Romoa: Robust Model Aggregation for the Resistance of Federated Learning to Model Poisoning Attacks

Yunlong Mao(✉), Xinyu Yuan, Xinyang Zhao, and Sheng Zhong

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
maoyl@nju.edu.cn

Abstract. Training a deep neural network requires substantial data and intensive computing resources. Unaffordable price holds back many potential applications of deep learning. Besides, it is risky to gather user's private data for training centrally. Then federated learning appears as a promising solution to having users learned jointly while keeping training data local. However, security issues keep coming up in federated learning applications. One of the most threatening attacks is the model poisoning attack which can manipulate the inference result of a jointly learned model. Some recent studies show that elaborate model poisoning approaches can even breach the existing Byzantine-robust federated learning solutions. Hence, it is critical to discuss alternative solutions to secure federated learning. In this paper, we propose to protect federated learning against model poisoning attacks by introducing a robust model aggregation solution named Romoa. Unlike previous studies, Romoa can deal with targeted and untargeted poisoning attacks with a unified approach. Moreover, Romoa achieves more precise attack detection and better fairness for federated learning participants by constructing a new similarity measurement. We conclude that through a comprehensive evaluation of standard datasets, Romoa can provide a satisfying defense effect against model poisoning attacks, including those attacks breaching Byzantine-robust federated learning solutions.

Keywords: Model poisoning attack · Robust model aggregation · Federated learning

1 Introduction

The breakthrough of deep neural networks (DNNs) largely depends on substantial training data. As the data volume grows rapidly, it becomes inefficient to gather the Internet end users' data to a central server. Meanwhile, explicitly collecting users' private data may lead to various privacy threats [30]. At this point, the federated learning (FL) concept [21,28] emerges to meet the demands

E. Bertino et al. (Eds.): ESORICS 2021, LNCS 12972, pp. 476–496, 2021.
https://doi.org/10.1007/978-3-030-88418-5_23

of learning models collaboratively. FL is a promising concept to achieve collaborative learning with participants' private data kept locally. However, training DNNs collaboratively creates a new attack surface in the FL setting [23].

Among various attacks recently identified in FL, the poisoning attack is one of the most threatening attacks. The adversary of poisoning attacks is capable of manipulating the collaboratively learned model to classify input incorrectly. The input can be a specifically targeted sample or an arbitrary sample depending on the poisoning attack is targeted [8] or untargeted [9]. According to the poisoning approach, poisoning attacks can be categorized into data poisoning [8,12,26,31] and model poisoning [1,3,9]. Many efforts have been made to eliminate the adversarial effect of data poisoning attacks. But solutions to defeating model poisoning attacks are still under discussion. Several robust model aggregation methods [15,36] have been proved effective when dealing with model poisoning attacks. However, recent studies [3,9] have shown that Byzantine-robust aggregation solutions are still vulnerable to model poisoning attacks if these solutions are not integrated into FL properly.

Meanwhile, the existing robust model aggregation solutions [9,15,36] largely depend on the member selection for model aggregation, which only allows a small fraction of the participants (sometimes just one participant) has the right to contribute to the global model each time. This strategy causes other participants' training efforts to be wasted. In this way, the individual fairness [16,35] of all participants will be at risk. Meanwhile, new maneuvers of model poisoning attacks keep coming up. The existing defense studies cannot catch up with the rapid evolution of these attacks. Hence, it is essential to integrate an effective defense scheme into FL, preventing potential model poisoning attacks.

But designing a proper defense solution for FL is quite challenging. The first question is how to identify model poisoning attacks precisely. Since deep learning uses some techniques with randomness in the training process, like stochastic gradient descent (SGD) optimizer and dropout operation, it is difficult to distinguish the attack from normal training fluctuations, especially when participants' data are not independent and identically distributed (non-i.i.d.). Additionally, the adversary could conceal the attack by lowering the degree of model manipulation or performing the attack opportunistically [3]. If we use some rigorous detecting rules, then false alarms are unavoidable. However, if we use some loose detecting rules, the adversarial participant will be missed. Furthermore, handling suspicious participants is another tricky problem. Using a subset of participants for aggregation can avoid the adversarial effect. But a sizable portion of learned knowledge will be discarded. In the end, some participants cannot make fair contributions to the jointly learned model.

To defend FL against model poisoning attacks and solve these problems at the same time, we propose an alternative solution for robust model aggregation, named Romoa. The basic idea of Romoa comes from two observations of model poisoning attacks: ① the global learning convergence will be slowed down by model poisoning attacks. In most cases, when the adversary tries to poison the global model, significant fluctuations will occur on the learning curve. The global model needs more training iterations to overcome these fluctuations introduced by the poison. ② model poisoning attacks cannot be accomplished at once.

To maintain the adversarial effect on the global model, the adversary needs to interfere with the learning procedure frequently. Romoa uses a hybrid method based on several similarity measurements and a lookahead strategy. Although related work [1,11] has used distance or similarity metrics to detect poisoning attacks, letting a similarity measurement work well with FL together is still under discussion. Romoa gives the first attempt to use hybrid similarity measurements in a lookahead way for identifying adversarial behaviors precisely and timely. Then we design a sanitizing factor in Romoa to eliminate the poisoning effect in FL model aggregation. For the concerns of FL fairness, Romoa calculates the sanitizing factors with momentum, which will lead to heavy punishment to significantly adversarial participants while honest participants can retain voting rights even if there are false alarms. In summary, we make following contributions in this paper:

- To protect FL against model poisoning attacks, we propose Romoa, a robust model aggregation solution. In Romoa, we design a hybrid similarity measurement method to identify the attacks precisely. Meanwhile, Romoa can ensure defensive effectiveness and individual fairness of FL simultaneously.
- We propose an alternative approach for the security analysis of FL by formalizing model poisoning attacks into a repeated game. Then we give a detailed analysis of Romoa in a game manner. The analysis result shows that the robustness of Romoa can be ensured based on Nash equilibrium.
- We evaluate Romoa with two standard datasets in image classification tasks. Meanwhile, we compare our solution with a well-known Byzantine-robust solution in the same attack settings. The experimental results verify the effectiveness of Romoa even in both solo and collusive attacks.

2 Problem Statement

2.1 Federated Learning

In federated learning (FL) [21,28], a central parameter server (PS) will coordinate n participants who join the same FL task such as image classification. For simplicity, we assume that each participant $P_i, i \in [1, n]$ has private training data $\boldsymbol{x}^i$ held by P_i only and all participants share the same DNN architecture and learning hyper-parameters. Generally, a mini-batch SGD optimizer is used by P_i to minimize a loss function $\mathcal{L}(\boldsymbol{\theta}^i)$ for model parameters $\boldsymbol{\theta}^i$. To update the local model, the gradient $\nabla_{\boldsymbol{\theta}^i}$ of $\boldsymbol{\theta}^i$ should be estimated as

$$\boldsymbol{g}^i(\boldsymbol{\theta}^i) = \frac{1}{m} \sum\nolimits_{j=1}^{m} \nabla_{\boldsymbol{\theta}^i} \mathcal{L}(\boldsymbol{\theta}^i, x_j),\ x_j \in \boldsymbol{x}^i. \tag{1}$$

A globally shared iteration counter $t \in [1, T]$ should be maintained by the PS. Given P_i's local gradient $\boldsymbol{g}^i_{t-1}$, P_i's model parameter $\boldsymbol{\theta}^i$ for the next iteration should be updated by $\boldsymbol{\theta}^i_t = \boldsymbol{\theta}^i_{t-1} - \eta \boldsymbol{g}^i$, where t indicates training iteration and η is a predefined learning rate. After the local training of all participants has been done, the PS will perform model aggregation by following a predefined strategy

such as averaging. In this way, the PS gives the model aggregation result as $\bar{\boldsymbol{\theta}}_t = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\theta}_t^i$. At the beginning of the $(t+1)$-th training iteration, all participants download the latest global model $\bar{\boldsymbol{\theta}}_t$ from the PS and update local models. After synchronizing with the PS, the above procedure should be repeated until the global model has achieved an expected usability or the maximum training iteration limit.

2.2 Model Poisoning Attack

Generally, all participants in a FL task can only exchange knowledge through a trusted PS[1]. Any legal participant can be an adversary who wants to poison the jointly learned model. It has been proved that collusive attacks can promote poisoning attacks significantly [9,11,31]. For collusive poisoning, we assume that the total amount of adversarial participants should be less than $\lceil n/2 \rceil$. Moreover, the adversary has stealth capability to avoid detection schemes. We note there are several ways for the adversary to attack stealthily. Here we will consider a general approach where the adversary mitigates the adversarial effect by reducing poison dosage. This approach can be characterized by a stealth factor for model poisoning attacks.

Adversarial Goal. The adversarial goal is not to destroy the FL framework. Instead, the adversarial goal is to corrupt the jointly learned model to behave abnormally. According to recent studies, there are two main categories of the adversarial goal, *targeted poisoning attack* [3] and *untargeted poisoning attack* [9]. The adversary of an untargeted poisoning attack aims to cause the misclassification of all input samples indiscriminately while the targeted poisoning adversary aims to cause the misclassification of specific (targeted) input samples. The existing defense studies commonly discuss two adversarial goals separately [1,9]. But we will take into account them at the same time and give a unified solution.

Untargeted Poisoning Attack (UPA). The most powerful UPA yet is established by Fang et al. in [9]. By replacing the local model with a compromised one, UPA can breach several Byzantine-robust solutions. Specifically, the adversary in UPA is to solve an optimizing problem for finding the opposite direction of model updating. This optimizing problem can be customized regarding a specific Byzantine-robust solution. Taking one adversary P_a as an example, the objective function of UPA[2] is

$$\begin{aligned} &\mathcal{O}^{\mathrm{UPA}} = \arg\max_{\boldsymbol{\theta}_a} \boldsymbol{s}^T(\bar{\boldsymbol{\theta}} - \bar{\boldsymbol{\theta}}_{\mathrm{adv}}), \\ &\text{subject to } \bar{\boldsymbol{\theta}} = \sum\nolimits_{i=1}^{n}\boldsymbol{\theta}_i, \bar{\boldsymbol{\theta}}_{\mathrm{adv}} = \boldsymbol{\theta}_{\mathrm{a}} + \sum\nolimits_{i=1, i\neq a}^{n}\boldsymbol{\theta}_i, \end{aligned} \tag{2}$$

[1] This assumption is reasonable since there are many effective solutions [2,5,17] that can protect participants from an untrusted central server. Discussion of an untrusted PS needs an exclusive study.

[2] For more details of this approach, please refer to [9]. We will use UPA as a general notation in this paper.

where $\boldsymbol{s}^T$ is a vector of the changing directions of the global model from the before-attack state $\bar{\boldsymbol{\theta}}$ to the after-attack state $\bar{\boldsymbol{\theta}}_{\text{adv}}$.

Targeted Poisoning Attack (TPA). Targeted poisoning attacks have been widely studied recently [1,3,29]. Different from UPA, TPA has specific interests in some data samples. Assume these samples all in one set $\boldsymbol{x}_{\text{TPA}} = \{x_1, x_2, \ldots, x_r\}$. Given the corresponding labels $\boldsymbol{y}_{\text{TPA}} = \{y_1, y_2, \ldots, y_r\}$, the adversary aims to have each sample $x_i \in \boldsymbol{x}_{\text{TPA}}$ misclassified as label y_i' after poisoning the global model, $y_i' \neq y_i$. Then the TPA objective function for the adversary P_a is

$$\begin{aligned} &\mathcal{O}^{\text{TPA}} = \arg\min_{\boldsymbol{\theta}_a} \mathcal{L}(\{x_i, y_i'\}_{i=1}^r, \bar{\boldsymbol{\theta}}_{\text{adv}}), \\ &\text{subject to } \bar{\boldsymbol{\theta}}_{\text{adv}} = \boldsymbol{\theta}_a + \sum\nolimits_{i=1, i\neq a}^{n} \boldsymbol{\theta}_i, \end{aligned} \tag{3}$$

where $\mathcal{L}(\cdot)$ is the loss function used in P_a's local training. We remark that a boost factor in original TPA [1,3] is omitted here, which will be combined with our stealth factor.

Stealth Factor. In the existing studies, the concealment of poisoning attacks is barely discussed because this will mitigate adversarial effect significantly. Previous work [3] uses a boost factor to enlarge the adversarial effect, but we find it also helpful for adversary's stealth. Generally, we define the adversary's goal in the t-th iteration as $\mathcal{A}_t = \boldsymbol{\theta}_{t-1}^a - \alpha_t(\boldsymbol{\theta}_{t-1}^a - \mathcal{O}_t^b)$, $t \in [1, T]$, $\alpha_t \in [0.0, 1.0]$, subjecting to the corresponding constraint. Specifically, when $\alpha_t = 1$, the adversary's goal $\mathcal{A}_t$ is to replace the local model with a poisoning model completely. When $\alpha_t = 0$, the adversary chooses not to attack this time. In other cases, $\mathcal{A}_t$ can be seen as a mixture of the global model and local poison model since $\mathcal{A}_t$ can be written as $(1 - \alpha_t)\boldsymbol{\theta}_{t-1}^a + \alpha_t \mathcal{O}_t^b$ equivalently.

Attack Evaluation. To demonstrate the effect of model poisoning attacks, we use the following experimental setting by default. A *baseline* FL task consists of a central PS and 10 participants. Two standard datasets and corresponding DNNs[3] are used, i.e., MNIST [19] and CIFAR-10 [18]. By default, we assume training datasets for participants are independent and identically distributed (i.i.d.) while a non-i.i.d. case will be discussed further. In each baseline task, a cross-entropy loss function and a SGD optimizer are used while the learning rate and the batch size are 0.001 and 64. These is one adversary (9 benign participants) using the same training setting as others by default, mounting UPA or TPA in the task. Two key metrics are commonly used to evaluate the effect of model poisoning attacks, model *accuracy* (for UPA and TPA) and target label *confidence* (for TPA only). Evaluation results for baseline tasks are shown in Fig. 1 and 2, which clearly demonstrate the effect of UPA and TPA in baseline tasks.

We will use *error rate* as a unified metric later for both TPA and UPA to measure the defense performance. The lower the error rate, the greater probability the adversary fails and the better the defense performance is. As for TPA, the error rate equals the attack confidence of the target class. The confidence score

[3] Detailed information of DNN architectures we used is given in the appendix.

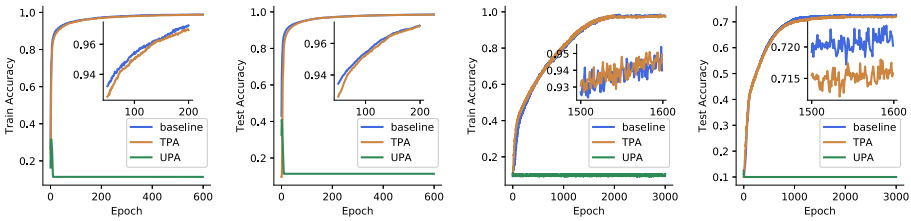

Fig. 1. Model accuracy compromised by model poisoning attacks with MNIST (left two) and CIFAR-10 (right two) datasets (one adversary).

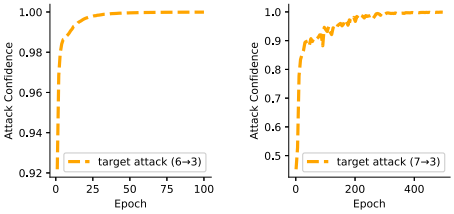

Fig. 2. Confidence of TPA with MNIST (left) and CIFAR-10 (right) datasets (one adversary). The TPA adversary intends to flip label 6 (number 6) to label 3 (number 3) in MNIST and label 7 (frog) to label 3 (bird) in CIFAR-10.

can be calculated by the probability of a sample being classified to a poisoned label. The error rate of UPA is calculated by one minus average accuracy across all input samples.

3 Robust Model Aggregation

The design of Romoa is inspired by some experimental observations of model poisoning attacks. In a compromised FL task, the learning procedure will be interfered by the adversary as long as the poison takes effect, whether in a stealthy manner or not. The interference can be observed from two aspects: notably extra training iterations for the global convergence and more unexpected fluctuations on the global learning curve. Given these abnormal appearances, it is still challenging to identify the adversary from normal FL participants especially when randomness and non-i.i.d. datasets are used. Recent studies [4,8,11,31] have investigated the feasibility of similarity measurement based solutions. These solutions are effective but the defensive effect is thwarted when dealing with attacks which are designed against Byzantine-robust solutions [3,9,15,31,32].

To tackle this problem, we propose a novel similarity measurement by combining hybrid similarity measurements with a lookahead strategy. On the basis of this lookahead similarity measurement, Romoa can identify the adversary precisely with negligible interference to training. After quantifying the divergence between participants, Romoa will assign a sanitizing factor to each participant.

The sanitizing factor is constructed based on temporal similarity measurement result and historical behaviors of each participant. Then local model parameters will be sanitized by corresponding factors during the model aggregation. Different from the existing solutions, Romoa uses a lookahead strategy to capture potential threats and no labors of any participant will be dropped simply. This feature provides FL with both robustness and individual fairness. Now, we will introduce Romoa in a constructing order.

3.1 Asynchronous Model Updating

Asynchronous parameter updating schemes (asynchronous updating for short) are effective in specific learning cases [34,40]. In asynchronous updating, the PS performs model aggregation every τ iterations to sync model states of all FL participants. For the period between two adjacent syncing points, participants are allowed to explore model states locally [33,34,40]. Apart from this, the syncing operation is the same as the original FL. Briefly, we give a general asynchronous updating in Algorithm 1, which will be the basis of Romoa.

Algorithm 1: Asynchronous updating algorithm.

Input : learning rate η, amount of participants n, moving rate γ, syncing period τ, maximal iteration T.
Output: globally learnt model $\bar{\boldsymbol{\theta}}$.

for $i \leftarrow 1$ **to** n **do**
  $\boldsymbol{\theta}_0^i \leftarrow rand(0,1)$ `/* initialization */`
end

Participant P_i:
for $i \leftarrow 1$ **to** n **do**
  for $t \leftarrow 1$ **to** T **do**
    $\boldsymbol{\theta}_t^i \leftarrow \boldsymbol{\theta}_{t-1}^i - \eta g_t^i(\boldsymbol{\theta}_{t-1}^i)$ `/* local training */`
    if τ *divides* t **then**
      upload $\boldsymbol{\theta}_t^i$
      download $\bar{\boldsymbol{\theta}}_t$ `/* syncing */`
      $\boldsymbol{\theta}_t^i \leftarrow \boldsymbol{\theta}_t^i - \gamma(\boldsymbol{\theta}_t^i - \bar{\boldsymbol{\theta}}_t)$
    end
  end
end

Parameter Server:
for $t \leftarrow 1$ **to** T **do**
  if τ *divides* t **then**
    $\bar{\boldsymbol{\theta}}_t \leftarrow \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\theta}_t^i$ `/* averaging aggregation */`
  end
end

3.2 Lookahead Similarity Measurement

Previous studies have discussed the possibility of using Euclidean distance or cosine similarity to measure the differences of DNN models between FL participants [1,4,11]. However, we find that simply calculating distance or similarity is not sufficient. Different from previous work, we design a novel similarity measurement method by using a lookahead strategy. The original lookahead strategy proposed in [39] is an alternative optimizer for improving the learning stability. We will use the lookahead strategy in a different way. In asynchronous updating, all participants are allowed to explore locally between two adjacent syncing points. We will take advantage of this feature and let the PS monitor the exploration stage. Then the PS can be aware of poisons generated in local exploration ahead of aggregation.

Assuming the whole asynchronous updating process can be divided into numerous periods, $T = K\tau, K \in \mathbb{N}$. All participants are required to upload local models during exploration. If t' counts continuously from the last syncing point t, P_i will perform τ local training iterations and upload $\boldsymbol{\theta}_{t'}^i$ to the PS for lookahead similarity measurement before the next syncing point, i.e., $t' \in [k\tau + 1, (k+1)\tau]$. But only local model parameters in the $((k+1)\tau)$-th iteration will be used for synchronization. After collecting all local models, the PS should perform parameter selection first. Specifically, parameters with high absolute values will be selected at the ratio of γ (generally assuming $\gamma = \frac{1}{n}$ if no further explanation is given). The selection result of $\boldsymbol{\theta}_{t'}^i$ is denoted by $\tilde{\boldsymbol{\theta}}_{t'}^i$, $|\tilde{\boldsymbol{\theta}}_{t'}^i| = \gamma|\boldsymbol{\theta}_{t'}^i|$. Let $[\boldsymbol{\theta}_{t'}^{j,w}]$ denote the index of w-th parameter of $\boldsymbol{\theta}_{t'}^j$ and $\left\{[\boldsymbol{\theta}_{t'}^{j,w}]\right\}$ as the corresponding index set. Finally, merge all participants' parameter selection results:

$$\hat{\boldsymbol{\theta}}_{t'}^i = \tilde{\boldsymbol{\theta}}_{t'}^i \cup \left\{\boldsymbol{\theta}_{t'}^j \middle| [\boldsymbol{\theta}_{t'}^{j,w}] \in \left\{[\tilde{\boldsymbol{\theta}}_{t'}^{j,w}]\right\}, j \in [1,n], i \neq j\right\}. \tag{4}$$

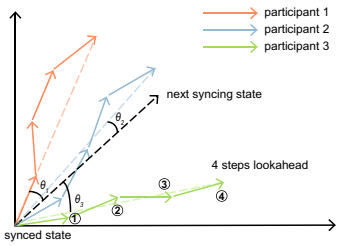

Fig. 3. Cosine similarity using a lookahead strategy ($\tau = 4$).

In the t'-th iteration, the PS calculates a lookahead aggregation of the selected parameters, $\bar{\boldsymbol{\theta}}_{t'} = \frac{1}{n}\sum_{i=1}^{n} \hat{\boldsymbol{\theta}}_{t'}^i$. If we treat the state of an expanded selection of parameters as a planar point, then we can illustrate the calculation of lookahead similarity in Fig. 3. The angle to be calculated is formed by

two edges. One is from the last syncing state (e.g., syncing point t) to the last lookahead state of a participant. The other one is from the last syncing state to the lookahead aggregation of all participants. Given these two updating paths both started with last synced state $\bar{\boldsymbol{\theta}}_t$, one ended with next syncing state $\bar{\boldsymbol{\theta}}_{t'}$, the other ended with participant P_i's selected parameters $\hat{\boldsymbol{\theta}}^i_{t'}$, we can define two non-zero vectors $[\bar{\boldsymbol{\theta}}^w_t, \bar{\boldsymbol{\theta}}^w_{t'}]$ and $[\bar{\boldsymbol{\theta}}^w_t, \hat{\boldsymbol{\theta}}^{i,w}_{t'}]$ (w denotes the index of parameters). Then element-wise cosine similarity measurement for any participant P_i is

$$S^{i,w}_{cosine} = \frac{(\hat{\boldsymbol{\theta}}^{i,w}_{t'} - \bar{\boldsymbol{\theta}}^w_t)(\bar{\boldsymbol{\theta}}^w_{t'} - \bar{\boldsymbol{\theta}}^w_t)^T}{(\sum_{\theta \in \{\hat{\boldsymbol{\theta}}^{i,w}_{t'} - \bar{\boldsymbol{\theta}}^w_t\}} \theta^2)^{(\frac{1}{2}} \times \sum_{\theta \in \{\bar{\boldsymbol{\theta}}^w_{t'} - \bar{\boldsymbol{\theta}}^w_t\}} \theta^2)^{\frac{1}{2}}}. \tag{5}$$

The above definition gives similarity measurement for parameters $\hat{\boldsymbol{\theta}}^i_{t'}$ which are selected according to absolute values. But the remaining unselected parameters also need to be measured properly. We use cosine similarity and Pearson correlation in a layer-wise way to capture divergences of the unselected parameters. If all parameters in the l-th layer of a DNN model is denoted by $\boldsymbol{\theta}^{i[l]}_{t'} \in \mathbb{R}^{M_l}$ (M_l is the total number of parameters in the l-th layer) and function $std(\cdot)$ yields standard deviation, then these two measurements are defined as

$$L^{i[l]}_{cosine} = \frac{(\boldsymbol{\theta}^{i[l]}_{t'} - \bar{\boldsymbol{\theta}}^{[l]}_t)(\bar{\boldsymbol{\theta}}^{[l]}_{t'} - \bar{\boldsymbol{\theta}}^{[l]}_t)^T}{(\sum_{\theta \in \{\boldsymbol{\theta}^{i[l]}_{t'} - \bar{\boldsymbol{\theta}}^{[l]}_t\}} \theta^2)^{\frac{1}{2}} \times \sum_{\theta \in \{\bar{\boldsymbol{\theta}}^{[l]}_{t'} - \bar{\boldsymbol{\theta}}^{[l]}_t\}} \theta^2)^{\frac{1}{2}}}, \tag{6}$$

$$L^{i[l]}_{pearson} = \frac{L^{i[l]}_{cosine}}{std(\{\boldsymbol{\theta}^{i[l]}_{t'} - \bar{\boldsymbol{\theta}}^{[l]}_t\}) \times std(\{\bar{\boldsymbol{\theta}}^{[l]}_{t'} - \bar{\boldsymbol{\theta}}^{[l]}_t\})}. \tag{7}$$

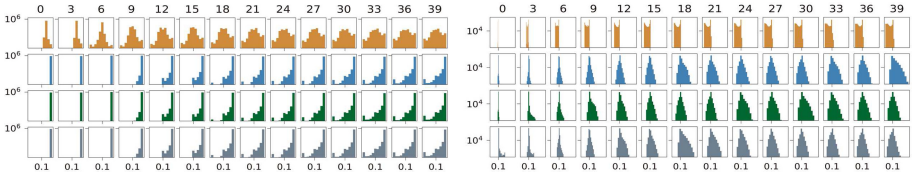

Fig. 4. Distributions of similarity measurement (left) and sanitizing factors (right) for 4 participants training with MNIST dataset (UPA adversary in the first row). X-axis is value of the results and Y-axis is the corresponding density. Indicator $0, 3, 6...$ is the number of epoch.

An example of the lookahead similarity measurement is given in Fig. 4. Obviously, the adversary (in the first row) has a totally different measurement distribution when compared with other honest participants.

3.3 Model Aggregation with Sanitizing Factor

To eliminate the adversarial effect of poisoning attacks while maintaining a unanimous model updating tendency, we introduce a sanitizing factor $\boldsymbol{F}$. The sanitizing factor is a weight vector for each parameter, which is constructed on the basis of lookahead similarity measurement results. Each parameter should be sanitized by a sanitizing factor when the PS performs model aggregation. In this manner, sharp fluctuations in the global learning process are supposed to be moderated. For converting similarity measurement results into sanitizing factors, we use a mean shift algorithm [10], which can estimate the density of model parameters and measurement results. Specifically, the mean shift algorithm takes the similarity measurement result as its input and yields clusters and corresponding centroids. For any $\theta_w \in \boldsymbol{\theta}$, if θ_w belongs to some cluster, then the centroid is denoted by c_w (the same centroid may be referred to as different identifiers). The generating function for element-wise sanitizing factors is

$$f^i_{S_{cosine}}(\theta_w) = \begin{cases} S^i_{cosine}(\theta_w) - c_w, \text{ if } \theta_w \text{ in } \hat{\boldsymbol{\theta}}^i, \\ 0, \text{ otherwise.} \end{cases} \tag{8}$$

Similarly, we can define two generating functions for layer-wise sanitizing factors. For θ_w in the l-th layer,

$$f^i_{L_{cosine}}(\theta_w) = L^{i[l]}_{cosine}(\theta_w) - c_w, \tag{9}$$

$$f^i_{L_{pearson}}(\theta_w) = L^{i[l]}_{pearson}(\theta_w) - c_w. \tag{10}$$

When three measurements results are combined, the minimum result is selected as a representative. Then sanitizing factor $\boldsymbol{F}^i_t$ for $\boldsymbol{\theta}^i$ can be defined as

$$\begin{aligned} \boldsymbol{F}^i_t &= \{\min\{f^i_{S_{cosine}}(\theta_w), f^i_{L_{cosine}}(\theta_w), f^i_{L_{pearson}}(\theta_w)\}\}_{\theta_w \in \boldsymbol{\theta}^i_t}, \\ \boldsymbol{F}^i_t &= \beta e^{\boldsymbol{F}^i_t} / \sum\nolimits_{j=1}^{n} e^{\boldsymbol{F}^j_t} + (1-\beta)\boldsymbol{F}^i_{t-1}, \end{aligned} \tag{11}$$

where β is a residual rate, accumulating $\boldsymbol{F}^i_t$ with its historical observations accumulatively ($\beta = 1/2$ if no further explanation is given). For the initialization, we set $\boldsymbol{F}^i_0 \leftarrow \frac{1}{n}$ since each participant is assumed to be honest from the very beginning. By integrating the sanitizing factors into parameter aggregation, we have Romoa as shown in Algorithm 2. Meanwhile, Fig. 4 shows distributions of sanitizing factors for the adversary and honest participants. It is shown that the adversary's sanitizing factors are totally different from others. And sanitizing factors will not affect honest participants since the historical records can prevent false positives.

4 Security Analysis

In this section, we will formalize model poisoning attacks and Romoa's defense into a game. Informally, FL can be seen as a finitely repeated game. We will first

construct a strategic game for one training iteration with potential adversaries in FL, named federated learning game (FLG). Then we show that all participants in FLG will choose to be honest or adversarial at the same time if no defense exists. Next, we extend FLG to a finitely repeated game, named repeated federated learning game (rFLG). Finally, we will show that Romoa is secure in rFLG if a Nash equilibrium can be achieved with all participants being honest (i.e., no attacks). We give the conclusion first and then show how to prove it.

Theorem 1. *FL with robust model aggregation (Romoa) is secure against model poisoning attack if the number of adversarial participants is less than* $\lceil n/2 \rceil$*, where* n *is the total number of FL participants.*

4.1 FLG: Federated Learning Game

The FLG is a strategic game, denoted by G, containing the interactions of all participants in each iteration. We assume that all participants are rational and should take actions simultaneously. The adversary can make the attack subtle or effective by controlling the poison dosage. All participants want to get a finally well-trained DNN model when the game ends. If the model functionality achieves higher than a threshold, then honest participants win. If an adversary gets a higher attack score, then the adversary wins. Furthermore, the adversary prefers to attack than learning honestly because the adversary is supposed to get extra revenues from a compromised model aggregation. Besides, the adversary can still win the game even if honest participants lose.

Each participant in a FL task is a natural player in FLG. Participant P_i has an action set A_i, which contains all available actions, $i \in [1, n]$. A utility function mapping an action set to a real-value utility score is $u_i : \boldsymbol{A} \leftarrow \mathbb{R}$. Please note that $u_i(\boldsymbol{a}) \geq u_i(\boldsymbol{a}')$ if and only if P_i has preference for action set $\boldsymbol{a}$ over action set $\boldsymbol{a}'$, where $\boldsymbol{a}$ and $\boldsymbol{a}' \in \boldsymbol{A}$. Now we can define FLG as a strategic game $G =< \mathcal{P}, \{A_i\}_{i=1}^n, \{u_i\}_{i=1}^n >$. The player set is denoted by $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$. For a general purpose of model poisoning attacks, we define available action set A_i as $\{q_0, q_1, q_2, \ldots, q_d\}$, where d is the maximal degree of the poison dosage. Specifically, the action q_0 indicates no poison while the rest actions $\{q_1, q_2, \ldots, q_d\}$ indicate the poison dosage increasing linearly (the player can choose any action by adjusting the stealth factor α). We use $|a_i|/d \in [0.0, 1.0]$ to represent the poison percentage of action a_i.

The utility function in FLG consists of two parts. The first part is information gain from model aggregation, denoted by $\frac{1}{n}\sum_{i=1}^n g(a_i)$, where $g(a_i) = 1 - |a_i|/d$ is a set-valued mapping. The second part is attack score, which is another set-valued mapping, denoted by $h(a_i) = |a_i|/d$. Then the corresponding utility function of P_i can be defined as

$$u_i(\boldsymbol{a}) = \frac{1}{n}\sum\nolimits_{j=1}^{n} g(a_j) + h(a_i), \tag{12}$$

where $a_i \in A_i$. We now give some intuitive interpretations about the utility function. Normally, P_i can get knowledge from other players through the model aggregation. But this knowledge will be hidden in a mixture of all participants'

Algorithm 2: FL with robust model aggregation (Romoa).

Input : learning rate η, amount of participants n, residual rate β, moving rate γ, syncing period τ, maximal iteration T.
Output: globally learnt model $\bar{\boldsymbol{\theta}}$.

for $i \leftarrow 1$ **to** n **do**
  $\boldsymbol{\theta}_0^i \leftarrow rand(0,1)$ /* initialization */
  $\boldsymbol{F}_0^i \leftarrow \frac{1}{n}$
end

Participant P_i:
for $i \leftarrow 1$ **to** n **do**
  for $t \leftarrow 1$ **to** T **do**
    $\boldsymbol{\theta}_t^i \leftarrow \boldsymbol{\theta}_{t-1}^i - \eta g_t^i(\boldsymbol{\theta}_{t-1}^i)$
    upload $\boldsymbol{\theta}_t^i$ /* lookahead updating */
    if τ *divides* t **then**
      download $\bar{\boldsymbol{\theta}}_t$ /* syncing */
      $\boldsymbol{\theta}_t^i \leftarrow \boldsymbol{\theta}_t^i - \gamma(\boldsymbol{\theta}_t^i - \bar{\boldsymbol{\theta}}_t)$
    end
  end
end

Parameter Server:
for $t \leftarrow 1$ **to** T **do**
  calculating $\boldsymbol{F}_t^i$ *for* P_i /* sanitizing factor */
  for $i \leftarrow 1$ **to** n **do** /* normalization */
    $\boldsymbol{F}_t^i \leftarrow \beta e^{\boldsymbol{F}_t^i} / \sum_{j=1}^n e^{\boldsymbol{F}_t^j} + (1-\beta)\boldsymbol{F}_{t-1}^i$
  end
  if τ *divides* t **then**
    $\bar{\boldsymbol{\theta}}_t \leftarrow \sum_{i=1}^n \boldsymbol{\theta}_t^i \boldsymbol{F}_t^i$ /* sanitized aggregation */
  end
end

information. Hence, the information gain from the aggregation should be scaled by $\frac{1}{n}$. Obviously, if all participants take normal action, then the total social welfare will equal to n while each player yields 1 utility.

We design the attack score carefully so that the adversary can get extra revenues for attack action while players' cooperation is still possible. Considering all possible outcomes, the adversary prefers to take the most effective action q_d if all the other participants act normally. In this case, the adversary can get $2 - \frac{1}{n}$ utility while other players get $1 - \frac{1}{n}$ utility. Since all players are rational, they will choose to take take the most effective poisoning action and end in 1 utility from the attack eventually, which also yield a total social welfare n.

4.2 rFLG: Repeated Federated Learning Game

Now we extend the FLG into a finitely repeated game rFLG to characterize players' interactions repeatedly for the recursive learning. Aiming at secure

aggregation, it is crucial to have undesirable behaviors punished. The sanitizing factors introduced in Romoa are designed exactly for this purpose. Given $G =< \mathcal{P}, \{A_i\}_{i=1}^n, \{u_i\}_{i=1}^n >$, rFLG can be defined as a finitely repeated game of G as $G_0 =< \mathcal{P}, H, S, \{u_i\}_{i=1}^n >$, where $\mathcal{P}$ and $\{u_i\}_{i=1}^n$ are the same player set and utility function set as G, $H = \{\Phi\} \cup \{\cup_{t=1}^T \boldsymbol{A}^t\}$ is the set of historical action profiles, Φ is the initial profile, T is a given positive integer, and $\boldsymbol{A} = \{A_i\}_{i=1}^n$. Additionally, S is the set of strategies for each player, which assigns an action in A_i to every finite sequence of action history. It should be noted that if $\boldsymbol{a}^t = (a^1, a^2, \ldots, a, \ldots, a^t)$, $a \in A_i$, $a' \in A_i$ and $u_i(a) \geq u_i(a')$, we will say that P_i has a preference for action sequence $(a^1, a^2, \ldots, a, \ldots, a^t)$ over action sequence $(a^1, a^2, \ldots, a', \ldots, a^t)$. To put Romoa into the rFLG, we make an abstraction of the sanitizing factors and use it to reconstruct original utility functions as

$$u_i^*(\boldsymbol{a}) = \sum\nolimits_{j=1}^{n} \frac{e_j}{n} g(a_j) + h(a_i), \tag{13}$$

where e_j can be seen as a predefined price of each player P_j charging for P_i's unsuitable behaviors. To have e_j worked in the same way as the sanitizing factors, we assume that e_j can be determined by the similarity between action profiles of P_i and P_j. Specifically,

$$e_j = \begin{cases} 1, \text{ if } g(a_i) \geq g(a_j), \\ g(a_i), \text{ otherwise.} \end{cases} \tag{14}$$

In this way, the adversary who takes attack action will be punished by other participants. Given e_j, we can derive another strategic game G^* from G. $G^* =< \mathcal{P}, \{A_i\}_{i=1}^n, \{u_i^*\}_{i=1}^n >$. Different from the original G, we can easily conclude that G^* has a unique Nash equilibrium where all players choose to take action q_0, which means no attacks. Then a FL task with Romoa can be defined as another rFLG $G_0^* =< \mathcal{P}, H, S, \{u_i^*\}_{i=1}^n >$. Furthermore, by following the theorem about Nash equilibrium of finitely repeated game, we can directly conclude that the outcome of the G_0^* consists of the Nash equilibrium of G^* repeated T times.

5 Evaluation

We evaluate Romoa in two aspects, defense capability and model usability. For the purpose of comparison, we use a FL task without any defense as a baseline where 9 participants use the default setting as previously introduced while one model poisoning adversary. We also compare Romoa with two well-known Byzantine-robust solutions Krum [4] and RFA [9] in the same settings[4] with participants scale from 10 to 200. All experimental results are averaged across multiple runs with MNIST [19] and CIFAR-10 [18] datasets respectively.

Recall that UPA aims to increase the error rate for all labels indiscriminately while TPA aims to increase the error rate only for the target samples.

[4] Since our FL setting is different from Krum and RFA, the results may vary slightly. But this does not hurt major conclusions.

Figure 5 shows error rates of UPA regarding different solutions. Note that both Romoa and RFA achieve nearly perfect defense on MNIST dataset but Romoa can outperform RFA on CIFAR-10 dataset. Meanwhile, UPA error rate reaches significantly high in the baseline in early stage and the performance of Krum shows that Byzantine-robust solution could be broken quickly. Defense results against TPA are shown in Fig. 6. All solutions are evaluated by confidences of true label and the poisoning label. True label confidence of the baseline is compromised quickly. But Romoa and Krum can provide strong protect in the same case. Note that Romoa has explicit advantage over RFA and outperforms Krum in poisoning label confidence.

As for the model usability evaluation, UPA and TPA may have different focuses. UPA influences global model accuracy seriously while TPA barely has influences on global model accuracy. Figure 7 and Fig. 8 are evaluation results of model accuracy in training and test regarding UPA and TPA respectively. As

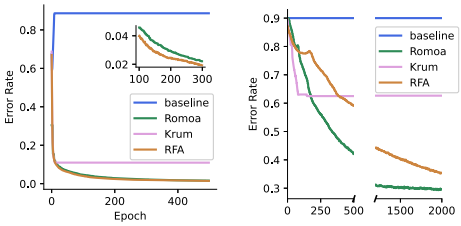

Fig. 5. Error rates of UPA with MNIST (left) and CIFAR-10 (right) datasets.

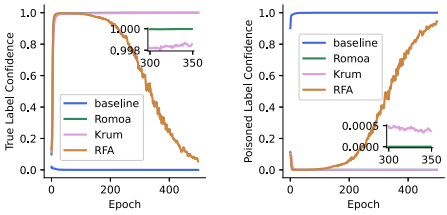

Fig. 6. Confidence of true label (left) and poisoned label (right) for the samples targeted by TPA with MNIST dataset.

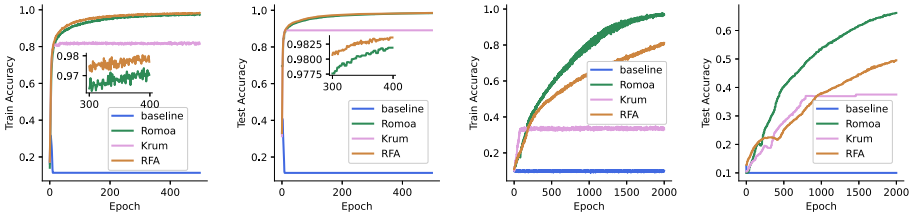

Fig. 7. Global model accuracy of the model attacked by UPA with MNIST (left two) and CIFAR-10 (right two).

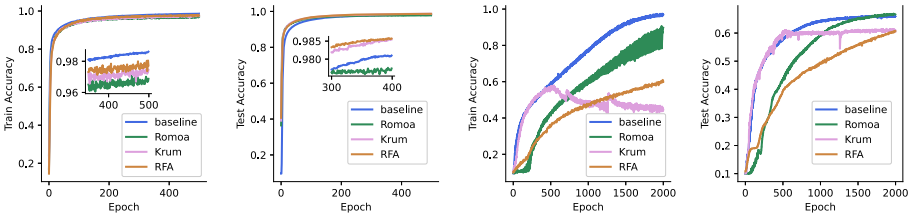

Fig. 8. Global model accuracy compromised by TPA with MNIST (left two) and CIFAR-10 (right two) datasets.

Table 1. Average error rate for FL tasks on different scales. Romoa can outperform Krum and RFA in most cases and achieve competitive in other cases.

Participants (n)			10			20		
Adversarial participants			1	3	4	1	6	9
MNIST	TPA	baseline	0.969517	0.944382	0.985758	0.973807	0.963074	0.990942
		Romoa	0.008289	**0.000183**	**0.000571**	0.021465	**0.013944**	**0.001301**
		RFA	**0.004690**	0.564633	0.687104	**0.009044**	0.61265	0.61265
		Krum	0.046238	0.062209	0.360937	0.058969	0.094315	0.461042
	UPA	baseline	0.884297	0.886125	0.886371	0.881747	0.886246	0.886527
		Romoa	**0.014700**	**0.043952**	**0.042849**	0.087651	**0.068632**	**0.108119**
		RFA	0.033200	0.105942	0.884133	**0.048608**	0.143554	0.886077
		Krum	0.111384	0.171478	0.898100	0.137760	0.160500	0.899100
CIFAR10	TPA	baseline	0.976524	0.993349	0.969406	0.942245	0.993748	0.988471
		Romoa	0.106586	0.116066	0.128794	0.105491	0.110650	0.19591
		RFA	0.101637	0.36399	0.713695	**0.009044**	0.171790	0.738469
		Krum	**0.100348**	**0.103137**	**0.100276**	0.099465	**0.099167**	**0.100321**
	UPA	baseline	0.899990	0.900000	0.900000	0.900000	0.900000	0.900000
		Romoa	**0.390807**	**0.473752**	**0.483586**	**0.497820**	**0.589919**	**0.675593**
		RFA	0.411122	0.899662	0.899919	0.527069	0.899814	0.900000
		Krum	0.631414	0.915900	0.899800	0.706112	0.881600	0.899300
Participants (n)			100			200		
Adversarial participants			1	33	49	1	66	99
MNIST	TPA	baseline	0.995910	0.995476	0.995090	0.995910	0.995476	0.995090
		Romoa	0.051435	0.136652	**0.152566**	0.108603	**0.189824**	**0.211517**
		RFA	**0.034294**	0.532786	0.988208	**0.091589**	0.682731	0.995961
		Krum	0.106764	**0.102102**	0.999547	0.131459	0.767544	0.987186
	UPA	baseline	0.867507	0.886444	0.886500	0.873398	0.886481	0.886500
		Romoa	**0.122615**	**0.189103**	**0.225616**	**0.188722**	**0.291795**	**0.292397**
		RFA	0.138354	0.885578	0.887790	0.195785	0.682731	0.995961
		Krum	0.296960	0.884200	0.931200	0.853001	0.875300	0.870700
CIFAR10	TPA	baseline	0.995910	0.995476	0.995090	0.995910	0.995476	0.995090
		Romoa	0.107185	0.117845	**0.126996**	0.117252	**0.112823**	**0.211517**
		RFA	0.102496	0.194651	0.914630	0.100821	0.234961	0.941820
		Krum	**0.101537**	**0.102102**	0.999547	**0.100606**	0.908100	0.902900
	UPA	baseline	0.899408	0.900000	0.900000	0.898762	0.900000	0.900000
		Romoa	**0.740108**	**0.834114**	**0.834114**	0.875135	**0.884951**	**0.888772**
		RFA	0.743340	0.899804	0.900000	**0.860587**	0.899931	0.900000
		Krum	0.900200	0.898500	0.908600	0.905888	0.911900	0.913200

Table 2. Error rate of UPA and TPA with non-i.i.d. datasets.

		No attack	TPA	UPA
MNIST	baseline	0.149003	0.880771	0.895005
	Romoa	0.133656	0.432999	0.16347
CIFAR10	baseline	0.460900	0.972004	0.900100
	Romoa	0.424200	0.738158	0.570603

we can see, Romoa can protect global model accuracy against UPA and TPA effectively. RFA can achieve the best performance on MNIST dataset but both RFA and Krum fail on CIFAR-10 dataset. We remark this failure may be caused by the neglect of training data diversity. We give more experimental results in Table 1, which takes into account more adversarial participants. Basically, we can conclude that Romoa can outperform Krum and RFA in most cases and Romoa has better performance when dealing with more adversarial participants.

To be more practical, we also evaluate Romoa with non-i.i.d. datasets. Intuitively, defense solutions based on similarity measurement methods cannot deal with participants' non-i.i.d. datasets. But Romoa can use hybrid similarity measurements in a lookahead manner to overcome this problem. The result in Table 2 is obtained by replacing the identical training data in default setting for 10 participants with assigning to each participant a distinct label and the corresponding data. Then the rest setting is the same as default with one adversary. In this case, we find Romoa can still frustrate UPA and TPA significantly.

6 Related Work

In recent years, the emergence of Federated learning [21,28] has attracted much attention and gives a new solution to better use the Internet end-users' big data. However, security issues keep coming up. Among all these threats, the poisoning attack [7,9,11,14,31,32,38] is highly threatening. According to different approaches used for poisoning, there exist two main categories, data poisoning attacks [8,12,26,31] and model poisoning attacks [1,3,9]. Despite different adversarial goals, the most significant feature shared by all poisoning attacks is that the model usability will be sabotaged, regarding some specific samples or all data samples.

On the other side, there are also many insightful studies of defensive solutions to poisoning attacks. To eliminate the adversarial effect, many efforts have been made to implement Byzantine-robust federated learning [22,24,25,27,36,37]. FLTrust [6] adopts the idea that the parameter server assign trust scores computed by cosine similarity to each participants, the global model is based on bootstrapping trust local models. However, in [6] it assumes that the parameter server must keep a small clean training dataset for robust aggregation. Apart from above-mentioned methods, another approach tends to apply knowledge distillation. Han et al. [13] introduced another robust federated learning method

called CoMT(Collaborative Machine Teaching), where the learner (parameter server) is taught by distributed teachers (participants) with collaboratively fine-tuning. Lin et al. [20] investigated the ensemble distillation algorithm, a more flexible model fusion aggregation rule under heterogeneous federated learning scenario. Although lots of new solutions have been proposed, the game of attack and defense continues. Recent studies have reported two model poisoning attacks [3,9], which can breach Byzantine-robust solutions easily. Now it is urgent to think about alternative solutions to the defense of federated learning. That is also a major motivation of our work.

7 Conclusion

Model poisoning attacks are critical threats to FL. The best solution to defeating poisoning attacks is still under discussion. We give a practical solution based on similarity measurement in this paper. Through the analysis in a game-theory manner, we show the correctness of Romoa. Based on comparative experiments, we find that Romoa can defend against poisoning attacks effectively and outperforms Byzantine-robust solutions Krum and RFA in most cases. But we also note that collusive poisoning attacks become unbeatable when the proportion of adversarial participants is relatively large, like about 50%. How to protect FL in this case is quite challenging, and we will take this into account in our future work.

Acknowledgement. The authors would like to thank the reviewers for their helpful comments. This work was supported in part by National Key R&D Program of China under Grant 2020YFB1005900, NSFC-61902176, BK20190294, NSFC-61872176, and Leading-edge Technology Program of Jiangsu Natural Science Foundation (No. BK20202001).

Appendix

1 DNN Architectures

The DNN architectures for baseline FL tasks with MNIST and CIFAR-10 datasets are shown in Fig. 9 and Fig. 10, respectively.

Layer	Output Shape	Param #
Conv2D(64,5)+ReLU	(None, 24, 24, 64)	1664
Conv2D(64,5)+ReLU	(None, 20, 20, 64)	102464
Dropout(0.25)	(None, 20, 20, 64)	0
Flatten	(None, 25600)	0
Dense(128)+ReLU	(None, 128)	3276928
Dropout(0.5)	(None, 128)	0
Dense(10)+Softmax	(None, 10)	1290

Fig. 9. DNN architecture for MNIST tasks.

Layer	Output Shape	Param #
Conv2D(64,3)+ReLU	(None, 32, 32, 64)	1792
Conv2D(64,3)+ReLU	(None, 32, 32, 64)	36928
MaxPooling2D(2,2)	(None, 16, 16, 64)	0
Conv2D(128,3)+ReLU	(None, 16, 16, 128)	73856
Conv2D(128,3)+ReLU	(None, 16, 16, 128)	147584
Conv2D(128,3)+ReLU	(None, 16, 16, 128)	147584
MaxPooling2D(2,2)	(None, 8, 8, 128)	0
Conv2D(256,3)+ReLU	(None, 8, 8, 256)	295168
Conv2D(256,3)+ReLU	(None, 8, 8, 256)	590080
Conv2D(256,3)+ReLU	(None, 8, 8, 256)	590080
MaxPooling2D(2,2)	(None, 4, 4, 256)	0
Flatten	(None, 4096)	0
Dense(1024)+ReLU	(None, 1024)	4195328
Dropout(0.5)	(None, 1024)	0
Dense(10)+Softmax	(None, 10)	10250

Fig. 10. DNN architecture for CIFAR-10 tasks.

References

1. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics, pp. 2938–2948 (2020)
2. S S Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly) logarithmic overhead. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 1253–1269 (2020)

3. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. In: International Conference on Machine Learning, pp. 634–643 (2019)
4. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: byzantine tolerant gradient descent. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 118–128 (2017)
5. Bonawitz, K., et al.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1175–1191 (2017)
6. Cao, X., Fang, M., Liu, J., Gong, N.Z.: Fltrust: byzantine-robust federated learning via trust bootstrapping. In: 28th Annual Network and Distributed System Security Symposium, NDSS (2021)
7. Cao, X., Jia, J., Gong, N.Z.: Data poisoning attacks to local differential privacy protocols. In: 30th {USENIX} Security Symposium ({USENIX} Security 21) (2021)
8. Chen, X., Liu, C., Li, B., Lu, K., Song, D.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)
9. Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to byzantine-robust federated learning. In: 29th {USENIX} Security Symposium ({USENIX} Security 20), pp. 1605–1622 (2020)
10. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans. Inf. theory **21**(1), 32–40 (1975)
11. Fung, C., Yoon, C.J., Beschastnikh, I.: The limitations of federated learning in sybil settings. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2020), pp. 301–316 (2020)
12. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017)
13. Han, Y., Zhang, X.: Robust federated learning via collaborative machine teaching. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 4075–4082 (2020)
14. Huang, H., Mu, J., Gong, N.Z., Li, Q., Liu, B., Xu, M.: Data poisoning attacks to deep learning based recommender systems. In: 28th Annual Network and Distributed System Security Symposium, NDSS (2021)
15. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B.: Manipulating machine learning: poisoning attacks and countermeasures for regression learning. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 19–35 (2018)
16. John, P.G., Vijaykeerthy, D., Saha, D.: Verifying individual fairness in machine learning models. In: Conference on Uncertainty in Artificial Intelligence, pp. 749–758 (2020)
17. Kadhe, S., Rajaraman, N., Koyluoglu, O.O., Ramchandran, K.: Fastsecagg: scalable secure aggregation for privacy-preserving federated learning. arXiv preprint arXiv:2009.11248 (2020)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
19. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
20. Lin, T., Kong, L., Stich, S.U., Jaggi, M.: Ensemble distillation for robust model fusion in federated learning. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (2020)

21. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282 (2017)
22. Muñoz-González, L. Co, K.T., Lupu, E.C.: Byzantine-robust federated machine learning through adaptive model averaging. arXiv preprint arXiv:1909.05125 (2019)
23. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE symposium on security and privacy (SP), pp. 739–753 (2019)
24. Portnoy, A., Hendler, D.: Towards realistic byzantine-robust federated learning. arXiv preprint arXiv:2004.04986 (2020)
25. Reisizadeh, A., Farnia, F., Pedarsani, R., Jadbabaie, A.: Robust federated learning: the case of affine distribution shifts. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (2020)
26. Shafahi, A., et al.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: Advances in Neural Information Processing Systems, pp. 6103–6113 (2018)
27. Shen, S., Tople, S., Saxena, P.: Auror: defending against poisoning attacks in collaborative deep learning systems. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 508–519 (2016)
28. Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1310–1321 (2015)
29. Suciu, O., Marginean, R., Kaya, Y., Daume III, H., Dumitras, T.: When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In: 27th {USENIX} Security Symposium ({USENIX} Security 18), pp. 1299–1316 (2018)
30. Toch, E., et al.: The privacy implications of cyber security systems: a technological survey. ACM Comput. Surv. (CSUR) **51**(2), 1–27 (2018)
31. Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L.: Data poisoning attacks against federated learning systems. In: European Symposium on Research in Computer Security, pp. 480–501 (2020)
32. Wang, H., et al.: Attack of the tails: yes, you really can backdoor federated learning. In: Advances in Neural Information Processing Systems (2020)
33. Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S.A.: Safa: a semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. (2020)
34. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. arXiv preprint arXiv:1903.03934 (2019)
35. Yeom, S., Fredrikson, M.: Individual fairness revisited: transferring techniques from adversarial robustness. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI, pp. 437–443 (2020)
36. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: towards optimal statistical rates. In: International Conference on Machine Learning, pp. 5650–5659 (2018)
37. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Defending against saddle point attack in byzantine-robust distributed learning. In: International Conference on Machine Learning, pp. 7074–7084 (2019)

38. Zhang, J., Chen, J., Wu, D., Chen, B., Yu, S.: Poisoning attack in federated learning using generative adversarial nets. In: 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 374–380 (2019)
39. Zhang, M.R., Lucas, J., Hinton, G., Ba, J.: Lookahead optimizer: k steps forward, 1 step back. In: Advances in Neural Information Processing Systems, pp. 9597–9608 (2019)
40. Zheng, S., et al.: Asynchronous stochastic gradient descent with delay compensation. In: International Conference on Machine Learning, pp. 4120–4129 (2017)