FedInv: Byzantine-Robust Federated Learning by Inversing Local Model Updates

Bo Zhao,[1] Peng Sun,[2,3,*] Tao Wang,[1] Keyu Jiang[1]
[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
[2]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China
[3]Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), China
bozhao@nuaa.edu.cn, sunpeng@cuhk.edu.cn, {wangtao1, jiangky}@nuaa.edu.cn

Abstract

Federated learning (FL) is a privacy-preserving distributed machine learning paradigm that enables multiple clients to collaboratively train statistical models without disclosing raw training data. However, the inaccessible local training data and uninspectable local training process make FL susceptible to various Byzantine attacks (e.g., data poisoning and model poisoning attacks), aiming to manipulate the FL model training process and degrade the model performance. Most of the existing Byzantine-robust FL schemes cannot effectively defend against stealthy poisoning attacks that craft poisoned models statistically similar to benign models. Things worsen when many clients are compromised or data among clients are highly non-independent and identically distributed (non-IID). In this work, to address these issues, we propose FedInv, a novel Byzantine-robust FL framework by inversing local model updates. Specifically, in each round of local model aggregation in FedInv, the parameter server first inverses the local model updates submitted by each client to generate a corresponding dummy dataset. Then, the server identifies those dummy datasets with exceptional Wasserstein distances from others and excludes the related local model updates from model aggregation. We conduct an exhaustive experimental evaluation of FedInv. The results demonstrate that FedInv significantly outperforms the existing robust FL schemes in defending against stealthy poisoning attacks under highly non-IID data partitions.

Introduction

Federated learning (FL) (McMahan et al. 2017; Kairouz et al. 2019) is a distributed machine learning framework that enables multiple clients to collaboratively train statistical models without raw data exchange under the orchestration of a parameter server. Thus, FL can significantly alleviate clients' privacy concerns compared to the conventional centralized machine learning paradigm. However, FL is susceptible to Byzantine attacks (e.g., data/model poisoning attacks) (e.g., (Jagielski et al. 2018; Biggio, Nelson, and Laskov 2012; Chen et al. 2017; Fang et al. 2018; Shafahi et al. 2018; Li et al. 2016; Fang et al. 2020; Shejwalkar and Houmansadr 2021)) due to the inaccessibility of clients' local training data and the uninspectable local training processes. Effective Byzantine-robust FL schemes (i.e., defense mechanisms) are highly desired to achieve satisfactory model performance via FL model training.

Thus far, researchers have proposed many Byzantine-robust FL schemes, which can be roughly grouped into two categories. The first kind of scheme (e.g., (Blanchard et al. 2017; Yin et al. 2018; Mhamdi, Guerraoui, and Rouault 2018)) identifies and excludes poisoned local model updates under the assumption that they are geometrically far away from benign ones. The second kind of scheme (e.g., (Xie, Koyejo, and Gupta 2019, 2020)) assumes the server maintains a public clean validation dataset. The server uses this dataset to evaluate local model updates in terms of test accuracy or loss. Those with poor performance are determined as poisoned local models and then are excluded from local model aggregation. Very recently, some works (e.g., (Cao et al. 2020; Prakash and Avestimehr 2020)) combine the above two and propose hybrid robust FL schemes.

However, most of the existing Byzantine-robust FL schemes have the following limitations:

- First, as demonstrated in (Fang et al. 2020; Shejwalkar and Houmansadr 2021), several advanced poisoning attacks could craft poisoned local models that are geometrically close to benign ones. Thus, the first kind of scheme's assumption (also the precondition) is violated, leading to its ineffectiveness.
- Second, centralizing a public validation dataset and using it to evaluate local model updates in the second kind of scheme cannot defend against advanced poisoning attacks with dual optimization objectives, such as backdoor attacks (Gu, Dolan-Gavitt, and Garg 2017; Bagdasaryan et al. 2020). Moreover, the centralization of a public validation dataset disobeys the principle of FL to some degree and may not be realized in some real-world cases.
- Third, all of these defense mechanisms suffer from significant performance degradation when a relatively large proportion of clients are compromised, or data among clients is highly non-independent and identically distributed (non-IID).

To tackle the issues mentioned above, in this work, we propose FedInv, a novel Byzantine-robust FL scheme by inversing local model updates. FedInv could effectively de-

[*]Peng Sun is the corresponding author.

fend against stealthy data and model poisoning attacks, even when the number of Byzantine clients is large and data partitions among clients are highly non-IID, thus promising satisfactory performance of FL model training.

The basic idea of FedInv is that before updating the global model in each round of FL, the server first performs privacy-respecting model inversion on each client's local model updates to synthesize a corresponding dummy dataset. Then, the Wasserstein distances of each generated dummy dataset to the remaining ones are calculated. Those clients associated with huge distances are determined as Byzantine clients, whose local model updates are excluded from local model aggregation.

The major contributions of this work are as follows:

- To the best of our knowledge, FedInv is the first Byzantine-robust FL framework that could thwart stealthy poisoning attacks, even under a rigorous adversarial setting with a great number of Byzantine clients and highly non-IID data distribution. Besides, FedInv eliminates the need for centralizing any data at the server.
- We propose to perform privacy-respecting model inversion on each client's local model update to synthesize a corresponding dummy dataset. By comparing the distribution divergences among these dummy datasets, we could always identify the benign clients and aggregate their submitted local model updates to achieve Byzantine-robust FL.
- We conduct an exhaustive experimental evaluation of FedInv on various datasets, models, adversary settings, attack methods. The results corroborate its superior performance compared to the existing Byzantine-robust FL schemes.

Background and Related Work

Background on Federated Learning

Federated learning (FL) enables multiple clients to collaboratively train globally shared machine learning models without disclosing their raw training data under the coordination of a parameter server. Consider an FL system involving N clients. Each client n ($n = 1, 2, \ldots, N$) has a training dataset $\mathcal{D}_n$ consisting of D_n data samples (i.e., $|\mathcal{D}_n| = D_n$). Generally, FL model training involves many rounds of interactions between clients and the server. In each round t ($t \in \{1, 2, \ldots, T\}$), the following steps are performed.

Step ① : each client n downloads the latest global model $\boldsymbol{\theta}^{t-1}$ from the server (note that $\boldsymbol{\theta}^0$ at the very beginning (i.e., $t = 1$) is a random initialization).

Step ② : each client n uses her training data to perform local model update on $\boldsymbol{\theta}^{t-1}$ via mini-batch stochastic gradient descent (SGD), and submits the local model update $\boldsymbol{\theta}_n^t$ to the server.

Step ③ : the server aggregates local model updates $\boldsymbol{\theta}_n^t$ from all the clients $n = 1, 2, \ldots, N$ to produce the global model update $\boldsymbol{\theta}^t$. Formally, $\boldsymbol{\theta}^t = \mathcal{A}(\boldsymbol{\theta}_n^t, n = 1, 2, \ldots, N)$, where $\mathcal{A}$ represents the adopted aggregation rule.

A classic and widely-adopted aggregation rule is federated averaging (FedAvg), where the local model updates are aggregated as $\boldsymbol{\theta}^t = \sum_{n=1}^{N} \frac{D_n}{\sum_{n=1}^{N} D_n} \boldsymbol{\theta}_n^t$.

Recent works (e.g., (Blanchard et al. 2017)) have demonstrated that the global model generated by a simple local model aggregation weighted by data size as in FedAvg can be easily manipulated by Byzantine clients conducting poisoning attacks. Thus, Byzantine-robust FL schemes (i.e., aggregation rules) are highly desired.

Byzantine-robust Federated Learning

One kind of Byzantine-robust FL scheme identifies and excludes poisoned local model updates under the assumption that they are geometrically far from benign ones. For example, `Krum` (Blanchard et al. 2017) 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` and `Median` (Yin et al. 2018) remove the F biggest and smallest values and take the average and median of the remaining ones as the aggregated value for each of the model parameters among all the local model updates. However, some recently proposed advanced poisoning attacks (Fang et al. 2020; Shejwalkar and Houmansadr 2021) can craft poisoned models geometrically close to benign ones, which will break the assumption (also precondition) of these schemes and circumvent the defenses.

Another kind of scheme needs to centralize a public clean validation dataset and use it to evaluate local model updates in terms of test accuracy or loss. Those with poor performances are excluded from local model aggregation. For example, the Error Rate based Rejection (ERR) (Fang et al. 2020) rejects local mode updates that have a large negative impact on the global model's accuracy. `Zeno` (Xie, Koyejo, and Gupta 2019, 2020) and the Loss Function-based Rejection (LFR) (Fang et al. 2020) use the loss decrease on the centralized validation dataset to rank the model's credibility. However, *poisoning attacks with dual optimization objectives like backdoor attacks (e.g., (Gu, Dolan-Gavitt, and Garg 2017)) may also have a good performance on the validation dataset and can escape from exclusion*. Besides, *these schemes violate the privacy-preserving principle (i.e., remaining data localized) of FL and may be difficult to implement in practice*.

Very recent studies combine the above two kinds of schemes and propose hybrid defense mechanisms. For example, `FLTrust` (Cao et al. 2020) trains a bootstrapping model on a public dataset and uses cosine similarities between local model updates and the trained bootstrapping model to rank the model's credibility. However, *`FLTrust` may also inherit the limitations of the two kinds of schemes mentioned above*. Similarly, `DiverseFL` (Prakash and Avestimehr 2020) trains a bootstrapping model for each client using some of the client's local data and compares the trained model with her submitted local model update to examine the local training process. However, *`DiverseFL` cannot deal with data poisoning attacks*.

Moreover, existing Byzantine-robust FL schemes suffer significant performance degradation (empirically validated

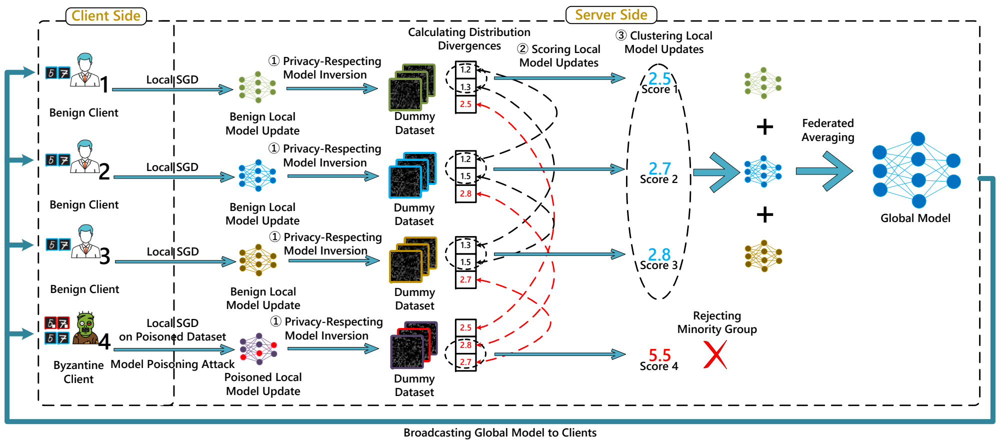
Client Side
Server Side
Calculating Distribution Divergences
② Scoring Local Model Updates
③ Clustering Local Model Updates
1
Local SGD
① Privacy-Respecting Model Inversion
Benign Client
Benign Local Model Update
Dummy Dataset
2
3
4
Local SGD on Poisoned Dataset
Byzantine Client
Model Poisoning Attack
Poisoned Local Model Update
1.2
1.3
2.5
1.5
2.8
2.7
2.5 Score 1
2.7 Score 2
2.8 Score 3
5.5 Score 4
Rejecting Minority Group
Federated Averaging
Global Model
Broadcasting Global Model to Clients

Figure 1: Schematic diagram of FedInv.

in our experiments) when a relatively large number of clients are compromised and the data distribution among clients is highly non-IID. FedInv aims to address all these issues. The detailed design is presented in the next section.

Detailed Design of FedInv

Adversary Setup

Before presenting the design details of FedInv, we first introduce the adversary setup. Specifically, FedInv aims at a rigorous adversary setup, where up to F ($F/N < 0.5$) clients can be compromised by the attacker, and Byzantine clients can exchange the necessary information to collude to launch poisoning attacks. Moreover, the local model updates from benign clients and the aggregation rule employed by the server are accessible by the attacker, enabling it to craft poisoned models similar to benign ones, making them difficult to distinguish.

Overview of FedInv

The framework of FedInv is shown in Figure 1, and the workflow in each round of FedInv is described as follows.

- **Step ① : Privacy-Respecting Model Inversion:** Performing privacy-respecting model inversion on each received local model update to synthesize a corresponding dummy dataset.
- **Step ② : Scoring Local Model Updates:** Scoring each local model update according to the Wasserstein distances of its corresponding dummy dataset to the majority.
- **Step ③ : Clustering Local Model Updates:** Local model updates are clustered into two groups based on their derived scores. Those in the majority group are aggregated to update the global model.

Privacy-Respecting Model Inversion

Model inversion (MI) under the white box setting refers to reconstructing the training data from model parameters. The recent work named Deep Leakage from Gradients (DLG) (Zhu and Han 2020) studied model inversion in FL, where the raw training data of clients are inferred from their shared gradients. Specifically, DLG first randomly generates a dummy dataset including inputs and corresponding labels. Then, the resulting dummy gradients are calculated over the current global model. DLG iteratively updates the dummy data to make it close to the clients' raw training data via minimizing the difference between dummy gradients and the shared actual gradients. Formally, to infer the training data (x, y) used for local model update at client n in round t, DLG solves the following optimization problem.

$$(x'^{*}, y'^{*}) = \underset{(x',y')}{\arg\min} \| \frac{\partial \ell((x', y'); \boldsymbol{\theta}^{t-1})}{\partial \boldsymbol{\theta}^{t-1}} - \boldsymbol{g}_n^t \|_2^2, \qquad (1)$$

where (x', y') (x' is the input, y' is the corresponding label) is the dummy data to be optimized, $\boldsymbol{\theta}^{t-1}$ is the current global model, $\ell(\cdot)$ is the loss function, and $\boldsymbol{g}_n^t$ is the calculated gradient over one single mini-batch of training data at client n in round t. The optimal solution (x'^{*}, y'^{*}) to the problem (1) can be very close to (x, y).

As demonstrated in (Zhu and Han 2020), directly sharing $\boldsymbol{g}_n^t$ leads to potential privacy leakage for client n. To alleviate clients' privacy concerns, FedInv allows each client n to perform E epochs of local model update via mini-batch SGD with a size of B. In this case, client n performs ED_n/B steps of local SGD. Then, FedInv requires each client n to submit her local model update $\boldsymbol{\theta}_n^t$ in round t along with her adopted number of local epochs E and mini-batch size B. At the server side, different from (Zhu and Han 2020), FedInv performs model inversion by matching the dummy gradient with the equivalent gradient $(\boldsymbol{\theta}_n^t - \boldsymbol{\theta}^{t-1})/(ED_n/B)$.

The equivalent gradient starts to cancel out, i.e., $(\boldsymbol{\theta}_n^t - \boldsymbol{\theta}^{t-1})/(ED_n/B) \rightarrow 0$, as the global model converges (Bagdasaryan et al. 2020). To enlarge the differences among the equivalent gradients of different clients such that the differences among clients' underlying data distributions can be clearly identified via model inversion, we introduce a scaling factor $s^{(t)}$ (i.e., s to the power of t) for each round t, and s is a hyperparameter to be tuned. Specifically, we turn to matching the dummy gradients with the scaled equivalent gradients $s^{(t)}(\boldsymbol{\theta}_n^t - \boldsymbol{\theta}^{t-1})/(ED_n/B)$. Besides, instead of optimizing both x' and y' of each dummy data sample, FedInv only optimizes x', while y' is sampled uniformly at random from all possible labels of the dataset and fixed. Then, the optimization problem (1) is reformulated as

$$x'^* = \arg\min_{x'} \| \frac{\partial \ell((x', y'); \boldsymbol{\theta}^{t-1})}{\partial \boldsymbol{\theta}^{t-1}} - \frac{s^{(t)}(\boldsymbol{\theta}_n^t - \boldsymbol{\theta}^{t-1})}{ED_n/B} \|_2^2. \tag{2}$$

Furthermore, to clearly show the differences among the dummy data of different clients, we respectively subtract the random initialization of the dummy data from the optimal dummy data x'^* and obtain a final dummy data for each client.

Scoring and Clustering Local Model Updates

For each client, we employ the Wasserstein distance metric (Weng 2019) to calculate the distribution divergences of her dummy dataset from the remaining ones. In particular, the distribution divergence $\mathcal{W}(x'_i, x'_j)$ between two dummy datasets (x'_i, y'_i) and (x'_j, y'_j) is computed as

$$\mathcal{W}(x'_i, x'_j) = \sum_{c=1}^{l} \sum_{d=1}^{m} Wass(x_i'^{,c,d}, x_j'^{,c,d}), \tag{3}$$

where $x_i'^{,c,d}$ and $x_j'^{,c,d}$ respectively represents the vector composed of the d-th features of all samples with label c in (x'_i, y'_i) at client i and (x'_j, y'_j) at client j, l and m are respectively the total number of labels and features of the synthesized dummy datasets. Note that $\mathcal{W}(x'_i, x'_j)$ can (to some degree) reflect the divergence between the underlying training data distributions of clients i and j.

Intuitively, since in our adversary setup, the majority of clients are benign (recall $F/N < 50\%$), the dummy datasets derived from poisoned local model updates have larger distribution divergences from the remaining ones. However, this may not be the case when the data distribution among clients is highly non-IID. Thus, instead of simply rejecting the local model updates from which the generated dummy datasets have larger distribution divergences, we consider clustering the local model updates from which the generated dummy datasets have moderate (not too large or too small) distribution divergences.

Specifically, when scoring local model update $\boldsymbol{\theta}_i^t$ from client i, FedInv determines $\mathcal{S} = \{\mathcal{W}(x'_i, x'_j) | j = 1, \ldots, N\}$, and extracts the majority group $\mathcal{M}$ from $\mathcal{S}$ via 2-Median clustering. If $|\mathcal{M}| > N - F$, then $|\mathcal{M}| - (N - F)$ elements in $\mathcal{M}$ with the largest distances to the median will be filtered out from $\mathcal{M}$. Then, the score of $\boldsymbol{\theta}_i^t$, denoted as q_i, is the summation of all elements in $\mathcal{M}$. Then, FedInv determines the score for each client, i.e., $\mathcal{Q} = \{q_i | i = 1, \ldots, N\}$, and extracts the majority group of $\mathcal{Q}$ via the same 2-Median clustering technique. The clients remained in the majority group are determined as benign clients, whose local model updates are aggregated to update the global model.

We summarize the procedures of FedInv in Algorithm 1.

Algorithm 1: FedInv

```
Input: Total number of clients $N$, total number of
       communication rounds $T$, total number of
       local epochs $E$, mini-batch size $B$, maximum
       possible number of Byzantine clients $F$,
       scaling factor $s$
Output: Global model $\boldsymbol{\theta}^T$
1  Server executes:
2  initialize $\boldsymbol{\theta}^0$
3  for $t \in \{1, 2, \ldots, T\}$ do
4      for $n \in \{1, 2, 3, \ldots, N\}$ in parallel do
5          $\boldsymbol{\theta}_n^t \leftarrow$ ClientUpdate($n, \boldsymbol{\theta}^{t-1}$)
6      end
7      initialize $x'$
8      for $n \in \{1, 2, 3, \ldots, N\}$ do
9          $x_n'^* \leftarrow$ Solving Equation (2)
10         $x'_n \leftarrow x_n'^* - x'$
11     end
12     $\mathcal{Q} \leftarrow \emptyset$
13     for $i \in \{1, 2, 3, \ldots, N\}$ do
14         $\mathcal{S} \leftarrow \emptyset$
15         for $j \in \{1, 2, 3, \ldots, N\} \setminus i$ do
16             $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{W}(x'_i, x'_j)$
17         end
18         $\mathcal{M} \leftarrow$ 2-Median($\mathcal{S}$)
19         if $|\mathcal{M}| > N - F$ then
20             remove the $|\mathcal{M}| - (N - F)$ elements with
                 the largest distances to the median in $\mathcal{M}$
21         end
22         $q_i \leftarrow$ Sum($\mathcal{M}$)
23         $\mathcal{Q} \leftarrow \mathcal{Q} \cup q_i$
24     end
25     $\mathcal{M} \leftarrow$ 2-Median($\mathcal{Q}$)
26     if $|\mathcal{M}| > N - F$ then
27         remove the $|\mathcal{M}| - (N - F)$ elements with
             the largest distances to the median in $\mathcal{M}$
28     end
29     $\Gamma \leftarrow \{i | q_i \in \mathcal{M}\}$
30     $\boldsymbol{\theta}^t \leftarrow \sum_{n \in \Gamma} \frac{D_n}{\sum_{n \in \Gamma} D_n} \boldsymbol{\theta}_n^t$
31 end
32 ClientUpdate($n, \boldsymbol{\theta}$): // run on client $n$
33 $\mathcal{D}_n \leftarrow$ (split $\mathcal{D}_n$ into batches of size $B$)
34 $\boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}$
35 for $e \in \{1, 2, 3, \ldots, E\}$ do
36     for batch $b \in \mathcal{D}_n$ do
37         $\boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}_n - \alpha \frac{\partial \ell(b; \boldsymbol{\theta}_n)}{\partial \boldsymbol{\theta}_n}$
38     end
39 end
40 return $\boldsymbol{\theta}_n$ to server
```

Experiments

Experimental Setup

Datasets We use multiple datasets from different domains in our experiments, including two image classification datasets and one human activity recognition (HAR) dataset.

- **MNIST** (LeCun, Cortes, and Burges 1998): 10-class handwritten digit image classification dataset consisting of 60000 training samples and 10000 testing samples.
- **Fashion-MNIST** (Xiao, Rasul, and Vollgraf 2017): 10-class fashion image classification dataset consisting of 60000 training samples and 10000 testing samples.
- **HAR** (Anguita et al. 2013): 6-class human activity recognition dataset collected via sensors embedded in 30 users' smartphones (21 users' datasets used for training and 9 for testing, and each has around 300 data samples).

We consider an FL system consisting of 20 clients. To simulate (different degrees of) non-IID data distribution among these clients, we adopt the same method as in (Cao et al. 2020). Specifically, for datasets with M classes, we divide the 20 clients evenly into M groups. For example, in this work, MNIST and Fashion-MNIST have 10 classes, i.e., $M = 10$. For HAR, we select 20 out of the 21 clients whose datasets are used for training and consider the data from one specific user as one class, i.e., $M = 20$. Then, we assign a data sample with class c ($c = 1, \dots, M$) to group c with probability $p > 0$ and to any other group with probability $(1 - p)/(M - 1)$. Within the same group, data are uniformly distributed to each client. The value of p indicates the non-IID degree of the data distribution among clients. In particular, data are IID if $p = 1/M$ and non-IID otherwise. A larger value of p implies a higher degree of non-IID.

Models We train various global models on the above three datasets to show the generality of FedInv. Specifically, for MNIST and Fashion-MNIST, we train a convolutional neural network (CNN) (Conv2d(1*6*3) $\rightarrow$ Softplus $\rightarrow$ MaxPool2d(2*2) $\rightarrow$ Conv2d(6*25*3) $\rightarrow$ Softplus $\rightarrow$ MaxPool2d(2*2) $\rightarrow$ Linear(1225*50) $\rightarrow$ Softplus $\rightarrow$ Linear(50*10) $\rightarrow$ Softplus) and a logistic regression (LR) classifier as the global models; for HAR, we train a fully-connected neural network (FCN) (Linear(561*150) $\rightarrow$ Softplus $\rightarrow$ Linear(150*6) $\rightarrow$ Softplus) and an LR classifier as the global models.

Parameter Settings of FL Model Training We train the global models for 20 communication rounds. In each round, each client performs $E = 10$ epochs of local model updates via mini-batch SGD with a batch size of $B = 100$ for MNIST and Fashion-MNIST and $B = 10$ for HAR. Other hyperparameters during model training are inherited from the default settings of Adam (Kingma and Ba 2014).

Adversary Parameters We construct a rigorous adversary scenario by configuring 3 parameters, i.e., *Byzantine percentage* (the ratio of Byzantine clients over all the clients), *backdoor sample percentage* (the ratio of backdoor samples in a mini-batch, note that this parameter is used for targeted attacks only), and *non-IID degree* (the probability p for data samples assignment). Unless otherwise specified, our experiments are performed under the settings of 40% Byzantine clients, 8% backdoor samples, and $p = 0.8$.

Environment of Experiments We conduct experiments using PyTorch 1.5.1 on a machine with a TITAN RTX GPU, two 12-core 2.5GHz CPUs, and 148GB virtual RAM.

Evaluated Poisoning Attacks

We implement both untargeted poisoning attacks and targeted (backdoor) poisoning attacks. The former includes:

- **Label Flipping Attack**: For each sample, we flip its original label c to $c + 1 \mod M$, where M is the total number of labels, and $c \in \{1, 2, \dots, M\}$.
- **Gaussian Attack**: Byzantine clients submit poisoned local model updates generated through the unit-variance Gaussian distribution.
- **Back-gradient Attack** (Muñoz-González et al. 2017): Byzantine clients craft a poisoned dataset, on which the trained local model has a maximized loss on the benign dataset.
- **Krum Attack** (Fang et al. 2020): Byzantine clients craft poisoned local model updates opposite from benign ones, and enable them to circumvent the defense of `Krum`.

Targeted poisoning attacks include:

- **Badnet** (Gu, Dolan-Gavitt, and Garg 2017): Byzantine clients inject label-flipped data samples with specific backdoor triggers into the training datasets. Models trained on such poisoned datasets can be evoked targeted misclassification by samples with the same triggers. In FL, to avoid forgetting the backdoor, the poisoned local model updates are submitted in each round.
- **Backdoor FL** (Bagdasaryan et al. 2020): Byzantine clients generate backdoored local model updates and scale them to ensure that they could replace the benign global model when it is about to converge. This attack is only launched in the last round of FL.

Baseline Aggregation Rules

We employ the following aggregation rules as baselines.

`FedAvg` (McMahan et al. 2017): performing data size weighted local model aggregation of all clients.

`MultiKrum` (Blanchard et al. 2017): selecting $N - F$ local model updates with the smallest Euclidean distances to the majority, and averaging them to update the global model.

`Loss Function-based Rejection (LFR)` (Fang et al. 2020): selecting $N - F$ local model updates with the largest loss decrease on the centralized clean validation dataset, and averaging them to update the global model.

`FLTrust` (Cao et al. 2020): calculating a trust score for each local model update based on its deviation from the server model update trained on a clean small training dataset (called root dataset), and computing the average of local model updates weighted by their trust scores to update the global model.

Performance Metrics

- **Accuracy**: To evaluate the performance in defending against untargeted poisoning attacks, we employ the global model's test accuracy on benign testing data.

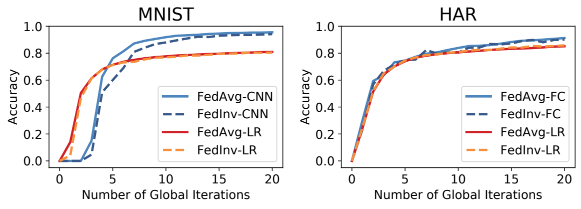

Figure 2: Accuracies without Byzantine attacks.

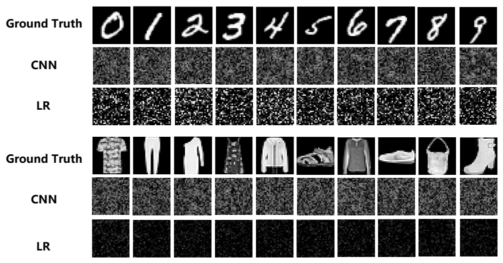

Figure 3: Comparisons between the dummy and original data samples.

- **Attack Success Rate (ASR)**: To evaluate the performance in defending against targeted attacks, besides the accuracy metric, we also test the global model's accuracy on testing data with backdoor triggers. The lower ASR, the better performance of the global model.

Convergence Performance of FedInv

We first consider an ideal case where all 20 clients are benign. Setting the parameter of non-IID degree as $p = 0.8$, we show the test accuracies of the global models trained via FedInv and FedAvg in Figure 2. Obviously, we can see that the global models trained via FedInv converge as nicely as FedAvg when no Byzantine attacks happen.

Privacy-Respecting Property of FedInv

We now show the desirable privacy-respecting property of FedInv. In a specific round of FedInv, we conduct privacy-preserving model inversion on a particular client's local model update (the mini-batch size is 10), which involves 700 steps of gradient descent on the randomly initialized dummy dataset for CNN and LR. We show the finally generated dummy data (images) in Figure 3. Compared to the original images, we cannot infer any private information from the synthesized images, which validates the privacy-respecting property of the model inversion in FedInv. We would like to note that this property applies to each round of FedInv.

Defense against Untargeted Attacks

We show the accuracies of various global models (e.g., CNN, FCN, LR) trained on different datasets using different FL schemes in Table **??**. The results demonstrate that our proposed FedInv consistently achieves better performance (i.e., higher accuracies of trained global models) compared to the baseline schemes.

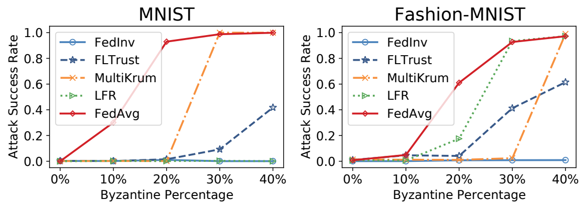

Figure 4: ASR versus Byzantine percentage.

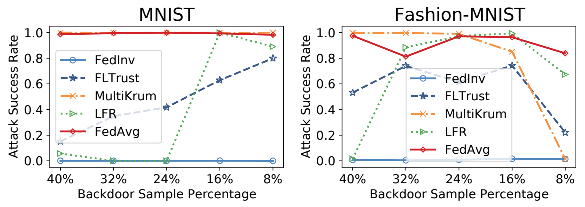

Figure 5: ASR versus backdoor sample percentage.

Defense against Targeted Attacks

We present the accuracies and ASR of various global models trained on different datasets using different FL schemes in Table **??**. The results show that our proposed FedInv consistently achieves higher accuracies of trained global models on benign testing data and at the same time maintains lower ASR on testing data with backdoor triggers, compared to the baseline methods.

Impact of Adversary Parameters

To further investigate the performance of FedInv under various adversary parameters, we take the Badnet attack as an example, and present the comparison between FedInv and the baseline methods under different Byzantine percentages, backdoor sample percentages, and non-IID degrees. Note that in this set of experiments, the default settings of the adversary parameters are 40% Byzantine clients, 24% backdoor samples, and 0.4 non-IID degree.

ASR versus Byzantine Percentage We show the ASR of the five FL schemes in Figure 4, where the value of Byzantine percentage varies from 0% to 40%. As shown in Figure 4, the ASR of the baseline FL schemes shows a clear upward trend, which reveals their ineffectiveness in identifying and excluding a relatively large number of Byzantine clients. In contrast, FedInv remains a low ASR even when 40% of clients are compromised, demonstrating its superior performance in rigorous adversarial settings in terms of many malicious clients.

ASR versus Backdoor Sample Percentage We show the ASR of the five FL schemes in Figure 5, where the backdoor sample percentage at Byzantine clients ranges from 40% to 8%. We can conclude that when the backdoor sample percentage decreases, the resulting poisoned local model updates are more challenging to detect for the baseline schemes, which is validated by their increasing ASR. FedInv keeps the ASR at a low level even though the backdoored

		CNN	FC	LR				CNN	FC	LR	
		MNIST	HAR	MNIST	HAR			MNIST	HAR	MNIST	HAR
Gaussian	FLTrust	93.47%	83.03%	77.58%	79.23%	Backgradient	FLTrust	92.34%	81.43%	77.23%	77.84%
	MultiKrum	94.00%	88.53%	79.28%	81.94%		MultiKrum	93.65%	19.20%	66.75%	46.42%
	LFR	94.12%	88.49%	79.41%	82.99%		LFR	93.78%	89.61%	77.85%	82.18%
	FedAvg	0.00%	12.28%	41.31%	23.37%		FedAvg	83.2%	78.00%	76.67%	73.97%
	FedInv	94.11%	89.63%	79.24%	82.14%		FedInv	94.21%	89.74%	79.17%	82.14%
Label Flip	FLTrust	90.34%	83.50%	77.32%	79.43%	Krum	FLTrust	93.00%	80.11%	76.80%	72.65%
	MultiKrum	68.00%	87.78%	29.15%	14.86%		MultiKrum	0.09%	0.00%	9.39%	0.00%
	LFR	94.42%	88.49%	79.47%	82.96%		LFR	93.07%	90.90%	79.38%	82.96%
	FedAvg	16.34%	49.94%	38.41%	29.58%		FedAvg	0.00%	0.01%	11.39%	0.00%
	FedInv	93.47%	89.52%	79.31%	82.05%		FedInv	94.00%	90.67%	79.21%	82.77%

Table 1: Accuracies under Untargeted Attacks

		MNIST	Fashion
Badnet	FLTrust	94.25%/87.17%	76.05%/55.34%
	MultiKrum	93.37%/99.84%	78.74%/1.34%
	LFR	94.06%/93.23%	80.13%/88.64%
	FedAvg	95.08%/94.94%	76.02%/85.53%
	FedInv	96.02%/0.07%	80.88%/0.92%
Backdoor FL	FLTrust	90.68%/1.67%	75.87%/3.13%
	MultiKrum	53.50%/47.25%	78.84%/0.94%
	LFR	94.39%/0.06%	78.23%/0.65%
	FedAvg	93.37%/91.14%	78.12%/78.31%
	FedInv	94.97%/0.04%	79.02%/0.62%

Table 2: Accuracies/ASR under Targeted Attacks

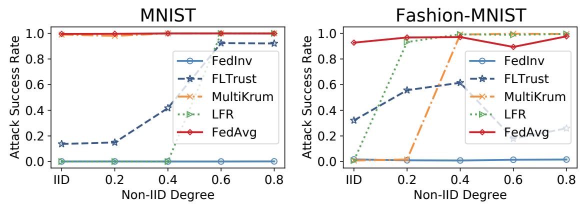

Figure 6: ASR versus non-IID degree.

samples constitute a minor proportion. Note that when training the CNN on Fashion-MNIST (i.e., right of Figure 5), a universe decrease of ASR is witnessed for all schemes. This is because the number of backdoored samples, in this case, is too small, leading to an unsuccessful backdoor attack.

ASR versus Non-IID Degree We show the relationship between the ASR of the five FL schemes and the non-IID degree (indicated by the probability value p) in Figure 6. The results demonstrate that for the baseline methods, the ASR rises with a higher degree of non-IID. The reason is that non-IID data distribution could increase the divergences between benign local model updates. Thus, it becomes more difficult to distinguish whether the outlying local model updates are caused by Byzantine attacks or non-IID data distribution. Surprisingly, the ASR under FedInv remains low, even under $p = 0.8$-non-IID. In all cases, our proposed FedInv consistently achieves a lower ASR than the baselines.

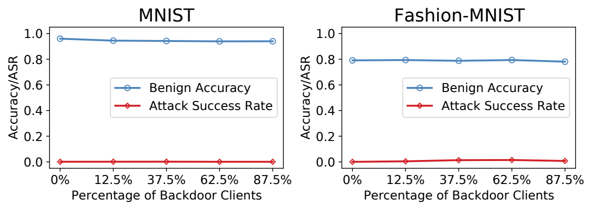

Figure 7: Accuracies and ASR of FedInv under colluding attacks.

Defense against Colluding Attacks

We finally examine whether FedInv could resist the collusion of Byzantine clients. We design a colluding attack that aims to ensure some poisoned local model updates can survive (i.e., enabling them to sneak into the majority group of FedInv and be aggregated via Badnet attack with 8% backdoor sample percentage), by sacrificing the remaining (i.e., making them easy to be identified and excluded via Gaussian attack). We present the accuracies and ASR of FedInv versus the Byzantine percentage in Figure 7, which shows that FedInv behaves well even when Byzantine clients collude to launch poisoning attacks.

Conclusion

This paper presented FedInv, a novel Byzantine-robust FL framework via privacy-respecting inversion on clients' local model updates. Specifically, FedInv first performs privacy-respecting model inversion on each client's local model updates to synthesize a corresponding dummy dataset before updating the global model in each round. Then, the Wasserstein distances of each dummy dataset to the remaining ones are calculated. The clients associated with exceptional Wasserstein distances are determined as Byzantine clients, whose local model updates are excluded from local model aggregation. We conduct an exhaustive experimental evaluation of FedInv. The results demonstrate that FedInv achieves superior performance in defending against various poisoning attacks than the existing robust FL schemes, especially when a relatively large proportion of clients are compromised and the data distribution among clients is highly non-IID.

Acknowledgements

We thank anonymous reviewers for their constructive comments to this work. This work was supported in part by the National Key R&D Program of China under Grant 2021YFB3100700, in part by the National Natural Science Foundation of China under Grant 62102337, in part by the Project funded by China Postdoctoral Science Foundation under Grant 2021M703079, and in part by the funding from Shenzhen Institute of Artificial Intelligence and Robotics for Society under Grant AC01202101032.

References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2013. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, 2938–2948. PMLR.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. 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.

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

Cao, X.; Fang, M.; Liu, J.; and Gong, N. Z. 2020. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *arXiv preprint arXiv:2012.13995*.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR*, abs/1712.05526.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. 2020. Local model poisoning attacks to byzantine-robust federated learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 1605–1622.

Fang, M.; Yang, G.; Gong, N. Z.; and Liu, J. 2018. Poisoning Attacks to Graph-Based Recommender Systems. In *Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018*, 381–392. ACM.

Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR*, abs/1708.06733.

Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C.; and Li, B. 2018. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, 19–35. IEEE Computer Society.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

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

LeCun, Y.; Cortes, C.; and Burges, C. 1998. MNIST handwritten digit database, 1998. *URL http://www. research. att. com/~ yann/ocr/mnist*.

Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data poisoning attacks on factorization-based collaborative filtering. *Advances in neural information processing systems*, 29: 1885–1893.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A.; and Zhu, X. J., eds., *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282. PMLR.

Mhamdi, E. M. E.; Guerraoui, R.; and Rouault, S. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 3518–3527. PMLR.

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

Prakash, S.; and Avestimehr, A. S. 2020. Mitigating byzantine attacks in federated learning. *arXiv preprint arXiv:2010.07541*.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 6106–6116.

Shejwalkar, V.; and Houmansadr, A. 2021. Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning. *Internet Society*, 18.

Weng, L. 2019. From gan to wgan. *arXiv preprint arXiv:1904.08994*.

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

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

Xie, C.; Koyejo, S.; and Gupta, I. 2020. Zeno++: Robust fully asynchronous SGD. In *International Conference on Machine Learning*, 10495–10503. PMLR.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 5650–5659. PMLR.

Zhu, L.; and Han, S. 2020. Deep leakage from gradients. In *Federated learning*, 17–31. Springer.