2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS) | 978-1-6654-7315-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPADS56603.2022.00046

FLForest: Byzantine-robust Federated Learning through Isolated Forest

1st Tao Wang
Nanjing University of Aeronautics and Astronautics
Nanjing, China
wangtao1@nuaa.edu.cn

2nd Bo Zhao
Nanjing University of Aeronautics and Astronautics
Nanjing, China
bozhao@nuaa.edu.cn

3rd Liming Fang*
Nanjing University of Aeronautics and Astronautics Shenzhen Research Institute
Shenzhen, China
fangliming@nuaa.edu.cn

***Abstract*—Federated learning (FL) is a privacy-preserving distributed machine learning technique that allows clients to jointly train a global model under the coordination of cloud server. However, malicious clients can corrupt the global model to predict incorrect labels for testing examples. Currently, existing mainstream Byzantine-robust FL methods are vulnerable to various adaptive attacks, and violates the privacy principle of FL. Moreover, these schemes will be less robust in the face of targeted poisoning attacks with few samples and data distributions that are highly non-independent and identically distributed(non-IID). In this work, to address these issues, we propose a novel Byzantine-robust FL framework based on Isolated Forest. Specifically, before the start of each round, FLForest will calculate the divergences between the model update and the model update of the previous round to decide whether to activate the defense. After that, FLForest trains an isolated forest based on model updates after after dimensionality reduction. Model updates isolated with fewer splits will be considered as malicious model updates and excluded from the global model's aggregation. Extensive experiments demonstrate that FLForest achieves better performance compared to baseline methods under highly non-IID distribution.**

***Index Terms*—Federated learning, Byzantine-robust, Robustness rule**

I. Introduction

Federated learning (FL) [1] is a framework that allows multiple clients to collaboratively train a shared model in a distributed environment without transferring their local training data. However, recent works [2] show that it is easy for edge devices (e.g. mobile devices, sensors, computers) to conduct Byzantine attacks (e.g., data/model poisoning attacks) through manipulating local training process to pollute the global model. Such attacks raise great security concerns and have become the roadblocks towards the real-world deployment of FL.

According to the attacker's goal, poisoning attacks could be classifed as *untargeted poisoning attacks*, which aim to make the corrupted global model makes incorrect predictions for a large number of testing examples indiscriminately [3]–[6], or *targeted poisoning attacks*, where the global model predicts attacker-chosen target labels for attacker-chosen target testing examples while the predicted labels for other non-target testing examples are unaffected [7]–[11]. For example, in image classification, the attacker might want the model to misclassify some "white cars" as airplanes, while ensuring that other cars are correctly classified.

Several defenses against poisoning attacks have been proposed to improve the robustness of FL through robust aggregations, clipping local model updates and leveraging the noisy perturbation. The current mainstream methods are mainly divided into two categories. The first kind of scheme assumes that malicious model updates do not exceed half of the total and could be removed if they are away from benign updates in geometric space [2], [12]–[16]. The second kind of scheme assumes that the server has a public clean dataset, and excludes malicious clients by verifying the performance of their local models on public dataset [17]–[19].

However, the current schemes have some limitations. Firstly, for the first kind of scheme, some advanced poisoning attacks could construct poisoning local model updates which are geometrically similar to normal local model updates to bypass defense [7], [9], [20]. Secondly, the second scheme needs to collect a clean dataset directly from the client, which violates FL's privacy principle. Thirdly, most schemes do not perform well when the data distribution is highly non-independent and identically distributed (non-IID), especially under strong stealth attacks such as backdoor attacks. Finally, most schemes remove a certain number of malicious model updates without choice even if the attack does not happen in this round, which will negatively affect the convergence speed of the global model.

To this end, we aim to address the above challenges to build an effective defense system for more general scenarios. In this work, we propose a novel Byzantine-robust FL framework called FLForest that could defend against both *untargeted poisoning attacks* and *targeted poisoning attacks* based on Isolated Forest [21]. Specifically, the server first calculates the divergence between the global model updates of the current and previous round, and compares it with a Laplace random matrix to determine whether the attack occurs. If a certain dimension of the divergence is bigger than the corresponding dimension of the Laplacian matrix, an attack is considered to have occurred. To improve the detection accuracy, we use low variance filtering to perform model dimensionality reduction on model updates. Then, the model updates after dimensionality reduction are classified through Isolated Forest,

* Liming Fang is corresponding author.

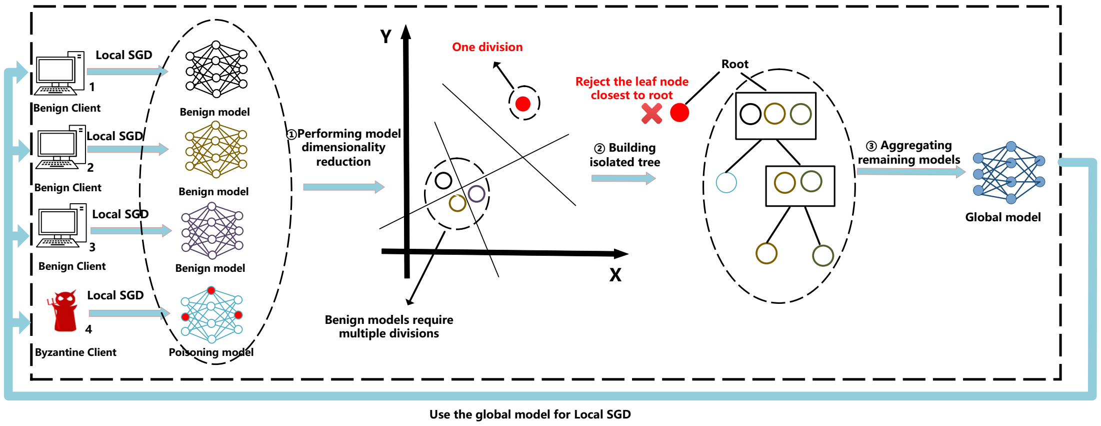


Fig. 1: Overview of FLForest

and the model updates isolated with fewer splits will be considered as malicious model updates and removed. Our key contributions are summarized as follows:

- We propose FLForest, a novel Byzantine-robust FL framework which could defend against both *untargeted poisoning attacks* and *targeted poisoning attacks*. To the best of our knowledge, this is the first Byzantine framework which introduces Isolation Forest into FL and simultaneously detects whether poisoning attack has occurred.
- We propose to determine whether the attack occurs according to the degree of change of the global model update, and use the method of model dimensionality reduction to improve the accuracy of Isolated Forest to identify malicious clients.
- We conduct extensive experiments on different benchmark datasets, various settings (IID and non-IID data distribution), and different model architectures, demonstrating that FLForest is more effective and robust under different poisoning attacks and highly non-IID data distribution compared to the existing Byzantine-robust FL schemes.

II. Background and Related Work

A. Background of Federated Learning

FL is a promising distributed paradigm for training a shared model while keeping all the training data localized. Specifically, the central server is responsible for maintaining and updating the global model and clients are responsible for maintaining and updating the local model. The server aggregates all the local updates to form a global model, and this completes one global epoch. The main method used by FL is `FedAvg` [1] which iteratively performs the following three steps:

- **Step 1.** Each client downloads the latest global model w_{glob} from the server.
- **Step 2.** Each client uses the global model w_{glob} training on the local dataset to minimize $f(w_i; D_i)$, where f is loss function, w_i and D_i is the client's local model and local dataset. The client performs stochastic gradient descent in multiple iterations to obtain the local model w_i. Then, each client sends its local model to the server.
- **Step 3.** The server aggregates the collected local models to update the global model w_{glob}. The most common aggregation rule is to use the amount of data occupied by each client as the weight to perform a weighted average, i.e., $w_{glob} = \frac{1}{|D|}\sum_1^n |D_i|w_i$, where D denotes the joint training data $D_1 \cup D_2 \cup \cdots D_n$.

B. Mitigate poisoning attacks in FL

Many robust aggregation rules have been proposed to mitigate poisoning attacks while maintaining the performance of FL. A typical method is to identify malicious clients by similarity between benign and poisoning models. For example, `Krum` [2] always selects the local model updates with the smallest Euclidean distances to the remaining ones and aggregates them to update the global model. `Trimmed Mean` remove the M biggest and smallest values and aggregate mean of the remaining values. `Median` sort each dimension in descending order and select the median as the final result [12]. `Bulyan` [12] is essentially a variant combination of `Krum` and `Trimmed Mean`. `Bulyan` first iteratively applies `Krum` to select the local model update collection. `Bulyan` then aggregates the local models using a variant of `Trimmed Mean`. However, recent advanced poisoning attacks could constructed poisoning model updates which are geometrically close to normal model updates to bypass such defenses [7], [9], [22].

Another typical rule argues that the server lacks the basis for identifying malicious clients, and could only cluster benign models according to the differences between model updates. Therefore, a clean public dataset is the premise of such methods. Zeno [18] rejects malicious clients based on the loss descent of local model on public dataset. Error Rate Based Rejection (ERR) [7] rejects local model updates that have a large negative impact on the accuracy of the global model. FLTrust [17] trians a server model on the root dataset which sample from the union of the clients' clean local training data. Then, each client is assigned with a trust score by calculating the cosine similarity between the local model and the server model. However, the backdoor models still maintains good performance on the main task because it only affects specific samples. In addition, the assumptions of such methods violate the privacy principle of FL, and a clean public dataset may be difficult to obtain in some FL scenarios.

In addition, [13], [23] show that applying differential privacy to the aggregated global model can improve the robustness against model poisoning attacks. Unfortunately, in extreme cases (highly non-iid data distribution), the existing robust aggregation methods could not prevent pollution from malicious local updates. Thus, there is an urgent necessity to design a novel robust aggregation rule in FL to enhance robustness against poisoning attacks which is complementary to existing robust aggregation approaches.

III. Our FLForest

A. Adversary Setup

In this section, we describe the specific design of FLForest and some relevant setting assumptions. We assume that the number of malicious clients M in the FL system does not exceed 50% of the total number of clients N. At the same time, to make malicious clients harder to be detected, we assume the attacker realizes the aggregation rule and other benign models' parameters.

B. Overview of FLForest

The workflow in each round of FLForest is described as follows (illustrated in Figure 1).

- In **step 1**, the server calculates the change of global model update from $t-1$ to t rounds. If the change is bigger than a Laplacian random matrix, the following steps are performed, otherwise, and the next round of FL is initiated.
- In **step 2**, the server reduces the dimension of the collected local model updates by using low variance filtering to get the most variable dimension.
- In **step 3**, the isolated forest is be trained for the model updates after dimensionality reduction, and the model update with the least number of divisions will be considered as malicious model and removed.

C. Poisoning Attack Detection Mechanism

Based on the training process of FL, we need to build a model of the impact of poisoning attacks on model parameters. Recent work FL-WBC [24] has demonstrated that the reason why attack effects remain in the aggregated model is that the AEPs reside in the kernel of H_i^t where AEPs denote as $\delta_t = w_{glob}^{t+1} - w_{glob}^{t}$, is the change of the global model parameters accumulated until t-th round due to the attack conducted by the malicious devices, and $H_i^t = \nabla^2 F(w_i^t, D_i)$. The essence of H_i^t is second-order partial derivatives of the loss function F, where the diagonal elements describe the change of gradients $\nabla F(w_i^{t+1}) - \nabla F(w_i^t)$ across iterations. $\nabla F(w_i^{t+1}) - \nabla F(w_i^t)$ could be approximated by $(\Delta w_i^{t+1} - \Delta w_i^t)/\eta_i^t$ where η_i^t is a fixing learning rate and Δw_i^t is equal to $w_i^t - w_i^{t-1}$.

Our detection mechanism is based on the following insight: when poisoning attack occurs, the direction of model update should be greatly different or even opposite to the direction of model update in the previous round. It means that $(\Delta w_i^{t+1} - \Delta w_i^t)/\eta_i^t$ should be big. However, since DNN has many parameters, most of the parameters do not change much in the face of some stealth poisoning attacks (e.g. backdoor attack). It is necessary to select the appropriate subset of model parameters for attack detection. We choose the method of low-variance filtering to reduce the model's dimension, calculate the variance of each dimension of the model parameters, and arrange them in descending order. Then, we select a certain number of parameters with the largest variance in order. According to the influence of poisoning attack on parameters, we think that the variable with low variance carries little information, so it could be deleted directly.

Therefore, we design our Poisoning Attack Detection Mechanism (PADM) as follows:

$$PADM = \begin{cases} 0, |(\hat{w}_{glob}^{t+1} - \hat{w}_{glob}^{t}) - \Delta\hat{w}_{glob}^{t}|_{r,c}/\eta^t \leq |\mathbb{L}|_{r,c} \\ 1, |(\hat{w}_{glob}^{t+1} - \hat{w}_{glob}^{t}) - \Delta\hat{w}_{glob}^{t}|_{r,c}/\eta^t > |\mathbb{L}|_{r,c} \end{cases} \tag{1}$$

where we denote $\hat{w}_{glob}^t$ is the global model after dimension reduction in the t-th round. $\hat{w}_{glob}^t$ is aggregated by model update after dimension reduction as follows:

$$\hat{w}_{glob}^{t} = \hat{w}_{glob}^{t-1} - \frac{\eta^t}{N}\sum_{i=0}^{N}\Delta\hat{w}_i^t \tag{2}$$

In this work, we set $\mathbb{L}$ as Laplace noise with $mean = 0$ and $std = \sigma$. $\mathbb{L}_{r,c}$ is the element on r-th row and c-th column of the Laplacian matrix $\mathbb{L}$. Because of the randomness of Laplacian matrix $\mathbb{L}$ and the selectivity of σ, it is difficult for an attacker to choose a suitable attack scale. When it is detected that an element in $|(\hat{w}_{glob}^{t+1} - \hat{w}_{glob}^{t}) - \Delta\hat{w}_{glob}^{t}|_{r,c}/\eta_i^t$ is bigger than or equal to the element in $\mathbb{L}_{r,c}$, it means that in the dimension of r-th row and c-th column , the model update has changed a lot, and PADM think that the poisoning attack has occurred. Therefore, PADM is set to 1, which means FLForest starts to run. Otherwise, continuing to carry out FL of next round.

D. FLForest

When a poisoning attack is detected, FLForest excludes malicious clients by Isolation Forest. Isolation Forest is an unsupervised anomaly detection algorithm that intuitively assumes that anomalies are few compared to normal samples and have unusual eigenvalues [21]. It is a tree-structured algorithm that isolates each instance in the dataset. Since anomalies are few and different, they are closer to the root. The normal point is isolated at the deeper end of the tree.

Isolation Forest requires constructing a collection of isolation trees for training data samples. Anomalies are instances with the shortest average path lengths on these trees. To construct an Isolation Forest, we first need to specify the number of isolation trees to create, and then use the following steps to create an isolation tree:

- **Step 1**. Randomly select Ψ points from the training dataset as subsamples and put them into the root node of an isolated tree.
- **Step 2**. Randomly specify a dimension, and randomly generate a cutting point p within the data range of the current node. The cutting point p is generated between the maximum and minimum values of the specified dimension.
- **Step 3**. The selection of this cutting point p generates a hyperplane, which divides the data space of the current node into two subspaces: the points less than p in the currently selected dimension are placed on the left branch of the current node, and the points greater than or equal to p are placed in the current the right branch of the node.
- **Step 4**. **Step 2** and **3** are repeated recursively until all instances are isolated in the leaf nodes of the isolation tree.

Isolated Forest algorithm has several advantages compared with the clustering-based machine learning model. It does not need to calculate the distance or density metric in high-dimensional data to detect outliers. Therefore, it eliminates the main computational cost of distance calculation required by different distance-based or density-based clustering methods. In addition, Isolation Forest has linear time complexity and low memory requirement.

Compared with the model parameters, the changes of the model parameters can better reflect the difference between the benign models and the poisoned models. Therefore, we take the changes in model updates $(\Delta w_i^{t+1} - \Delta w_i^t)/\eta_i^t$ as the initial dataset for the Isolation Forest. We believe that any kind of poisoning attack will have a large impact on the value of the neuron, so that the model can misclassify the normal samples. And these abnormal neurons are the basis for the division of Isolation Tree.

Isolation forests also could not avoid the dimensional disaster, because the dimensions during training are randomly selected, which will cause a lot of dimensional waste and noise. Fortunately, in the above attack detection stage, we have already obtained the model updates after dimension reduction i.e $(\Delta \hat{w}_i^{t+1} - \Delta \hat{w}^t)/\eta_i^t$. Then, Isolation Forest performs anomaly detection on these, and model updates with an anomaly score of -1 will be flagged as malicious model updates and will not participate in this round of global model's aggregation.

We summarize the procedures of FLForest in Algorithm 1.

Algorithm 1: FLForest

```
Input: Client updates $\{w_i^{t+1} : i \in N\}$ and
       $\{w_i^t : i \in N\}$, the global model $w_{glob}^t$ and
       $w_{glob}^{t+1}$, the learning rate $\eta^t$, Total number of
       clients $N$, maximum possible number of
       Byzantine clients $M$,total number of local
       epochs $E$
Output: the global model $w_{glob}^{t+1}$ with our defense
1  Client $i$ executes:
2  $w_i^t \leftarrow w_{glob}^t$
3  for $e \in \{1, 2, 3, \cdots, E\}$ do
4      for batch $b \in D_i$ do
5          $w_i^{t+1} \leftarrow w_i^t - \eta^t \frac{\partial f(w_i^t; b)}{\partial w_i^t}$
6  return the model $w_i^t$ to server
7  Server executes:
8  $w_{glob} \leftarrow random\, initialization$
9  for $t \in \{1, 2, 3, \cdots, T\}$ do
10     Randomly generate a Laplace noise matrix $\mathbb{L}$ with
         mean = 0 and std = $\sigma$;
11     for $i \in \{1, 2, 3, \cdots, N\}$ do
12         $w_i^{t+1} \leftarrow ClientUpdate$
13     if $t$ is 1 then
14         $PADM \leftarrow 1$
15     else
16         for $i \in \{1, 2, 3, \cdots, N\}$ do
17             $\hat{w}_i^{t+1} \leftarrow w_i^{t+1}$ by dimensionality reduction
18             $\hat{w}_{glob}^{t+1} = \frac{1}{N} \sum_{i=1}^{N} \hat{w}_i^{t+1}$
19         $\Phi \leftarrow |(\hat{w}_{glob}^{t+1} - \hat{w}_{glob}^t) - \Delta \hat{w}_{glob}^t|_{r,c} / \eta^t$
20         if $\Phi|_{r,c} > \mathbb{L}|_{r,c}$ then
21             $PADM \leftarrow 1$
22         else
23             $PADM \leftarrow 0$
24     if PADM is 1 then
25         Train an isolated forest on the dataset
             $\mathbb{W} = \{\Delta w_i^t | i = 1, 2, 3, \cdots, N\}$
26         The abnormal score of poisoning model is $-1$
27         $w_{glob}^{t+1} = \frac{1}{N-M} \sum_{i=1}^{N-M} w_i^{t+1}$
28     else
29         $w_{glob}^{t+1} = \frac{1}{N} \sum_{i=1}^{N} w_i^{t+1}$
```

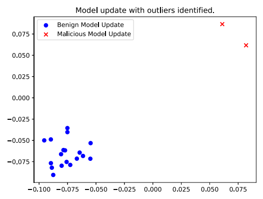
IV. Evaluation

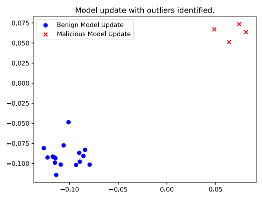
In this section, we conduct an exhaustive experiment on FLForest.

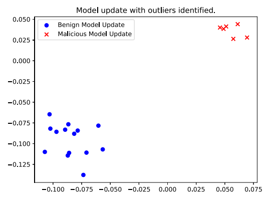
A. Experimental Setting

We examined the effect of FLForest under untargeted attacks on the image benchmark datasets MNIST, CIFAR10 and under targeted attacks on MNIST and Fashion-MNIST [25]–[27].

a) FL System Settings.: We consider an FL system consisting of 100 clients. 20 clients are randomly selected for each round to participate in the training. We divide the dataset into two types: IID and non-IID distributions.


(a) 10% Malicious clients


(b) 20% Malicious clients


(c) 30% Malicious clients

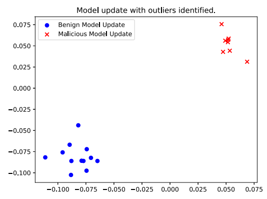
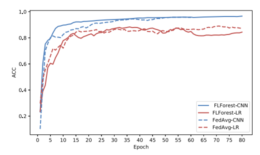

(d) 40% Malicious clients

Fig. 2: Model updates with two dimensions demonstrate that FLForest is able to identify updates from malicious and benign clients. The blue Os and the red Xs represents the model update of benign client and malicious client respectively.


(a) MNIST

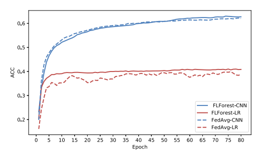
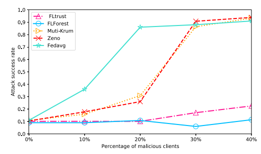

(b) CIFAR

Fig. 3: Accuracies without poisoning attacks on MNIST and CIFAR10.


(a) MNIST

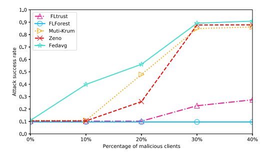

(b) FMNIST

Fig. 4: Attack success rate under different percentge of malicious clients on MNIST and FMNIST

TABLE I: Dataset description and parameters

Dataset	Classes	Examples	Features	Model used
MNIST	10	6000	784	2 conv and 2 fc/LR
CIFAR10	10	5000	1024	4 conv and 2 fc/LR
Fashion-MNIST	10	6000	784	3 conv and 2 fc

To simulate non-IID distribution among these clients, for example, in MNIST which consists a total of $60,000$ training images, we first sort the dataset according to the data label, and then divide it into 200 groups of data slices with a size of 300, and then divide them into two slices for each client when the non-IID degree is 0.2.

To make the attack easier to succeed, we adjust the ratio of malicious clients, backdoor samples and IID/non-IID distribution to create a rigorous environment. Unless otherwise specified, our experiments are performed under the settings of 40% malicious clients, 0.3 non-IID degree and backdoor attack (Backdoor sample 10%).

b) Parameter settings of FLForest.: We set the communication round of the global model as 80 epochs for MNIST and Fashion-MNIST and 100 epochs for CIFAR10. we set the optimizer as Adam [28], the global learning rate η and the local learning rate β as 0.001. We set the batch size $b = 20$ for all datasets except CIFAR10, where we set $b = 32$. The number of local training iterations is 10 for all datasets. The std of the Laplacian random matrix is set to 0.001.

Table I details the information of datasets and model structures used for the experiments.

c) Performance Metrics.: For untargeted attacks, we verify the accuracy of the global model on the validation set. For targeted attack, we verify the accuracy of the global model on the validation set and the attack success rate (accuracy on backdoor samples).

B. Baseline Aggregation Rules

The aggregation rules we used are as follows:

- **FedAvg:** [1] Performing weighted average based on the proportion of the dataset owned by the client.
- **Multi-Krum:** [2] Selecting $N - M - 2$ local models which are closest to the majority group measured by Euclidean Distance, and averaging them to update the global model.
- **Zeno:** [18] Selecting $N - M$ local models with the largest estimated loss descent in public dataset, and averaging them to update the global model.
- **FLTrust:** [17] Selecting local models with cosine similarity similar to that of the server model which trained on a root dataset. Assigning a corresponding score to these local models and averaging them weighted by their scores.

C. Evaluated Poisoning Attacks

We consider both untargeted poisoning attacks and targeted poisoning attacks (Backdoor) as follows:

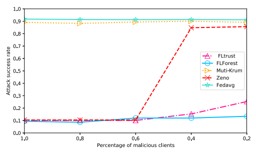
- **Labelflipping attack:** Flipping the labels of the samples into other classes. For example, in MNIST, we label digit "3" as "6", digit "4" as "5", and so on.
- **Back-gradient attack:** [5] Crafting the samples which are adjusted in the direction of increasing loss function.
- **Gaussian attack:** The parameters of the model are randomly generated by Gaussian distribution.

TABLE II: Accuracies under Data Poisoning Attacks.

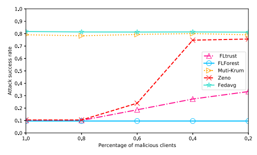
Attack	Defense	CNN		LR	
		MNIST	CIFAR	MNIST	CIFAR
Labelfliping	FedAvg	47.65%	31.43%	33.48%	18.43%
	FLTrust	88.19%	60.57%	77.30%	40.16%
	Mutli-Krum	64.92%	47.09%	38.61%	23.84%
	Zeno	85.28%	61.84%	75.21%	37.72%
	FLForest	**92.87%**	**65.47%**	**80.96%**	**42.52%**
Back-gradient	FedAvg	24.53%	14.74%	18.51%	12.86%
	FLTrust	86.94%	57.71%	78.20%	40.68%
	Mutli-Krum	85.72%	57.03%	75.34%	41.22%
	Zeno	86.49%	59.48%	79.14%	38.60%
	FLForest	**91.94%**	**65.32%**	**81.3%**	**41.87%**

TABLE III: Accuracies under Model Poisoning Attacks.

Attack	Defense	CNN		LR	
		MNIST	CIFAR	MNIST	CIFAR
Gaussian	FedAvg	6.08%	30.68%	5.35%	12.75%
	FLTrust	94.09%	62.40%	87.47%	43.36%
	Mutli-Krum	93.91%	61.18%	88.03%	42.17%
	Zeno	94.16%	63.53%	85.91%	41.01%
	FLForest	**95.94%**	**64.47%**	**90.27%**	**43.77%**
Krum-attack	FedAvg	8.63%	5.94%	29.41%	19.58%
	FLTrust	93.48%	59.73%	81.62%	36.42$
	Mutli-Krum	1.65%	10.46%	1.40%	11.28%
	Zeno	92.09%	63.97%	87.99%	43.39%
	FLForest	**93.52%**	**64.82%**	**87.50%**	**42.91%**

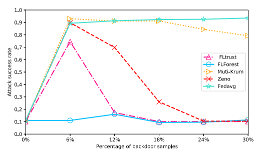


(a) MNIST

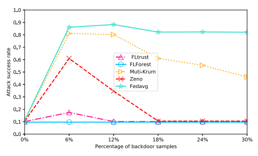


(b) FMNIST

Fig. 5: Attack success rate under different degree of non-IID on MNIST and FMNIST



(a) MNIST



(b) FMNIST

Fig. 6: Attack success rate under different percentge of backdoor samples on MNIST and FMNIST

- **Krum attack:** [7] Constructing a local model that is the closest in Euclidean distance to the other local models, but in the opposite direction to the global model.
- **Badnet:** [29] Injecting a trigger or pattern onto the sample and flipping the label in each round. Trained models will misclassify because of specific triggers.
- **Backdoor FL:** [30] In the last iteration of FL, the scaled backdoor model was submitted to replace the global model.
- **Distributed Backdoor Attack (DBA)**: [8] DBA decomposes a global trigger pattern into separate local patterns and embed them into the training set of different adversarial parties respectively.

D. Experiment result

1) Validity of FLForest on Geometric Spaces: Fig. 2 shows the distribution of benign model updates and poisoned model updates with two dimensions in geometric space when the proportion of malicious clients varies from 10% to 40%. We can observe that benign model updates and poisoned model updates clearly belong to different clusters. The directions of benign model update and poisoned model update are very different, which can be seen from the difference in the quadrants of the two distributions. This shows that it is feasible for FLForest to increase the recognition success rate through dimensionality reduction.

2) Convergence of FLForest: Fig. 3 shows the convergence of FLForest and `FedAvg` on CNN and LR under highly non-IID distribution. We observe that FLForest performs as well as `FedAvg` without poisoning attacks.

3) Impact of the number of malicious clients: Fig. 4 shows the attack success rate of different FL methods under Badnet attack on MNIST-0.2 and Fashion-MNIST-0.2, when the fraction of malicious clients increases from 0 to 40%. As we can see, FLForest achieves the lowest attack success rate comparing to other baseline methods. For backdoor attacks, the more malicious clients participating, the more conducive to backdoor injection. This can be seen from the fact that the success rate of attacking `FedAvg` increases with the increase of the proportion of malicious clients. When the proportion of malicious clients in `Zeno` and `Mutli-Krum` reaches 30%, the attack success rate increases rapidly which is closing to `FedAvg`. And `FLTrust` shows a lower attack success rate, but still higher than our FLForest. These results further demonstrate that FLForest has better performance in the face of backdoor attack with 40% malicious clients.

4) Impact of non-IID Degree: Fig. 5 shows that the attack success rate of different FL methods where non-IID degree ranges from 1 to 0.2. 1 represents the IID distribution. As the number continues to decrease, the degree of non-IID continues to increase. The results demonstrate that the attack success rate rises with higher degree of non-IID for baseline schemes. Because higher degree of non-IID increases divergences between benign model updates, making these baseline methods harder to distinguish malicious model updates. Surprisingly, FLForest still maintains the same attack success rate as IID under 0.2-non-IID, which fully demonstrates FLForest's tolerance to non-IID.

5) Impact of Backdoor Sample Percentage: Fig. 6 illustrates the attack success rate of different FL methods under BadNet attack where the backdoor sample percentage ranges from 0% to 30%. The experimental results show that when the proportion of backdoor samples is 6%, the attack success rate of other FL methods except FLForest is high. This is because when the proportion of backdoor samples is small, the distance between model parameters and performance of

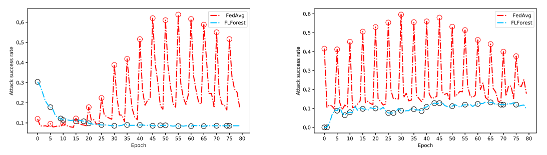
TABLE IV: Accuracies under Untargeted Attacks. The accuracy rate and attack success rate of the main task are separated by /.

Attack	Defense	MNIST	Fashion
BadNet	FedAvg	94.01%/93.55%	86.49%/83.78%
	FLTrust	95.32%/84.36%	85.65%/75.27%
	Mutli-Krum	96.73%/93.42%	84.31%/60.63%
	zeno	95.56%/94.39%	86.40%/81.61%
	FLForest	**96.65%/12.80%**	**86.78%/9.54%**
Backdoor FL	FedAvg	97.85%/96.34%	85.49%/84.56%
	FLTrust	96.95%/10.02%	83.65%/15.31%
	Mutli-Krum	93.07%/11.96%	79.34%/30.16%
	zeno	94.66%/9.48%	84.71%/15.50%
	FLForest	**97.54%/9.20%**	**85.7%/13.18%**
DBA	FedAvg	96.23%/94.12%	85.54%/83.70%
	FLTrust	93.67%/54.73%	86.10%/44.37%
	Mutli-Krum	94.91%/68.36%	85.64%/73.08%
	zeno	94.65%/39.10%	84.21%/52.16%
	FLForest	**96.14%/14.8%**	**86.29%/15.6%**

model do not change much while FLForest removes useless model parameters and improves the success rate of backdoor identification. The performance of baseline methods keeps improving when the proportion of backdoor samples increases. For example, when the proportion of backdoor samples is 24%, `FLTrust` and `Zeno` obtain the same low attack success rate as FLForest. And FLForest always maintains a low attack success rate.

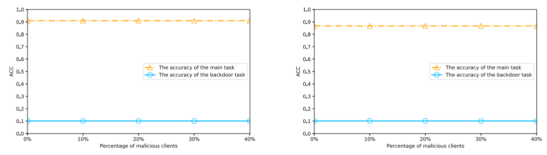
6) Effectiveness of PADM: We test the effectiveness of PADM on FLForest under BadNet in Fig. 7. We set malicious clients to submit poisoning models every five rounds. The red circles represent the rounds of poisoning attacks, and the black circles represent the rounds that PADM detected poisoning attacks. The results show that PADM detected all rounds of poisoning attacks which fully demonstrates the effectiveness of PADM. Also, we can observe that in some rounds where the poisoning attack did not occur, PADM still considers the attack to be detected. Because highly non-IID may has a large change in some dimensions of the model update, resulting in these changes still being retained after dimensionality reduction, making PADM think that the attack occurs. However, there is no impact on the final performance of the model.

7) Defense against Colluding Attacks: We finally examine the performance of FLForest in colluding attacks on malicious clients. We design a collusion attack against FLForest, assuming that malicious client knows the dimensionality reduction ratio set by the server and the parameters of benign model update. Then, malicious client calculates the average value of benign model updates after dimension reduction and sets the corresponding dimension of all poisoned model updates to this value to ensure that the poisoned model can survive. We present the accuracy in main task and attack success rate when the fraction of malicious clients increases from 0 to 40% in Fig. 8. As we can see, FLForest still maintains a low attack success rate which shows the robustness of FLForest to collusion attacks. This is because if a malicious client modifies fewer dimensions, the other dimensions become new divisions of the isolation forest. Conversely, if the malicious client modifies a large number of dimensions, then the harm of the model update will be greatly reduced.



(a) MNIST (b) FMNIST

Fig. 7: Attack success rate of PADM when poisoning attack occurs on MNIST and FMNIST



(a) MNIST (b) FMNIST

Fig. 8: Attack success rate under colloing attack on MNIST and FMNIST

E. Robustness Against Untargeted Poisoning Attacks

We demonstrate the overall performance of FLForest and other baseline methods under different data poisoning attacks in Table II and Table III. Experimental results show that `FedAvg` is not robust to any kind of data poisoning attack. We can observe that FLForest achieves better performance than existing baseline schemes, especially under the untargeted attack that is optimized for the method. For instance, on MNIST and CNN, when `Krum` faced Krum attack, the accuracy dropped to 1.65%. Even `FedAvg` is more accurate than `Krum` because this untargeted attack is not optimized for `FedAvg`. In the face of non-optimized attacks such as Gaussian attacks, most FL schemes have good performance while `FedAvg`'s performance drops rapidly. Furthermore, when faced with labelflipping attack on MNIST-0.2, the accuracy of `FLTrust`, `Mutli-Krum` and `Zeno` drops by 5.9%, 29.09% and 8.88% respectively compared to Gaussian attack. And FLForest still performs well compared to other schemes under highly non-IID environment and poisoning attack of 40% malicious client.

F. Robustness Against Targeted Poisoning Attacks

Table. IV shows that main task accuracy and attack success rate of different FL methods in the face of target poisoning attacks. We can observe that existing FL methods except FLForest achieve a high attack success rate against BadNet attacks with a backdoor sample ratio of 6%. This shows that the existing FL methods are not robust under untargeted attacks with few samples and highly non-IID environment. While backdoor FL only submits the scaled model in the last round, most of the existing FL schemes are robust. As we can see, FLForest can maintain high accuracy on validation set while maintaining a low attack success rate. The accuracy of the global model on backdoor samples is always maintained at about 10%, because we inject backdoors on some samples of each class and flip the labels to the same. This is in line with our expectations.

V. Conclusion

In this paper, we propose a novel Byzantine-robust FL framework FLForest to achieve Byzantine robustness against malicious clients based on Isolated Forest. Through model dimensionality reduction, malicious clients are more easily identified by Isolated Forest. The biggest difference between FLForest and other FL robust methods is that an attack detection mechanism is added, which reduces the workload of the server in the absence of attacks and increases the generalization of the global model. Moreover, FLForest does not require prior knowledge of clients' data distribution or public dataset and inherits the good spatiotemporal complexity of Isolation Forest. We conduct exhaustive experiments on 3 public datasets and under a variety of different settings, and compare with 4 different baseline methods. The experimental results have proved that FLForest achieves better performance compared to other Byzantine-robust FL schemes under different poisoning attacks, especially in highly non-IID distribution.

Acknowledgment

This work is supported by the National Key R&D Program of China (Grant No.2021YFB3100700), the National Natural Science Foundation of China (No.61872181), Natural Science Foundation of Jiangsu Province (Grant No.BK20200418, BE2020106) and the Shenzhen Science and Technology Program (Grant No. JCYJ20210324134408023).

References

[1] 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.

[2] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 118–128.

[3] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. [Online]. Available: http://icml.cc/2012/papers/880.pdf

[4] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 19–35.

[5] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 27–38.

[6] C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *CoRR*, vol. abs/1808.04866, 2018. [Online]. Available: http://arxiv.org/abs/1808.04866

[7] 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.

[8] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, 2019.

[9] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," *Internet Society*, p. 18, 2021.

[10] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *arXiv preprint arXiv:2007.05084*, 2020.

[11] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[12] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[13] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *CoRR*, vol. abs/1911.07963, 2019. [Online]. Available: http://arxiv.org/abs/1911.07963

[14] 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.

[15] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 2019, pp. 1544–1551. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33011544

[16] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

[17] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," in *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/fltrust-byzantine-robust-federated-learning-via-trust-bootstrapping/

[18] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901.

[19] ——, "Zeno++: Robust fully asynchronous sgd," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 495–10 503.

[20] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[22] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[23] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.

[24] 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," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 613–12 624, 2021.

[25] Y. LeCun, "The mnist database of handwritten digits," 1998. [Online]. Available: http://yann. lecun. com/exdb/mnist/

[26] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[27] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[30] 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.