defense focuses on pre-emptively sanitizing the training dataset and detecting anomalies to identify poisoned data before global training communication, which represents a widely adopted solution [28]. Furthermore, active defense strategies have gained prominence as a means to mitigate persistent damage resulting from poisoning attacks [29]. These strategies go beyond early detection and aim to actively counteract the impact of attacks, thereby safeguarding the integrity of the FL system. Active defense mechanisms are crucial in effectively handling the aftermath of poisoning attacks and reducing their long-term consequences. By combining both passive and active defense strategies, FL systems can proactively prevent attacks and effectively mitigate their impact, thus ensuring the reliability and robustness of the FL process.

Data poisoning attacks can be defended against data contamination by examining the raw training data for data sanitization, enhancement, and transformation through data pre-processing defense strategies. Cao et al. [30] demonstrate that poisoning attacks against a limited number of malicious clients are secure by proposing an integrated FL framework, FLCert. The key idea is to group clients into groups, learn a global model for each group of clients using any existing FL method, and perform majority voting in the global model to classify the test inputs. There are also methods to detect malicious users based on the anomalies of poisoning. Cao et al. [31] propose a FedRecover method that recovers an accurate global model from a poisoning attack with minimal computational and communication costs for the client. The key idea is that the server estimates the client's model updates rather than requiring the client to compute and communicate them during the recovery process. Saad et al. [32] designed a novel framework to automatically detect malicious actors in the FL process, utilizing deep reinforcement learning to dynamically select a participant as a trusted entity, and the trusted participant will be responsible for identifying poisoning model updates using unsupervised machine learning. Jiang et al. [33] proposed MCDFL to defend against label flipping attacks by generating a network to detect the quality of training data for each client. It can identify malicious clients by recovering the distribution on the potential feature space to detect the data quality of each client.

The defense methods for model poisoning attacks detect anomalies in the victim model parameters and eventually correct their normal function. Cao and Gong [34] proposed a fake client-based model poisoning attack called MPAF. The main idea is to assume that the attacker injects fake clients into the FL system and sends elaborate fake local model updates to the cloud server during the training process, such that the learned global model has low accuracy for many indiscriminate test inputs. The robust training strategy makes the model more robust in the training phase and increases the fault tolerance of the poisoned data, thus making it more difficult for subsequent poisoning attacks to succeed. Zhang et al. [35] constructed a secure FL-based network intrusion detection system, i.e., secFedNIDS. the main idea is to identify poisoned models and reject them to participate in the global intrusion detection model by an online unsupervised poisoned model detection method. And they proposed a classpath similarity-based poisoned data detection method to filter out poisoned traffic data and avoid their participation in the subsequent local training. Shi et al. [36] proposed a federated anomaly analysis (FAA-DL) framework to defend against local model poisoning attacks in distributed machine learning. In this framework, a lightweight anomaly detection method is used to isolate potential malicious local models and a privacy-preserving anomaly verification method based on function encryption is used to verify the anomalies with high accuracy. Then FAA-DL accurately eliminates the verified anomalies and aggregates the remaining normal local models to generate a global model. Kumari et al. [37] introduce a novel and more general backdoor defense framework, BayBFed, which uses the probability distribution of client-side updates to detect malicious updates in FL and uses a novel detection algorithm to efficiently detect and filter malicious updates using this probability measure.

Further advancements in leveraging blockchain technology can significantly enhance the resilience of FL systems against poisoning attacks. By integrating blockchain with the FL framework, the detrimental impact of poisoning attacks can be effectively mitigated through the utilization of decentralization and consensus mechanisms. Al Mallah and López [38] proposed a defense mechanism against poisoning attacks that leverages blockchain technology to monitor and detect malicious clients in parallel. The approach filters out unreliable workers, and miners in the decentralized FL environment randomly select a subset of reliable workers to continue the FL process. This methodology improves the overall security, scalability, and time efficiency of the FL system. In a similar vein, Kalapaaking et al. [39] proposed a privacy-preserving verification method to identify and eliminate poisoned local models in FL scenarios. Through the utilization of secure multiparty computation (SMPC)-based cryptographic inference processes, the privacy of local model parameters is effectively preserved while simultaneously detecting and eliminating attacked local models. Upon successful verification, the authenticated shares of the local model are transmitted to the blockchain for the aggregation process. The resulting global model is securely stored in tamper-proof storage, and participants retrieve the global model from the blockchain for verification of its authenticity.

III. Preliminaries

In this section, we introduce the concepts of data poisoning attacks in FL and data augmentation defense strategies. The details are shown in the following. Nomenclature summarizes the notation used in this article.

A. Federated Learning

FL is a type of distributed machine learning that does not rely entirely on the central curator. Google [40] proposes the concept of FL to leverage data from personal mobile devices for model training while protecting user privacy. In the general FL paradigm, there are multiple clients and one server, but the central server is not necessary for blockchain-based FL learning [41].

On the clients, the chosen participants download the global model and initial parameters first. Then, each device iterates its model and computes gradients by local training datasets. Next, the model updates are uploaded to the server. Last, the clients received the return global model parameter updates from the server and applied the updates to the local model parameters.

Algorithm 1 FL Paradigm

```
Input: communication rounds $T$; number of clients $n$; selected clients fraction $c$; number of local epochs $E$; local batch size $B$; each client dataset $D_i$; learning rate $\eta$;
Output: global model parameter $\omega$
1:  function LocalUpdate($i$, $\omega_t$)
2:      for each local epoch $E$ do
3:          for each batch $b$ in $B$ do
4:              $g_{t+1}^i = \nabla l(w_t; b)$;
5:              $\omega_t^i \leftarrow \omega_t^i - \eta g_{t+1}$;
6:      clip_grad_norm_($g_{t+1}^i$);                    ▷ Clip gradient
7:      Return $g_{t+1}^i$;
8:  Initial global model $\omega_0$;
9:  for each round $t$ in $T$ do
10:     $m \leftarrow \max(c \cdot n, 1)$;
11:     for each client $i$ in $n$ do
12:         LocalUpdate($i$, $\omega_t$);
13:     $\omega_{t+1} \leftarrow \omega_t - \eta \sum_{i=1}^{m} \frac{|D_i|}{|D|} g_{t+1}^i$;     ▷ Aggregate
14: Return model parameter $\omega_T$;
```

On the server, a random fraction C of clients is selected at the beginning of each round, and the server sends the current global model state to these chosen clients. Then the parameters of the locally trained model are centralized, and the server distributes an aggregated average model return to these clients. This communication process is repeated until the global model reaches the desired state. The general FL framework has n participants $U_1, U_2, \ldots, U_n$, each hold training datasets $D_1, D_2, \ldots, D_n$. The minimize objective function can be expressed as

$$\arg\min_{\omega} \sum_{i=1}^{m} p_i F_i(\omega_i) \tag{1}$$

where $\omega \in \mathbb{R}^d$ is the parameter updates after aggregated, m is the number of selected clients, $F_i(\omega_t^i)$ is the local objective function for the ith client, and p_i specifies the proportion of the ith client local samples $|D_i|$ in the total samples. When the clients completed the tth round training, they uploaded their gradient to the server. Then the $t+1$th round global model parameters update as

$$\omega_{t+1} \leftarrow \omega_t - \eta \sum_{i=1}^{m} p_i \nabla F_i\left(\omega_t^i\right). \tag{2}$$

FedAvg [40] is a classical FL aggregation algorithm that trains the global model by stochastic gradient descent, and the overall training paradigm is given in Algorithm 1.

Because of the distributed training property of FL, the aggregated server has no access to the client's local data and can only get gradient updates for local model training. This allows data poisoning attacks to flourish.

B. Poisoning Attack

In the poisoning attack scenario, an attacker can modify the p_S proportion of the total training samples for poisoning.

Suppose there are $|D|$ training samples and the attacker adds a perturbation Δ_i to modify the ith sample $S_i(x_i, y_i)$ to find the optimal perturbation such that the target $p_s \cdot |D|$ samples are misclassified or incorrect. To obtain the false label y_i^f, the minimization loss function is

$$\min_{\Delta \in c} \sum_{i=1}^{p_s \cdot |D|} l\left(F\left(x_i, \theta(\Delta_i), y_i^f\right)\right) \tag{3}$$

where $l(\cdot)$ is the loss of poisoning attacks, $F(x, \theta)$ is the objective function of training samples $x \in \mathbb{R}^|D|$ with the parameters $\theta \in \mathbb{R}^{p_s \cdot |D|}$, and the $\theta(\Delta_i)$ needs to satisfy

$$\theta(\Delta_i) \in \arg\min_{\theta} \frac{1}{N} \sum_{1}^{N} l(F(x_i + \Delta_i, \theta), y_i). \tag{4}$$

The attacker launches the poisoning attack with different intentions and in different ways. Whatever the method by which the poisoner launched the poisoning attack, their attack goals include targeted attacks and untargeted attacks while avoiding a significant decrease in global accuracy. The poisoner launches targeted attacks to cause a specified skew in the training model. For example, by modifying the true labels of the training data to cause the training model to make class-specific misclassifications without significantly affecting the training results. Alternatively, the untargeted attacks are not carefully designed to skew the model in a specified way. Their genuine purpose is only to corrupt the training phase if possible, causing the training model to be inaccurate or even unusable.

1) The goal of targeted attacks is to decrease the performance of a model in a specific case without affecting other cases. The label-flipping attack is the main method of targeted data poisoning attacks. The attacker changes the source label of training samples y_i, to the label of another target class y_i^f, while keeping the other classes' integrity to avoid detection. However, the features of the training samples keep the same, leading to the misclassification of the attacked model classifier. Furthermore, the attacker may flip multiple source labels to a specific target label. This results in the target label's classification accuracy improving while causing serious damage to the accuracy of other classes.
2) The goal of the untargeted attack is to decrease the overall performance of the global model. The attacker adds Gaussian or random noise to its local gradient to change the true gradient value or flip the sign of its local gradient positive or negative while keeping the magnitude of the gradient constant. The attacker expects to decrease the test accuracy of the model across all classes without the model being subjected to a specified skew, and this type of poisoning is more easily detected.

C. Mixup Principle

Mixup is a popular data augmentation technique that generates additional data by linear interpolation between actual data instances and has commonly been applied to image classification tasks and has been shown to improve the testing

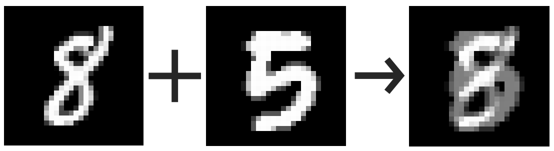

Fig. 2. Mixup data augmentation sample.

accuracy on various datasets. Mixup is a generic vicinal distribution as

$$\mu(\tilde{x}, \tilde{y}|x_i, y_i) = \frac{1}{n} \sum_{j}^{n} \mathbb{E}_\lambda[\delta(\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \\ \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j)] \tag{5}$$

where λ follows the Beta(α, α) distribution, for $\alpha \in (0, \infty)$. It means that sampling from the mixup vicinal distribution produces virtual feature-target vectors. The hyperparameter α controls the interpolation strength between the feature-target pairs. Where $(\tilde{x}, \tilde{y})$ and (x_i, y_i) are two feature-target vectors randomly selected from the training data, and $\delta(\cdot)$ is the Dirac mass function

$$\begin{aligned} \tilde{x} &= \lambda \cdot x_i + (1 - \lambda) \cdot x_j \\ \tilde{y} &= \lambda \cdot y_i + (1 - \lambda) \cdot y_j. \end{aligned} \tag{6}$$

Mixup regularizes class boundaries and removes small nonconvex regions by assigning convex combinations of training samples to convex combinations of labels. As shown in Fig. 2, when α is set to 0.5, mixup generates the data enhancement samples as shown on the right side of the figure, whose data label is generated according to the distribution of $0.5y_i + 0.5y_j$.

Zhang et al. [20] demonstrated that the mixup models have superior robustness to adversarial samples. Inspired by this, we use mixup to facilitate the removal of poisoned regions in the client's local space, where the poisoned data instances are assigned adversarial labels, and the space is (nonpoisoned) instances with different labels. Our work appropriately modifies mixup under FL local client restrictions to make it defend against data poisoning attacks and mitigate performance degradation under FL of non-IID data.

IV. FL Data Augmentation

In this section, we introduce a robust defense framework called FLDA, which effectively mitigates data poisoning attacks in FL by employing local data mixup on the clients. We begin by discussing the threat model associated with poisoning attacks in FL settings and outline our defense strategy. Subsequently, we present the comprehensive framework of FLDA, highlighting its key components and interactions. We then delve into the development of algorithms that are specifically designed for the client and server sides of the FLDA framework. Finally, we provide a theoretical analysis of FLDA's defensive performance, assessing its efficacy in countering poisoning attacks.

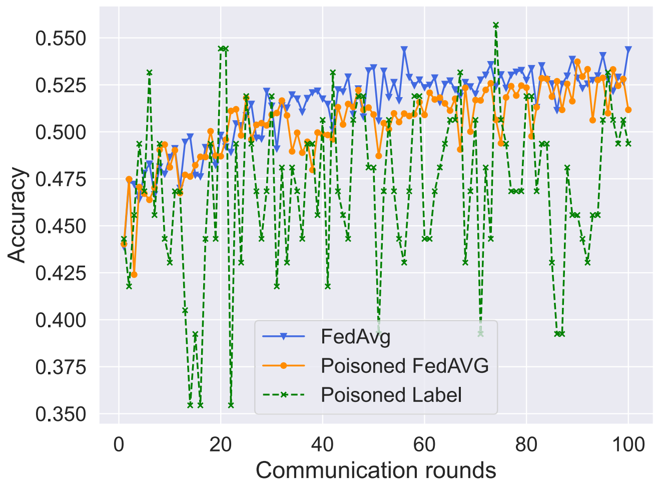

Fig. 3. Impact of data poisoning attacks on test accuracy.

A. Threat Model and Defense Strategy

The adversary launches the data poisoning attack by faking to be malicious clients among the FL participants. This threat model has a dramatic impact on the training results of FL regardless of whether the attack is targeted or not. Among them, label-flipping attacks that bring no significant changes to the global model accuracy are difficult to detect, which results in skewed target label prediction results. We simply simulated the scenario under label-flipping attack on the CIFAR-10 dataset with the CNN model, as shown in Fig. 3, this type of poisoning attack can silently cause a target class to be misclassified. The global model of poisoned FedAVG does not significantly degrade the accuracy but makes the attacked labels continue to be severely affected in subsequent training.

The local training of benign and malicious clients currently changes when the poisoning attack occurs. Recall that the normal FL paradigm, where the benign clients i then upload the t round update after local training as

$$\omega_{t+1}^{i} \leftarrow \omega_{t}^{i} - \eta \nabla F_i\left(\omega_t^i, D_i\right). \tag{7}$$

However, when the local client is the malicious one, the update ω_{t+1} is altered to

$$\omega_{t+1}^{i} \leftarrow \omega_{t}^{i} - \eta\left((1-\zeta)\nabla F_i\left(\omega_t^i, D_i\right) + \zeta \nabla F_i\left(\omega_t^i, \Delta_i\right)\right) \tag{8}$$

which ζ is the adversary's attack degree. The parameter server directly aggregates these malicious client insecure updates for global model updates. The data poisoning attack is then successfully launched. Therefore, in this work, our defense strategy should have the following goals.

1) Mitigating the impact of malicious clients on global model aggregation.
2) It cannot disturb the normal participation of benign clients in global training.
3) Since the clients hold non-IID data, valuable data from all participants should be utilized in every possible way.

We consider designing defense strategies for both local clients and the server side to achieve these goals. The local data augmentation algorithm is designed to train the local model by the mixed local datasets during local training on the client's side. The server then utilizes the unique resource "gradient updates" to detect clients' contributions and redesign the aggregation weights to mitigate the impact of malicious clients on the global model.

B. Overall Framework

To address the threat of data poisoning attacks against FL with non-IID, we propose the federated learning data augmentation (FLDA) defense framework. The design motivation for FLDA stems from the idea of how to implement FL local data augmentation. The impact of poisoned data from malicious clients on global aggregation will be mitigated by assuring the execution of FLDA through the incentive mechanism and update clustering.

The data augmentation process is carried out first for all participants of FL. The augmented data are employed for local training regardless of whether the client is benign or not. On the one hand, considering the non-IID problem of federated training, the client's local data mixup algorithm can improve the data quality and alleviate the non-IID degree. On the other hand, direct client-side local data augmentation can avoid the poisoned data of malicious clients from polluting other local models, while mitigating the model skewing caused by the modification of specified labels by targeted attacks.

During the FL training process, the local data of clients remains strictly confined to the client side and is solely uploaded to the global aggregator for updates. This means that the global aggregator, situated on the server side, only has access to the gradient history of the selected clients during each communication round. Given that the local data of malicious clients has been deliberately poisoned, noticeable disparities arise between the updates generated by these malicious clients and the updates from benign clients throughout the training process. By analyzing these gradient histories, we can effectively identify anomalies in the model updates performed by the clients. To facilitate this anomaly detection, we establish a global participant gradient storage on the server side, which stores the updated information received from the clients. This accumulated information enables us to cluster the clients into two distinct categories: benign and poisoned. Importantly, to avoid raising suspicion among the malicious clients, the server continues to select all clients for participation in the federated training as per the normal selection frequency. However, we can strategically diminish the model weights assigned to the poisoned clients, thereby minimizing their influence on the global model aggregation. The entire flow of the proposed framework is depicted in Fig. 4, providing a comprehensive overview of the processes involved in detecting and mitigating the impact of poisoning attacks in FL.

C. Incentive Mechanism

Several recent research efforts have confirmed the effectiveness of data augmentation in mitigating poisoning attacks. However, the FL paradigm only requires local clients to upload updates of local models and cannot enforce additional operations. To be specific, the clients will not actively carry

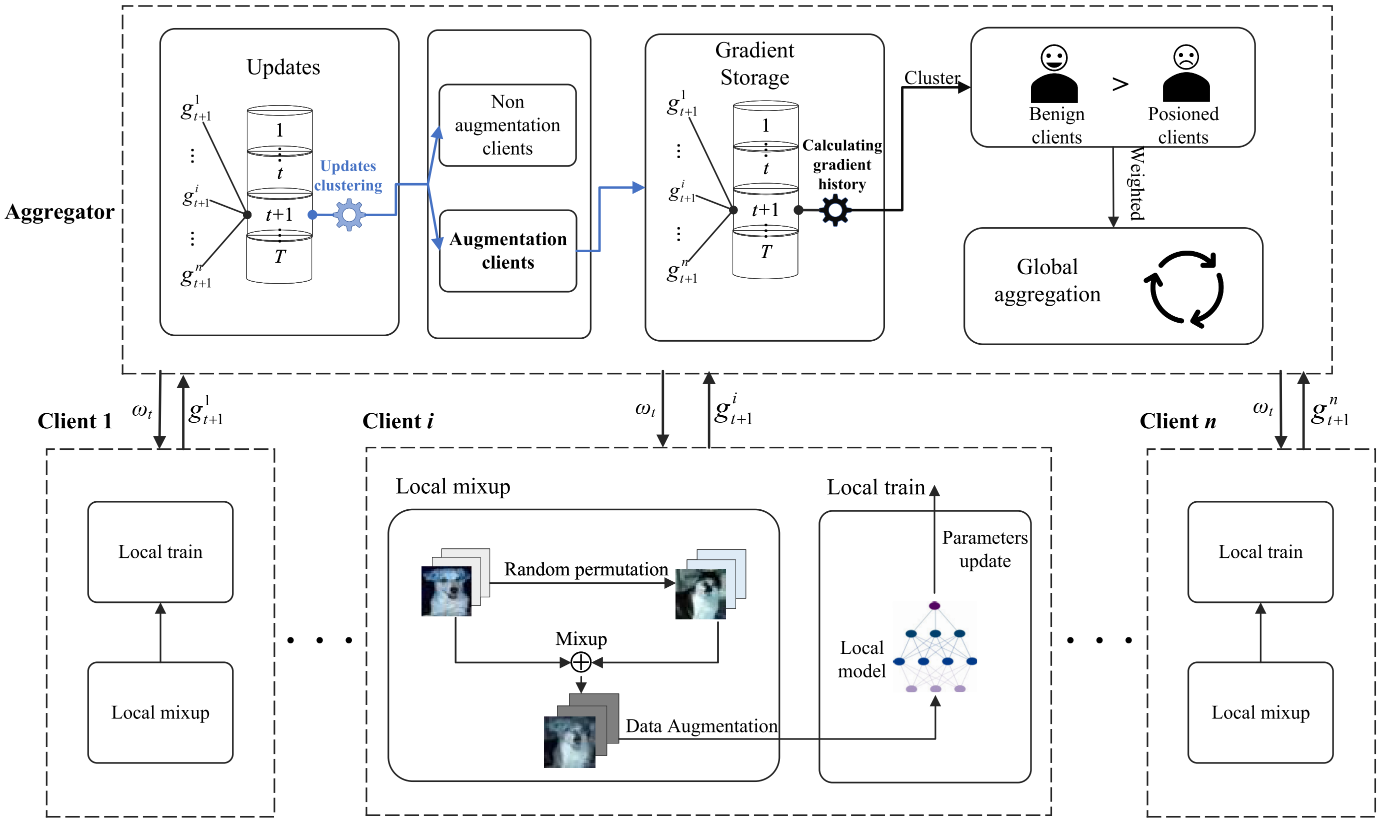

Fig. 4. Overall framework.

out the extra consumed actions on the local devices, especially for malicious users. Inspired by the paper [42], it is necessary for more participants to actively contribute their valuable data resources in global training. Therefore, to ensure that the client can actively execute data augmentation, we need to provide a payment to the client by setting a corresponding incentive mechanism.

The incentive mechanism we design needs to contain the following elements: payment allocation, client selection, and weight measurement. First, the key to the incentive mechanism is how to determine the distribution of payment, giving different levels of payment according to the contribution of the client. For example, benign clients who do not execute data augmentation, malicious clients who execute data augmentation, and benign clients who execute data augmentation, etc. Next, we need to select the clients that will participate in the global aggregation. To ensure that the defense is effective, clients that do not execute data augmentation under the current round should be pruned. There may be malicious clients among the remaining clients, but these need to be retained to ensure that local data augmentation mitigates the non-IID issue. Therein lies the need for weight measurement. Finally, we will discuss how to measure the weight of client aggregation in Section IV-E. The final contribution is determined based on the weight of each client in the global aggregation.

D. Local Data Augmentation at Clients

At the terminal of the FL paradigm, the non-IID data issue often leads to poor training results of the client's local model. In this scenario, the adversary then launches the data poisoning attack by faking benign clients or injecting poisoned data through the backdoor to further disrupt the federated training results. However, due to the privacy-preserving features of FL, the server or curator does not have direct access to the raw data of the client. This makes it difficult to identify malicious clients at the user level.

The utilization of data augmentation approaches has been recognized as an effective strategy for mitigating the non-IID data problem in FL. In our pursuit of the defense objectives outlined in Section IV-A, we leverage data augmentation techniques to address both the non-IID and data poisoning challenges. In this article, we demonstrate that the client-side data augmentation solution based on data mixup can mitigate the threat of poisoning attacks. Before the FL training starts, we deploy both **local mixup** and **local train** modules on the client side.

In the local mixup module, we employ the local mixup algorithm only applied within the client's local data. For the client's local data, we pre-separate it into two blocks. One is the local data in normal order, and the other is randomly permuted from the local data. Next, the algorithm generates a new data augmentation set using the proportion of λ value obtained from the Beta distribution. This new dataset has new training samples and labels which will be used for the federated training process, while the raw local dataset samples will not take part in the model training. In the local train module, we use the data augmentation set for the training of the local model. Specifically, to avoid contingency errors under

Algorithm 2 Local Data Augmentation

Input: local dataset samples X and labels Y; Beta distribution parameter α_1, α_2; local batch dataset D_B and size B; learning rate η;

Output: local mixup dataset D_i

```
1:  function LOCALMIXUP($X$, $\alpha_1, \alpha_2$)
2:      $\lambda = Beta(\alpha_1, \alpha_2)$;
3:      $X' = randperm(X)$;
4:      $Y' = randperm(Y)$;
5:      $X_m = \lambda \dot{X}' + (1-\lambda)\dot{X}$;
6:      Return $X_m$, $Y'$
7:
8:  function LOCALTRAIN($X_m$, $Y'$, $\omega_t$)
9:      $Y_p = \omega(X_m)$;
10:     $loss = \lambda l(Y, Y_p) + (1-\lambda) l(Y', Y_p)$;
11:     $g_i = \omega_{t+1} - \omega_t$;
12:     Return $g_i$;
13:
14: Download global model $\omega_t$;
15: $X_m, Y' =$ LOCALMIXUP($X, Y, \alpha_1, \alpha_2$);
16: $g_i =$ LOCALTRAIN($X_m, Y, Y', \omega_t$);
17: Return gradient update $g_i$;
```

the augmented data, the module will regenerate a new dataset after the start of each global training round. First, the module takes the augmented data samples as local model inputs to obtain training results. Next, the loss values of the training results for the original and augmented labels are calculated based on the proportion of λ. Finally, the training of the local model is completed to get the updated gradient of the local model.

In the local data augmentation phase, the raw data is avoided to be directly involved in the training of the local model as shown in Algorithm 2. The augmented data is used in each round of local training to reduce the impact of malicious clients on the global model. In lines 1–6, local mixup random permute the local dataset, in which randperm($\cdot$) is the permutation function. In this module, we get the mixed training samples X_m and the random permute labels Y'. In lines 8–12, the Local train module trains the local model by augmentation dataset to get the predicted result Y_p. Then, the loss value of the predicted results for the true and random permute labels is calculated, where $l(\cdot)$ is the loss function. Finally, the local gradient update is obtained by subtracting the global model of this round from the model weights trained with the augmented data.

E. Updates Clustering at Server

Ensuring the availability of the FLDA framework places a particular emphasis on the server's responsibilities. The server assumes two critical tasks: pruning clients that do not execute data augmentation before the regular global aggregation and detecting malicious clients participating in the aggregation process. As depicted in Fig. 4, the server initiates the process by first identifying and excluding client updates that do not involve data augmentation, thereby excluding them from the current round of aggregation. Subsequently, the server employs a detection mechanism to identify malicious clients among the remaining client updates.

To facilitate update pruning, we have designed an update clustering method that leverages general k-means clustering to determine data augmentation clusters. The specific steps of this approach are outlined below.

1) Calculate the mean value of these updates.
2) The update with the closest l_2 distance from the mean value is selected as the initial clustering center c_1, and the update with the greatest l_2 distance from the mean value is selected as the initial clustering center c_2.
3) Calculate the distance of each update from these two clustering centers and allocate it to the closest cluster.
4) For both clusters, calculate the mean value in the cluster and use the new one as the cluster center.
5) Repeat steps 3–4 until the clustering centers have not changed.
6) The updates intra these two clusters are aggregated into two sub-global models, respectively. Measure the accuracy of the sub-global models on the test dataset and prune out the updates of the sub-global model with low accuracy.

At the center of the FL paradigm, the non-IID data issue often leads to poor training results of the global model aggregation. Although we have incorporated local model training with data augmentation at the client level, the subsequent introduction of poisoning by malicious clients, coupled with the non-IID nature of the data, continues to impact the aggregation process. The parameter aggregation server is responsible for collecting gradient update information from participating clients in each training round. However, directly identifying malicious clients based on these historical gradients proves to be a complex task, primarily due to the influence of local client data augmentation. Therefore, we propose an approach based on historical gradient clustering to classify the participants into benign and malicious clients in a fuzzy manner.

Considering the defense goals under the non-IID issue, we should utilize the model updates from the participants even if they are malicious clients. Therefore, we thought of modifying the global model aggregation weights to accept gradient updates from the participants normally without making active choices. In this case, the more serious consequences of an adversary discovering that a malicious client has been identified and launching a new attack can be avoided. The impact of data poisoning attacks on model aggregation is further reduced by reducing the weight of malicious clients participating in aggregation while not being detected by the adversary.

On the parameter server side, we decompose the global aggregation task into three distinct components: gradient storage, benign client detection, and gradient aggregation. To achieve these objectives, we propose the employment of a gradient history detection algorithm. First, during each communication round of the federated training, the server accumulates and stores the gradient updates transmitted by the selected participants. Subsequently, the server calculates the difference, denoted as h_i, between the gradient

Algorithm 3 Gradient History Detection Algorithm

Input: client gradient updates g_i; communication rounds T; number of chosen clients m; learning rate η

Output: model parameter ω_T

```
1:  for epoch $t$ in $T$ do
2:      $G_t, H_t = [], []$;
3:      for client $i$ in $m$ do
4:          $G_t.append(g_i)$;
5:          $h_i = |g_i - mean(G_{t-1})|$;
6:          $H_t.append(h_i)$;
7:      $KC = k - Means((H_t, 2)$;
8:      for $i, c$ in $KC$ do
9:          if $i$ in $c_1$ then
10:             $p_i = (|c_1|/(|c_1 + c_2|)/m$;
11:         else
12:             $p_i = (|c_2|/(|c_1 + c_2|)/m$;
13:     $\omega_{t+1} \leftarrow \omega_t - \eta \sum_{i=1}^{m} p_i \cdot g_i$;
14: Return model parameter $\omega_T$;
```

value g_i contributed by each client and the average value of historical gradients from the previous round, preserving these differences. Based on these computed differences, the clients are classified into two categories: benign clients and malicious clients. In practical scenarios, benign clients tend to significantly outnumber malicious ones. Finally, the global gradient aggregation is performed by assigning weights to each participating user, proportionate to the ratio of clients in the two clusters. The detailed process is illustrated in Algorithm 3. Specifically, lines 3–6 involve the storage of gradient history differences and the current gradient values on the server side. The k-means algorithm is applied with a cluster count of two in line 7 to classify the gradient history differences. Subsequently, in lines 8–13, the weights for gradient aggregation are determined based on the respective proportions of clients belonging to each category. Finally, the global model is updated for the subsequent round of training. Through the utilization of this proposed algorithm, our framework successfully addresses the crucial tasks of gradient storage, benign client detection, and gradient aggregation on the parameter server side. This approach ensures the integrity and effectiveness of the FLDA defense mechanism, bolstering the robustness of the FL system against data poisoning attacks.

F. Theoretical Analysis

We derive the convergence guarantee similar to FedAvg using the proposed FLDA framework. We follow the notation in the algorithm describing FedAvg, with the difference being the weight setting of the server aggregation phase. Before giving our theoretical results, we first made the following assumptions 1–5 for each client the functions $F_1, \ldots, F_N$.

Assumption 1: $F_1, \ldots, F_N$ are L-smooth: for all v and w, $F_k(v) \leq F_k(w) + (v - w)^T \nabla F_k(w) + (L/2)||v - w||_2^2$.

Assumption 2: $F_1, \ldots, F_N$ are μ-strongly convex: for all v and w, $F_k(v) \geq F_k(w) + (v - w)^T \nabla F_k(w) + (\mu/2)||v - w||_2^2$.

Assumption 3: Let ξ_t^k be sampled from the kth device's local data uniformly at random. The variance of stochastic gradients σ_k^2 in each device is bounded: $\mathbb{E}||\nabla F_k(w_t^k; \xi_t^k) - \nabla F_k(w_t^k)||^2 \leq \sigma_k^2$ for $k = 1, \ldots, N$.

Assumption 4: The expected squared norm of stochastic gradients G is uniformly bounded, i.e., $\mathbb{E}||F_k(w_t^k; \xi_t^k)||^2 \leq G^2$ for all $k = 1, \ldots, N$ and $t = 1, \ldots, T - 1$.

Assumption 5: Assume M contains a subset of K indices randomly selected with replacement according to the sampling probabilities $p1, \ldots, pn$.

We define F^* and F_k^* be the minimum values of F and F_k, respectively. We use the term $\Gamma = F^* - \sum_{k=1}^{n} p_k F_k^*$ for quantifying the degree of non-IID. When then Γ obviously goes to zero as the number of samples grows, the data are IID. Then, we have the following convergence guarantee on our proposed FLDA framework.

Theorem 1: Let Assumptions 1 to 5 hold and L, μ, σ_k, G be defined therein. Choose $\kappa = (L/\mu)$, $\gamma = \max\{8\kappa, E\}$. Then FLDA satisfies

$$\mathbb{E}[F_k(w_t)] - F^* \leq \frac{\kappa}{\gamma + T - 1}\left(\frac{2(B + C)}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}||w_1 - w^*||^2\right) \tag{9}$$

where

$$B = \sum_{k=1}^{N} p_k^2 \sigma_k^2 + 6L\gamma + 8(E - 1)^2 G^2$$
$$C = \frac{4}{K} E^2 G^2. \tag{10}$$

Proof: The server establishes M by IID with replacement sampling an index K with replacement according to the sampling probabilities $p_1, \ldots, p_N$. Bounding the expected distance between the perturbed gradients with our defense and raw gradients, the expected difference between $\overline{V}_{t+1}$ and $\overline{W}_{t+1}$ is bounded by

$$\mathbb{E}_m||\overline{v}_{t+1} - \overline{w}_{t+1}||^2 \leq \frac{4}{K}\eta_t^2 E^2 G^2. \tag{11}$$

Then

$$\begin{aligned}\mathbb{E}||\overline{w}_{t+1} - w^*||^2 &= \mathbb{E}_m||\overline{w}_{t+1} - \overline{v}_{t+1}||^2 + \mathbb{E}_m||\overline{v}_{t+1} - w^2||^2 \\ &\leq (1 - \eta_t\mu)\mathbb{E}_m||\overline{w}_t - w^2||^2 + \eta_t^2(B + C).\end{aligned} \tag{12}$$

Then by the strong convexity of $F(\cdot)$, We have

$$\mathbb{E}[F(\overline{w}_t)] - F^* \leq \frac{L}{2}\Delta_t \leq \frac{L}{2}\frac{v}{\gamma + t}. \tag{13}$$

When we set $\kappa = (L/\mu)$, $\gamma = \max\{8\kappa, E\} - 1$, then let $\eta_t = (2/\mu)(1/(\gamma + t))$ to obtain the proof. □

V. Experimental Evaluation

A. Experimental Setup

1) Datasets: In the experimental evaluation, our proposed framework is validated on different datasets using commonly used experimental datasets for FL, including EMNIST [43], Fashion-MNIST [44], and CIFAR-10 [45]. EMNIST is an extension of the handwritten dataset MNIST, consisting of

TABLE I
DATASETS

Dataset	Training samples	Test Samples	Classes
EMNIST	731,668	82,587	62
Fashion-MNIST	60,000	10,000	10
CIFAR-10	50,000	10,000	10

TABLE II
ACCURACY(%) UNDER UNTARGETED ATTACK

	FedAvg	Mud-Hog	Foolsgold	vanilla FLDA	FLDA
EMNIST	74.77	81.87	80.25	78.48	83.56
Fashion-MNIST	87.54	93.48	96.93	94.06	97.97
CIFAR-10	60.32	67.61	67.53	62.15	67.92

gray-scale handwritten digits and letters, and we choose the "byclass" split as the experimental dataset. It contains 731 668 training samples and 82 587 test samples from 62 categories of upper and lower case letters. Fashion-MNIST is a set of clothing product images in 28 × 28 grayscale, whose size, format, and training/test set partition is exactly the same as the original MNIST. CIFAR-10 is a color image classification dataset consisting of a predefined set of 50 000 training examples and 10 000 test examples. Each example belongs to one of these ten classes including cats, deer, dogs, frogs, airplanes, cars, horses, ships, birds, and trucks. The datasets used in our experiments are shown specifically in Table I.

The experiments were implemented in Python 3.8 with Pytorch 1.9 and were performed on a PC with Ubuntu 20.04.5 LTS, AMD CPU R9-5950x, RTX 3080Ti GPU, and 64 GB RAM. To make the experimental results more intuitive and uniform, we choose the label-flipping attack as the attack method of data poisoning throughout the evaluation. And we use the simple two-layer CNN network model for EMNSIT dataset training and ResNet18 as the network model for CIFAR-10.

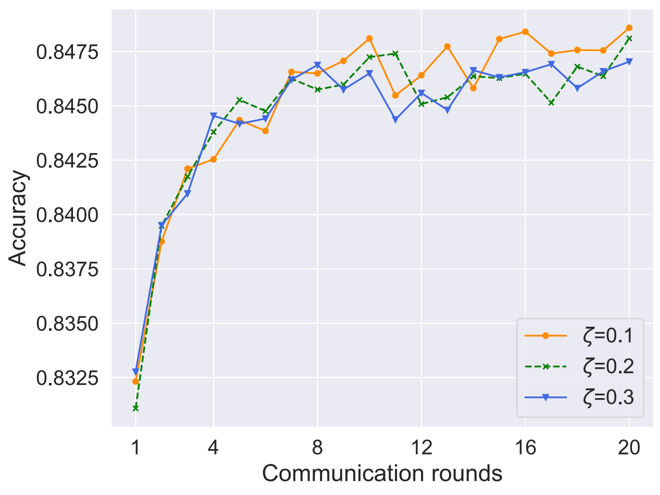

Fig. 5. Accuracy of attack degrees on EMNIST.

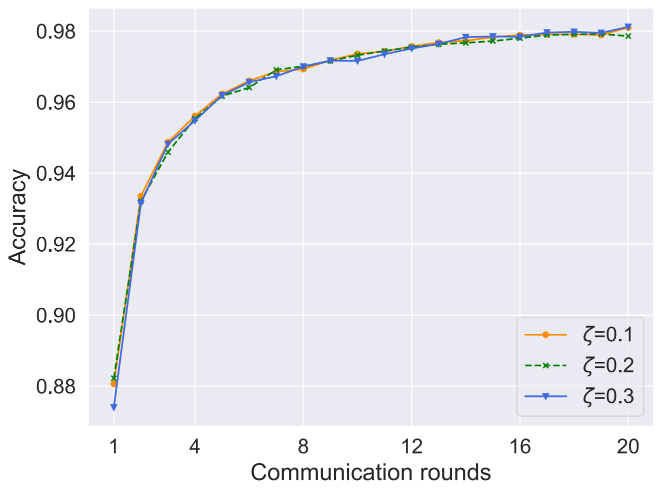

Fig. 6. Accuracy of attack degrees on Fashion-MNIST.

B. Experimental Results

We conducted an evaluation of the defensive efficacy of the FLDA framework against untargeted attacks. In our experimental setup, we specified the following parameters: the number of participating clients (N), the proportion of selected clients (c), the number of local training rounds (E), the learning rate (η), the local batch size (B), the non-IID degree (Γ), and the Beta distribution parameter (α). Specifically, we set these parameters to 50, 0.1, 10, 0.01, 64, 0.3, and 1.0, respectively. To assess the defense capabilities of the proposed framework, we conducted validation experiments using a fixed attack degree (ζ) of 0.3. The results, presented in Table II, demonstrate the performance of FLDA in comparison to other methods. The analysis reveals that FLDA exhibits commendable performance, especially when considering the EMNIST dataset, indicating its effectiveness as a defense mechanism against untargeted attacks.

We proceeded to assess the defensive capability of the FLDA framework against targeted attacks, specifically label flipping attacks, at various attack levels. To conduct this evaluation, we fixed the values of several parameters: the number of participating clients (N), the fraction of chosen clients (c), the number of local training rounds (E), the learning rate (η), the local batch size (B), the degree of non-IID (Γ), and the Beta distribution parameter (α). We set these parameters to 50, 0.1, 10, 0.01, 64, 0.3, and 1.0, respectively. In this experiment, the selection of the number of local training rounds (E) was determined by the fraction of chosen clients (c). For example, we set $E = 10$ when $c = 0.1$ and $E = 5$ when $c = 0.2$. We established three sets of the FLDA framework, each corresponding to different attack degrees (ζ), to demonstrate the defense performance. The classification accuracy on the EMNIST, Fashion-MNIST, and CIFAR-10 datasets at attack degrees of 0.1, 0.2, and 0.3 is illustrated in Figs. 5, 6, and 7, respectively. These figures showcase the effectiveness of the FLDA framework in mitigating the impact of label flipping attacks across different datasets and attack degrees.

As can be seen from the figures, the proposed FLDA framework can defend against data poisoning attacks at different degrees. The training results of FL are not more severely affected with the FLDA defenses. However, it is inevitable

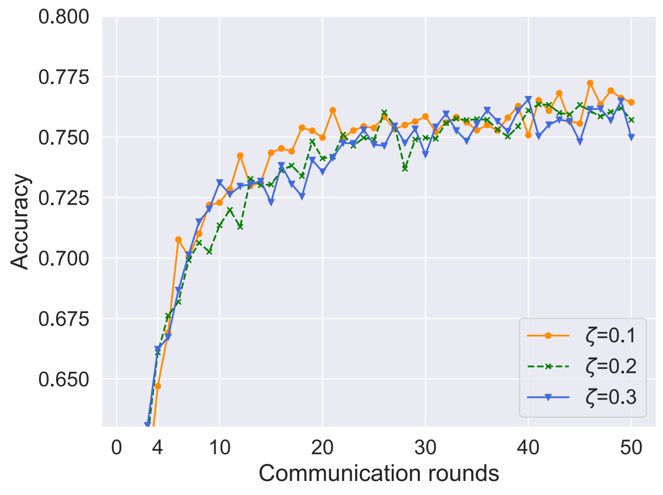

Fig. 7. Accuracy of attack degrees on CIFAR-10.

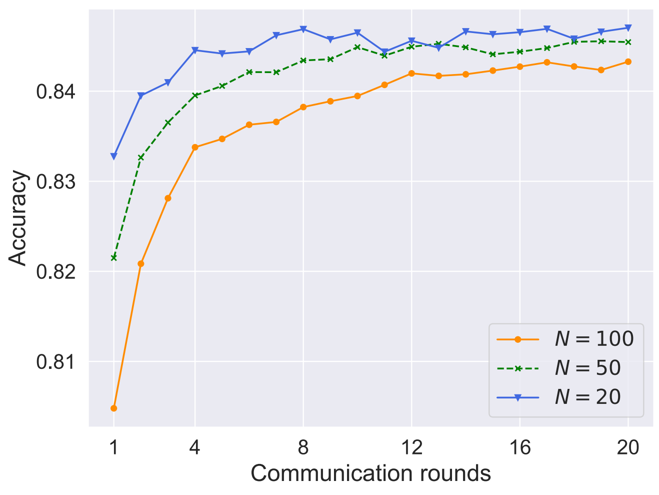

Fig. 8. Accuracy of the number of clients on EMNIST.

TABLE III
ACCURACY(%) OF NON-IID DEGREES ON CIFAR-10

Γ	0	0.1	0.3	0.5	1.0
FedAvg	66.72	67.70	64.74	58.38	40.34
Mud-Hog	75.75	73.65	68.26	63.37	58.52
Foolsgold	70.43	68.92	65.73	60.93	54.81
vanilla FLDA	73.78	72.41	70.85	65.04	53.72
FLDA	76.46	74.25	70.95	66.57	60.33

TABLE IV
ACCURACY(%) OF NON-IID DEGREES ON EMNIST

Γ	0	0.1	0.3	0.5	1.0
FedAvg	80.72	77.61	72.45	65.77	54.64
Mud-Hog	86.33	85.87	81.56	76.38	70.27
Foolsgold	80.63	76.92	73.88	68.31	62.84
vanilla FLDA	82.81	81.56	79.24	75.25	70.10
FLDA	85.57	87.27	81.68	78.12	67.95

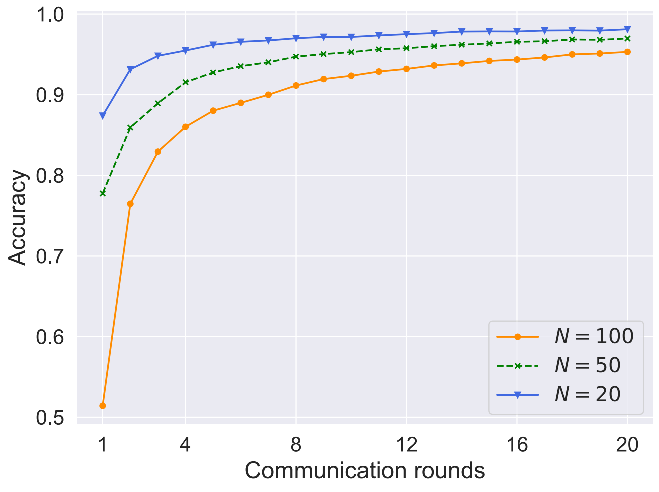

Fig. 9. Accuracy of the number of clients on Fashion-MNIST.

that the federated model accuracy decreases slightly when the attack degree increases.

We next evaluation whether our proposed solutions can support the defense performance to be sustained under different degrees of non-IID Γ. There are some other FL poisoning attack defense solutions that have been proposed to solve the non-IID issue. We have chosen the classical FedAvg [40] as the benchmark method and the Mud-HOG [46] and Foolsgold [47] methods as the comparison methods. Further, we delete the gradient history detection process in the FLDA framework and use only the vanilla aggregation scheme, which is named vanilla FLDA. We set the number of participating clients N, the fraction of chosen clients c, the number of local training rounds E, the learning rate η, the local batch size B, the attack degree ζ, and the Beta distribution parameter α to 50, 0.1, 10, 0.01, 32, 0.2, and 1.0, respectively. As Tables III and IV show the classification accuracy (%) on CIFAR-10 and EMNIST datasets at 0, 0.1, 0.3, 0.5, and 1.0 non-IID degrees.

The experimental results presented in the tables highlight the effectiveness of the proposed FLDA framework in defending against data poisoning attacks across different attack degrees. It can be observed that FedAvg did not experience severe accuracy degradation, as the label-flipping attack had a limited impact on the overall training results. In contrast, the FLDA framework demonstrated its capability to effectively address the non-IID issue in FL, outperforming other existing solutions. Notably, both vanilla FLDA and FLDA showed negligible performance degradation even in the more extreme cases where Γ was set to 0.5 or 1.0. Furthermore, it is worth noting that the accuracy of the global model remained stable and approached convergence, as specified in Theorem 1, after the designated number of training rounds. This finding emphasizes the efficacy of the proposed defense solution with data augmentation, as it not only mitigates the non-IID issue but also provides robust defense against poisoning attacks.

Furthermore, we conducted an assessment of parameter effects in alternative FL settings. The influence of non-IID data and the utility of data augmentation are also contingent upon the size of data held by local clients. To evaluate the impact of data size on the accuracy of FLDA, we performed ablation experiments by keeping other parameters constant and simulating different user partitions with the same dataset. The attack degree ζ was set to 0.3, as illustrated in

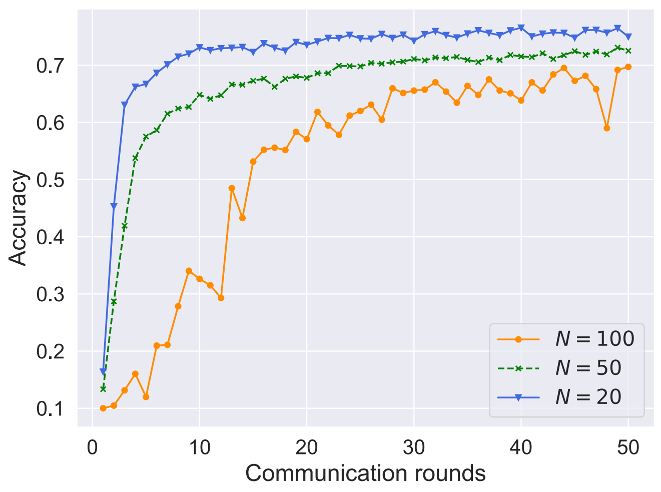

Fig. 10. Accuracy of the number of clients on CIFAR-10.

Figs. 8, 9 and 10. We observed that when there is a larger number of clients, indicating a smaller local data volume, the convergence performance of FLDA deteriorates. Moreover, the increase in the number of clients under the same attack degree results in a higher count of malicious clients, exacerbating the decline in performance. Notably, during the initial communication rounds of federated training, FLDA experienced more pronounced degradation. However, after gradual convergence, the training accuracy of different client sizes N exhibited a decrease ranging from 1% to 4%.

VI. Conclusion

In this article, we propose the FLDA defense framework, which aims to address data poisoning attacks through the implementation of local data mixup on the clients. Initially, we investigate the impact of data poisoning attacks on FL with non-IID data and confirm their detrimental effects. Subsequently, we introduce the FLDA defense framework to alleviate the degradation of model accuracy by employing a lightweight mixup approach on the clients' local training models, enhancing their robustness. Moreover, to leverage the benign client data and reduce the involvement of malicious clients, we develop a gradient history detection algorithm to identify and minimize the participation of suspicious malicious users during server aggregation. To evaluate the effectiveness of FLDA, we conduct experiments on EMNIST, Fashion-MNIST, and CIFAR datasets. The results demonstrate that FLDA successfully mitigates the impact of data poisoning attacks without compromising the overall training performance. However, it is worth noting that when the client data size is small and the attack degree is high, there is a slight reduction in the effectiveness of FLDA's defense. In future work, we plan to delve deeper into the underlying causes of these challenges and explore alternative approaches, such as leveraging the consensus mechanism of clients' blockchain, rather than relying solely on the incentive mechanism. By doing so, we aim to further enhance the defense capabilities and resilience of FLDA against sophisticated attacks.

References

[1] A. D. Boursianis et al., "Internet of Things (IoT) and agricultural unmanned aerial vehicles (UAVs) in smart farming: A comprehensive review," *Internet Things*, vol. 18, May 2022, Art. no. 100187.

[2] D. C. Nguyen et al., "Federated learning for smart healthcare: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–37, Mar. 2023.

[3] Z. Lian et al., "DEEP-FEL: Decentralized, efficient and privacy-enhanced federated edge learning for healthcare cyber physical systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3558–3569, Sep. 2022.

[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[5] J. Guo et al., "ADFL: A poisoning attack defense framework for horizontal federated learning," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 6526–6536, Oct. 2022.

[6] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–36, Jul. 2023.

[7] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," 2022, *arXiv:2207.01982*.

[8] J. Chen, X. Zhang, R. Zhang, C. Wang, and L. Liu, "De-Pois: An attack-agnostic defense against data poisoning attacks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3412–3425, 2021.

[9] X. Xiao, Z. Tang, C. Li, B. Xiao, and K. Li, "SCA: Sybil-based collusion attacks of IIoT data poisoning in federated learning," *IEEE Trans. Ind. Informat.*, vol. 19, no. 3, pp. 2608–2618, Mar. 2023.

[10] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "ShieldFL: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1639–1654, 2022.

[11] F. Khan, R. L. Kumar, M. H. Abidi, S. Kadry, H. Alkhalefah, and M. K. Aboudaif, "Federated split learning model for Industry 5.0: A data poisoning defense for edge computing," *Electronics*, vol. 11, no. 15, p. 2393, Jul. 2022.

[12] Z. Li, Z. Zheng, S. Guo, B. Guo, F. Xiao, and K. Ren, "Disguised as privacy: Data poisoning attacks against differentially private crowdsensing systems," *IEEE Trans. Mobile Comput.*, early access, May 16, 2022, doi: 10.1109/TMC.2022.3173642.

[13] R. Wang, X. Wang, H. Chen, S. Picek, Z. Liu, and K. Liang, "BRIEF but powerful: Byzantine-robust and privacy-preserving federated learning via model segmentation and secure clustering," 2022, *arXiv:2208.10161*.

[14] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[15] K. Ozfatura, E. Ozfatura, A. Kupcu, and D. Gunduz, "Byzantines can also learn from history: Fall of centered clipping in federated learning," 2022, *arXiv:2208.09894*.

[16] C. Xu, Y. Jia, L. Zhu, C. Zhang, G. Jin, and K. Sharif, "TDFL: Truth discovery based Byzantine robust federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4835–4848, Dec. 2022.

[17] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2020, pp. 480–501.

[18] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019.

[19] E. Borgnia et al., "Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 3855–3859.

[20] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.

[21] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 1605–1622.

[22] A. Panda, S. Mahloujifar, A. N. Bhagoji, S. Chakraborty, and P. Mittal, "SparseFed: Mitigating model poisoning attacks in federated learning with sparsification," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2022, pp. 7587–7624.

[23] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[24] J. Fan, Q. Yan, M. Li, G. Qu, and Y. Xiao, "A survey on data poisoning attacks and defenses," in *Proc. 7th IEEE Int. Conf. Data Sci. Cyberspace (DSC)*, Jul. 2022, pp. 48–55.

[25] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 22, 2022, doi: 10.1109/TNNLS.2022.3182979.

[26] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with KPCA and K-means," in *Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Aug. 2021, pp. 551–559.

[27] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li, "FL-WBC: Enhancing robustness against model poisoning attacks in federated learning from a client perspective," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 12613–12624.

[28] P. Gupta, K. Yadav, B. B. Gupta, M. Alazab, and T. R. Gadekallu, "A novel data poisoning attack in federated learning based on inverted loss function," *Comput. Secur.*, vol. 130, Jul. 2023, Art. no. 103270.

[29] I. M. Ahmed and M. Y. Kashmoola, "CCF based system framework in federated learning against data poisoning attacks," *J. Appl. Sci. Eng.*, vol. 26, no. 7, pp. 973–981, 2022.

[30] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "FLCert: Provably secure federated learning against poisoning attacks," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3691–3705, 2022.

[31] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "FedRecover: Recovering from poisoning attacks in federated learning using historical information," 2022, *arXiv:2210.10936*.

[32] S. Ben Saad, B. Brik, and A. Ksentini, "Toward securing federated learning against poisoning attacks in zero touch B5G networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1612–1624, Jun. 2023.

[33] Y. Jiang, W. Zhang, and Y. Chen, "Data quality detection mechanism against label flipping attacks in federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1625–1637, 2023.

[34] X. Cao and N. Z. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 3395–3403.

[35] Z. Zhang, Y. Zhang, D. Guo, L. Yao, and Z. Li, "SecFedNIDS: Robust defense for poisoning attack against federated learning-based network intrusion detection system," *Future Gener. Comput. Syst.*, vol. 134, pp. 154–169, Sep. 2022.

[36] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated anomaly analytics for local model poisoning attack," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 2, pp. 596–610, Feb. 2022.

[37] K. Kumari, P. Rieger, H. Fereidooni, M. Jadliwala, and A.-R. Sadeghi, "BayBFed: Bayesian backdoor defense for federated learning," 2023, *arXiv:2301.09508*.

[38] R. Al Mallah and D. López, "Blockchain-based monitoring for poison attack detection in decentralized federated learning," in *Proc. Int. Conf. Electr., Comput., Commun. Mechatronics Eng. (ICECCME)*, Nov. 2022, pp. 1–6.

[39] A. P. Kalapaaking, I. Khalil, and X. Yi, "Blockchain-based federated learning with SMPC model verification against poisoning attack for healthcare systems," *IEEE Trans. Emerg. Topics Comput.*, early access, Apr. 21, 2023, doi: 10.1109/TETC.2023.3268186.

[40] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. PMLR*, 2017, pp. 1273–1282.

[41] M. Xu, Z. Zou, Y. Cheng, Q. Hu, D. Yu, and X. Cheng, "SPDL: A blockchain-enabled secure and privacy-preserving decentralized learning system," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 548–558, Feb. 2023.

[42] R. Zeng, C. Zeng, X. Wang, B. Li, and X. Chu, "Incentive mechanisms in federated learning and a game-theoretical approach," *IEEE Netw.*, vol. 36, no. 6, pp. 229–235, Nov. 2022.

[43] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2921–2926.

[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[45] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[46] A. Gupta, T. Luo, M. V. Ngo, and S. K. Das, "Long-short history of gradients is all you need: Detecting malicious and unreliable clients in federated learning," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2022, pp. 445–465.

[47] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in Sybil settings," in *Proc. 23rd Int. Symp. Res. Attacks, Intrusions Defenses*, 2020, pp. 301–316.

Chunyong Yin received the B.S. degree from the Shandong University of Technology, Zibo, China, in 1998, and the M.S. and Ph.D. degrees in computer science from Guizhou University, Guiyang, China, in 2005 and 2008, respectively.

He was a Post-Doctoral Research Associate with the University of New Brunswick, Fredericton, Canada, in 2011 and 2012. He is currently a Professor and the Dean with the Nanjing University of Information Science and Technology, Nanjing, China. He has authored or coauthored more than 20 journal and conference papers. His current research interests include privacy preserving and sensor networking, machine learning, and network security.

Qingkui Zeng received the master's degree in computer science and technology from the Anhui University of Science and Technology, Huainan, China, in 2021. He is currently pursuing the Ph.D. degree with Nanjing University of Information Science and Technology, Nanjing, China.

His current research interests are deep learning and network security.