Model Poisoning Defense on Federated Learning: A Validation Based Approach

Yuao Wang[1][0000−0002−1976−7661], Tianqing Zhu[1][0000−0003−3411−7947], Wenhan Chang[1][0000−0003−3350−5171], Sheng Shen[2][0000−0003−4734−1008], and Wei Ren[1][0000−0001−8590−1737]

[1] China University of Geosciences,Wuhan
tianqing.e.zhu@gmail.com
[2] School of Computer Science, University of Technology Sydney
Sheng.Shen-1@student.uts.edu.au

Abstract. Federated learning is an improved distributed machine learning approach for privacy preservation. All clients collaboratively train the model using on-device data, and the centralized server only aggregates clients′ training results instead of collecting their data. However, there is a serious shortcoming for federated learning that the centralized server cannot detect the validity of clients′ training data and correctness of training results due to its limitation on monitoring clients′ training processes. Federated learning is vulnerable to some attacks when attackers maliciously manipulate training data or updates, such as model poisoning attacks. Attackers who execute model poisoning attacks can negatively affect the global models' performance on a targeted class by manipulating the label of this class at one or more clients. Currently, there is a gap in the defense methods against model poisoning attacks in federated learning. To address the above shortcoming, we propose an effective defense method against model poisoning attack in federated learning in this paper. We validate each client′s local model with a validation set. The server will only receive updates from well-performing clients to protect against model poisoning attacks. We consider the following two cases: all clients have a very similar distribution of training data and all clients have a very different distribution of training data, and design our methods and experiments for both cases. The experimental results show that our defense method can significantly reduce the success rate of model poisoning attacks in both cases in a federated learning setting.

Keywords: Federated learning · Model poisoning attack · Defense.

1 Introduction

Federated learning is an improved distributed machine learning approach that involves many clients collaboratively training a model under the orchestration of a server [12]. During every round of training, clients who are chosen and allocated the model by the server use on-device data to train the model and send model updates back to the server. And then, the central server averages all updates

to update the global model as the result of training this round. Compared to traditional distributed machine learning, federated learning provides a substantial privacy preservation that clients′ data keep on the device instead of being collected by others [5] [16]. Each client is an isolated data island, and data never leaves the island. Once used to train the model, the only ship to leave is a model update to the central server [2] [7]. Federated learning can also overcome the limitation of a data island on the size and characteristics. Therefore, federated learning ideally benefits both clients to protect their data being used securely, and the server to involve more clients [5] [13].

Although federated learning made huge improvements in the privacy and security of distributed machine learning, federated learning is still attracting adversaries and malicious attacks. Recently, many attack methods are being proposed against federated learning. Poisoning attack is one of the powerful attacks in which adversaries inject crafted attack points into the training data or tamper with the model training process. [3] [6].Poisoning attacks cause significant threats to federated learning.

Most of the proposed poisoning attacks fall into two broad categories, data poisoning attack ,and model poisoning attack. Data poisoning attack focuses on the training data collection phase, injecting malicious data into the training dataset before the training process starts, while model poisoning attack targets on model training phase, directly manipulating the local model parameters [3] . Some researchers have studied how to defend against data poisoning attacks in federated learning setting. Cao et al. [2]defended data poisoning attacks by solving a maximum clique problem to ignore suspected local models during the global model aggregation.

However, few works focus on how to defense model poisoning attacks in federated learning due to some difficulties. Firstly, federated learning involves a huge amount of clients collaboratively training the model, and it is hard to guarantee none of them is malicious. Secondly, the central server cannot observe clients since only updates will be uploaded to the server [1]. Clients may adversarially use poisoned data to train the model which can result in a negative impact on the model performance without the servers being aware. Therefore,the key point to solve the problem is how to find out the malicious participant without breaking the privacy of the participant, so that the central server will not receive the parameter updates submitted by the malicious participant .Thus defense against the model poisoning attacks.

In order to solve the challenge and problems mentioned above, we propose a new defense method against model poisoning attacks in federated learning. Our key contributions can be summarized as follows: 1. we propose a new defense method against model poisoning attacks which validates each client's local model with a validation set, and then gives each client a label based on the validation results, finally the central server receives only updates from the performing clients. 2. Our proposed method can hugely reduce the success rate of model poisoning attacks by validating each client in each round of federated learning training with the validation set. 3. Our method protects against model poisoning attacks

by screening out malicious clients so that the central server does not aggregate updates from existing clients. We will introduce some background knowledge in section 2, and our proposed method in section 3. We demonstrate our experiment settings, results and analysis in section 4.

2 Background

2.1 Federated learning

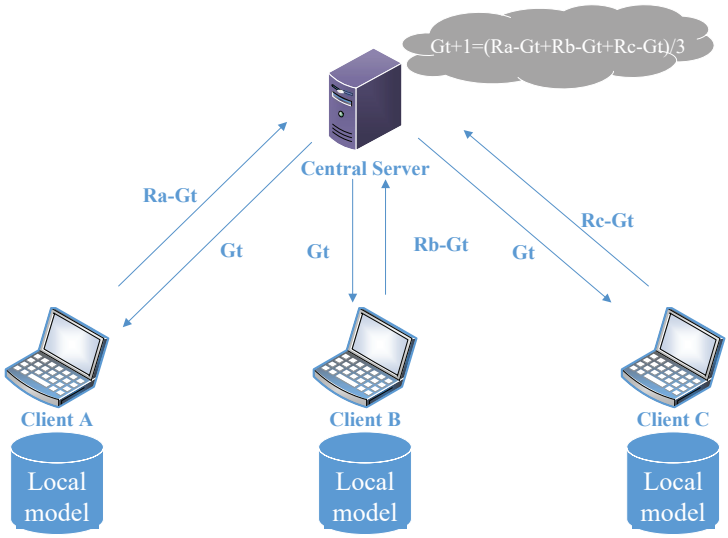


Fig. 1: Federal learning training process

Federated learning is an improved approach of distributed machine learning, which significantly reduces risks of privacy leakage [5] [7]. Clients collaboratively train a model without their data being collected by others. Fig.1 shows the training process of federated learning: at each round t, the server announces a task and wait for all clients′ responses for availability. The server then selects a sub-group of clients who are ready to train the model and sends the current global model parameter G_t to these clients. Then the chosen clients start training the received model using on-device data and update the local global R_i. When most clients finish training, they send model updates $R_i - G_t$ back to the central server, and the central server updates the global model for this round by averaging all updates from clients [2].The process is repeated until the model converges.

That is, each client K runs a certain number of SGD steps locally and computes the average gradient on its local data at the current model G_t. Then the central server averages the updated local model to generate the updated global

model $G_{t+1} \leftarrow \sum_{k=1}^{K} \frac{1}{K} G_{t+1}^{i}$. Each client sends an update to the central server as $G^k \leftarrow \sum G^k - \eta \bigtriangledown F_k(G^k)$. Among them, η is the learning rate [10].

Federated learning involves many policies to protect users′ privacy and security [8]. The secure aggregation protocol, for example, protects data from other threats by ensuring that updates from clients remaining encrypted, thus a central server can only observe an encrypted update from clients instead of true value. Secure aggregation protocol includes a secret sharing scheme,differential privacy [18] [19], secure multiparty computation and homomorphic encryption [7]. Through these strategies, federated learning is able to defense against many attacks, prevent information from being leaked, and reduce communication costs. However, federated learning protocol designs may contain vulnerabilities for both potentially malicious servers and any malicious participant. Such attacks pose significant threats to federated learning [9].

2.2 Model poisoning attack

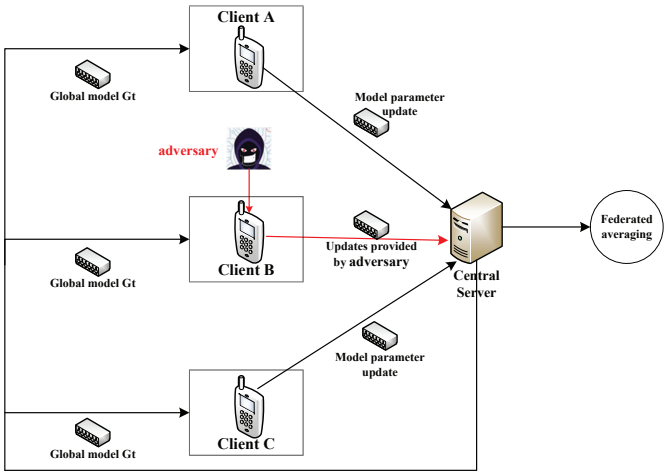


Fig. 2: Model Poisoning Attack. The attacker attacked client B and took control of all its local data. All updates sent by the attacked client B to the central server were specified by the attacker.

Poisoning attack mainly refers to the process of training data collection or model training in which the attacker attacks the model to achieve a malicious purpose.

Poisoning attacks have two categories: data poisoning attacks and model poisoning attacks [4]. For data poisoning attacks, the attacker contaminates the samples of the training dataset. On the other hand, the attacker can also execute model poisoning attacks by uploading malicious updates to the centralized server

instead of modifying the training data directly [9] , which results in a hugely negative impact on the model performance [14].

Model poisoning attack in federated learning The target of model poisoning attack is to make the global model misclassify an attacker′s expected class of things. Different from data poisoning attacks that aim at the integrity of training dataset collection, the adversary executing model poisoning attack in federated learning setting manipulates the label of a targeted class of data into a wrong label. The wrong label results in a wrong prediction result for the target class from the local model updates which will be sent to the central server for aggregation, and then negatively affect the global model′s performance on this class of things [9]. Normally, model poisoning attack can lead the global model to reach a very high accuracy of prediction results on the manipulated class but keep performing well on other classes.

The attacker may have strong capabilities such as directly controlling the local training data of the client being attacked and modifying the value of the updates. This kind of attack can achieve high performance on both the main task of the model and the backdoor task as attacker′s expectation. As we mentioned above, the centralized server cannot observe the clients′ training data updates and be aware of malicious updates from the attacker, which causes a serious challenge to defend against such attacks. Currently, there is not too much effective work addressing the vulnerability of federated learning against model poisoning attacks.

3 Defenses method for model poisoning

In order to solve the above problems, we propose a defense method. Our defense method successfully defended against the model poisoning attack.

3.1 Model poisoning defense overview

Our basic idea is to identify the malicious participants so that the central server does not aggregate updates from the malicious participants, thus preventing model poisoning attacks. Therefore, we propose a defense strategy based on the verification set. At each round of training, after each client has trained the local model with the global model and the local training dataset, we use the validation set to verify the classification accuracy of the local model.If the local model has a much lower classification accuracy than other local models, we mark the client as a malicious client. The central server will not aggregate updates from that client. We take two scenarios into consideration, respectively each client is similar in composition and each client has only one type of data.

3.2 scenario 1: the client is similar in composition

In this case, each client's local training dataset is composed similarly.In model poisoning attacks, the attacker's goal is to cause the federated learning model to

misclassify a set of selected inputs.As mentioned in our defense policy, we need to screen out malicious participants. Therefore, we need to find out whether each client is "partial", in order to protect against backdoor attacks. "Partial" refers to misclassification of a certain type of numbers but accurate classification of other types of numbers.

Algorithm 1 Each client is similar in composition. M represents the number of clients participating in training in each round, T represents the total number of communication rounds, D represents the data set, A represents the classification accuracy of a model, and I represents the threshold.

initialize x_0
for each round $t = 1, 2, ..., T$ **do**
  $S_t \leftarrow (random\ set\ of\ M\ clients)$
end for
for each client $i \in S_t$ **do**
  $S_{t+1} \leftarrow ClientUpdate(i, x_t)$
  $A_{class}(x^i_{t+1}, D_v)$
end for
for each dataset j **do**
  if $A_{class}(x^i_{t+1}, D_j) < I$ **then**
    $x_i \leftarrow$ "$Partial$"
  end if
  if $A_{class}(x^i_{t+1}, D_j) > I$ **then**
    $x_i \leftarrow$ "$non - partial$"
  end if
end for
if $x_i =$ "$non - partial$" **then**
  $x_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} x^i_{t+1}$
end if

In each round of training, the central server gives the current global model parameter G_t to each client.Each client performs local training on the local dataset based on the global model parameters. After training, we obtain the local model x^i_{t+1} .In the normal federated learning process, each client sends the updates $x^i_{t+1} - G_t$ back to the central server once the local model training is completed. But in our defense method, we validate the local model x^i_{t+1} with a validation set before committing the updates to central server. We need to verify the classification accuracy of the main tasks of the model and the accuracy of each type of number classification (verification of the accuracy of each type of number classification is to judge whether it is "partial"). The main task mentioned here is the main training task of the model, while the subtask is the target chosen by the attacker to control the behavior of the model.

The defense strategy has three steps. The first step is to verify the classification accuracy of the main tasks of the model. We have to note the classification success rate of the local model for each client on the validation set for reference.

The second step is to verify the classification accuracy of each type of number. First of all, we divide the validation dataset in this round of training into 10 new validation datasets according to the label. Each new validation dataset contains only one class of data in order to verify the local model′s classification accuracy for each class of Numbers. Then we validate the local model with each of the previously mentioned validation sets and record the results. If the classification accuracy rate of each type of the number of a client is higher than 65%(Experimental results show that the classification success rate of the client is more than 80% without attack while in the case of poisoning attack, the classification success rate of the client is about 50%. Based on the experimental results, we selected the median value of 65% as the threshold.), the client will be labeled "non-partial". The server will aggregate the updates from the client in this round of training. On the other hand, if the classification accuracy rate of a certain type of number is less than 65%, the client is labeled "partial", and the server is in this round of training does not aggregate updates on the client-side. The third step is to verify the label. If a client has a label "non-partial", it is judged as a normal client. However, if a client has a label "partial", it is judged as a malicious client.

3.3 scenario 2: client has only one type of data

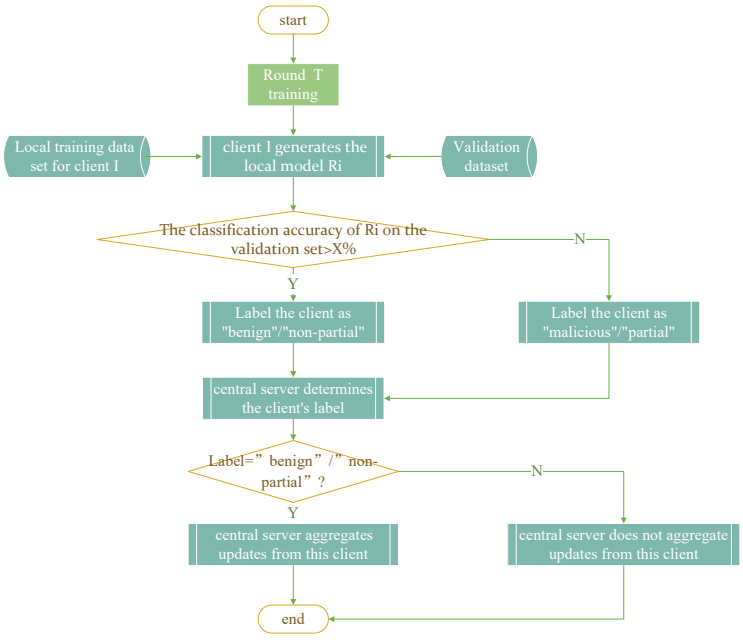


Fig. 3: Defense algorithm flow chart.

In this case, each client's local dataset has only one class of numbers. The attacker attacks one or more clients, modifies the parameter update of the attacked client, and causes the model to predict a certain class of numbers as the target label set by the attacker. Therefore, we need to find out which client is malicious and then screen out the malicious clients. This situation is different from the previous one, because each client's local training data set has only one class of numbers, so we don't need to judge "partial " issues.

Similarly, after each client has trained the local model, we validate the local model x_{t+1}^i with a validation set.Unlike the other scenario, we do not need to classify validation datasets by label, but use validation datasets directly. We need to verify the classification accuracy of the model. Similarly, after each client has trained the local model, we validate the local model with a validation set. There are two steps. The first step is to verify the classification accuracy of the model. If a local model has a higher classification accuracy than 12% on the validation set, the client is marked as "benign". On the other hand, if its classification accuracy on the validation set is less than 12 %,(Experimental results show that the classification success rate of the client is more than 15% without attack while in the case of poisoning attack, the classification success rate of the client is about 9%. Based on the experimental results, we selected the median value of 12% as the threshold.) the client is marked as "malicious". The second step is to validate the client's label. If a client is labeled "benign," the central server receives updates and aggregates them from that client. Conversely, if a client is labeled "malicious," the central server identifies it as an attacker and does not aggregate updates from it.

3.4 Discussion on the proposed method

Our approach can screen out malicious clients so that the central server does not aggregate updates from malicious clients. This is the way to defend against model poisoning attacks. We hope that our method can fully protect the model against poisoning attacks, so that the model's global task classification success rate and subtask classification success rate are consistent with the normal federated learning rate without attacks.

The key point of our approach is to determine whether a client is malicious or benign by verifying the global task classification success rate of each client's local model and the classification success rate of each type of data in each round of training,and then screen out malicious clients to protect against attacks.

4 Experiments

4.1 Experimental setup

We used a three-layer neural network with an input layer of 28×28, a hidden layer of 12 neurons, and an output layer of 10 neurons.We evaluate our attack strategies and defense strategies using MNIST dataset. The MNIST dataset

Algorithm 2 Each client has only one type of data. M represents the number of clients participating in training in each round, T represents the total number of communication rounds, D represents the data set, A represents the classification accuracy of a model,and H represents the threshold.

initialize x_0
for each round $t = 1, 2, ..., T$ **do**
  $S_t \leftarrow (random\ set\ of\ M\ clients)$
end for
for each client $i \in S_t$ **do**
  $S_{t+1} \leftarrow ClientUpdate(i, x_t)$
  $A_{class}(x^i_{t+1}, D_v)$
end for
if $A_{class}(x^i_{t+1}, D_v) < H$ **then**
  $x_i \leftarrow$ "$malicious$"
end if
if $A_{class}(x^i_{t+1}, D_v) < H$ **then**
  $x_i \leftarrow$ "$benign$"
end if
if $x_i =$ "$benign$" **then**
  $x_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} x^i_{t+1}$
end if

is mainly composed of some pictures of handwritten digits and corresponding labels. There are 10 types of pictures, which correspond to 10 Arabic numerals from 0 to 9 respectively.

A. Each client is similar in composition

We mainly consider the case where the purpose of the malicious agent is to misclassify a single example in the desired target class (r = 1) while classifying other inputs correctly. For the MNIST dataset, the example belongs to class 4 (sandal), and its purpose is to misclassify it as the class 0.

We select 5,000 images from the 60,000 training images of the MNIST dataset as the training set for our experiment, and select 20,000 as the validation set. In each validation round, we randomly select one-tenth of the data from the validation set as the validation set for the round. That is, 2000 digital images are used as the verification set each time, so that each verification set is different. We first validate the classification accuracy of each client's local model with the selected validation set. Then, we divide the validation set into ten new validation sets according to the label of the digital image. The data label in the first new validation set is all "0", the data label in the second new validation set is all "1", and so on. The filtered validation set is used to verify the classification accuracy of the local model for each type of data.

We set the number of agents K to 10, that is, select all agents in each iteration. We run federated learning until the pre-specified test accuracy is reached.

B. Each client has only one type of data

Same as case A, we mainly consider the case where the purpose of the malicious agent is to misclassify a single example in the desired target class (r = 1) while classifying other inputs correctly. For the MNIST dataset, the example belongs to class 4 (sandal), and its purpose is to misclassify it as class 0.

We selected 30,000 training images from 60000 training images of MNIST dataset, and our experimental training data set was screened from these 30,000 digital images. We divided 30,000 digital images by label into 10 categories to form 10 new datasets. There is only one class of data in each dataset. For the data set labeled as '4', we divided it into three new datasets, which will be used as training data to participate in the experiment. For the other nine datasets, 1000 data are randomly selected from each dataset to form nine new datasets respectively. These nine datasets will be used as training data to participate in the experiment.

We set the number of agents K to 12, that is, select all agents in each iteration. The twelve datasets that we have previously partitioned will be divided into local training datasets as twelve clients. We run federated learning until the pre-specified test accuracy is reached.

4.2 Model poisoning attack results

In this section, we use the adversarial goals laid out in the previous section3.1 to formulate the adversarial optimization problem. We then show how to achieve targeted model poisoning.

A. Each client is similar in composition

We first implemented federated learning for ten clients without an attacker, and the results are shown in Fig.4.

In the experiment, the attacker attacked client A and controlled all the local data of Client A. After poisoning client A, the attacker tampered with the updates that Client A was going to send to the central server with malicious updates, expanded the weight of malicious updates, and made the global model deviate from the direction the attacker wanted.

The results (the results are shown in Fig.5) show that after the attack, the classification accuracy of the benign client and the malicious client to the global main task is significantly reduced, and finally when the model converges, the classification accuracy is reduced to 50%. Compared with the non-attack condition, the model poisoning attack significantly reduces the classification accuracy of the model. For backdoor missions, that is, classify the number '4' as '0', the results show that backdoor missions have a high attack success rate in both malicious and benign models. Finally, when the model converges, the attack success rate of the backdoor mission reaches 80%.These results demonstrate the magnitude of the impact of model poisoning attacks on federated learning.

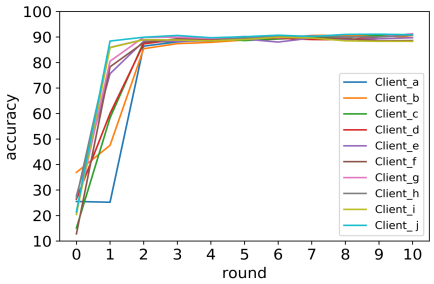


Fig. 4: Classification accuracy of five clients in normal federated learning training.

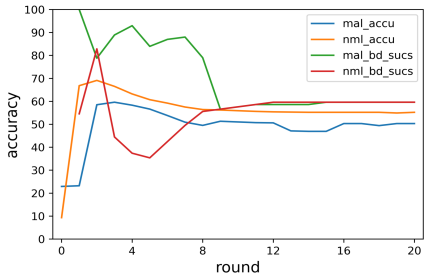
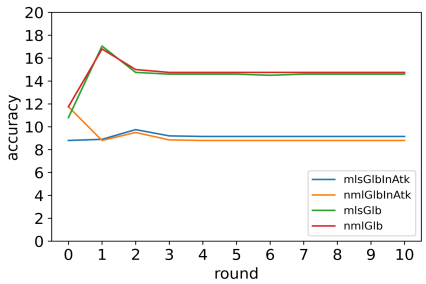


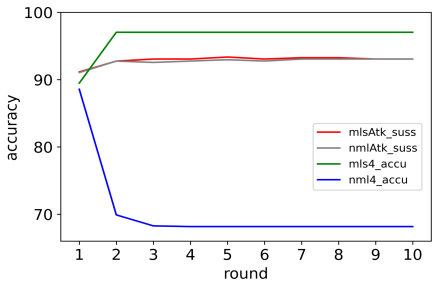
Fig. 5: Backdoor accuracy and Global task accuracy.

B. Each client has only one type of data

We first implemented the normal federated learning training without an attacker and we recorded the results, with which we will then compare the results of the model poisoning attack. We then implemented a model poisoning attack. The attacker poisons client D (the local data set label for client D is all '4') and takes control of all its local data. The attacker tampered with the update parameters of the client, sending malicious updates to the central server and increasing the weight of the malicious updates, which made the global model deviate from the direction the attacker wanted.



(a) Comparison of global task classification accuracy.



(b)Comparison of subtask classification success rate and attack success rate.

Fig. 6: Comparison of the classification accuracy of global and subtasks between benign and poisoned clients under normal federated learning training and attack.

We compared the classification success rate of benign and poisoned clients for global and subtasks in the case of normal federated learning training and

model poisoning attacks. We selected the client D which was attacked in the attack situation and recorded the classification accuracy of the client in the normal training situation and the attacked situation. In addition to client D, we randomly selected a client from other benign clients, and recorded its classification accuracy in the two cases, and statistically compared the results.

Fig.6(a) shows the comparison of the classification accuracy of global tasks between benign and poisoned clients under normal federated learning and training conditions and under attack conditions. Under normal federated learning conditions, the global task classification accuracy of the two clients is around 15%.However, in the case of the model poisoning attack, the classification accuracy of the two clients on the global task dropped to about 9%.This proves that the model poisoning attack has a great impact on the global task of the model, which will lead to the reduction of the classification accuracy of the global task of the model.

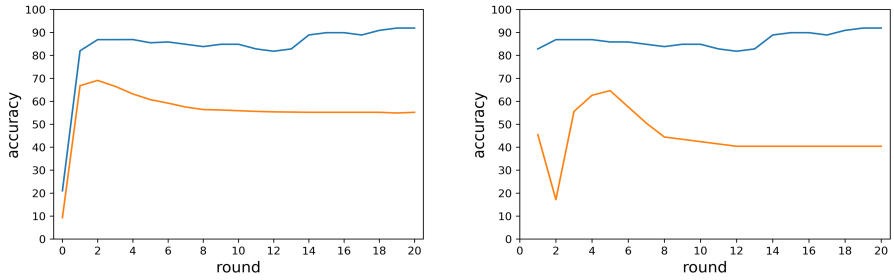
Fig.6(b) shows the classification accuracy of benign clients and poisoned clients on subtasks under normal federated learning and training conditions and the attack success rate of benign clients and poisoned clients on subtasks under attack conditions. Under normal federated learning conditions, the benign client's classification accuracy for subtasks is around 70%.Client D, which is the client that was poisoned in the attack case, had a classification accuracy of about 96 percent for subtasks. However, in the case of model poisoning attack, the attack success rate of the two clients on subtasks reached about 92%.This proves that the attack success rate of model poisoning attacks against model subtasks is very high, and also proves that model poisoning attacks are powerful.

4.3 The performance of our defenses method

A. Each client is similar in composition

In the first round of training, our defense strategy uses a validation set to validate the local model of client A after client A has trained its own local model. When the validation set with all labels of '4 'is used for verification, we find that the classification accuracy of the local model for digital image '4' is extremely low. Therefore, client A is labeled "partial". The central server sees the "partial" label and chooses not to aggregate updates from client A.

Fig.7 (a) shows the classification accuracy of the same benign client for the global task when our defense method is used and when it is not. In the case of no defense, the model poisoning attack reduces the classification accuracy of global tasks to 50%.After using our method for defense, the success rate of global task classification increased to 90%. Fig.7 (b) shows the classification accuracy of subtasks for the same benign client when our defense method is used and when it is not. In the case of no defense, the model poisoning attack reduces the classification accuracy of subtasks to 20%.After using our method for defense, the model's classification success rate for global tasks increased to 90% .The experimental results show that the defense method is effective and successful.



(a) The accuracy of main missions. (b)The accuracy of backdoor missions.

Fig. 7: The model compares the accuracy of main missions and backdoor missions with and without defense.

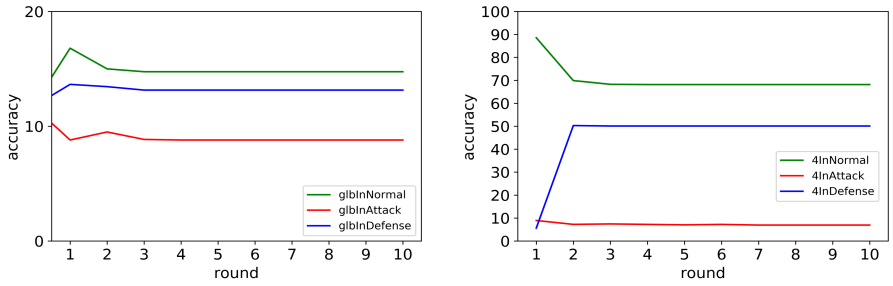
B. Each client has only one type of data

In the first round of training, our defense strategy uses a validation set to validate the local model of client D after client D has trained its own local model. We find that the classification accuracy of the local model of client D is extremely low. Therefore, client D is labeled "malicious ". The central server sees the "malicious" label and chooses not to aggregate updates from client D.

We compared the model's classification success rates for global and subtasks under three conditions: normal federated learning training, model poison attack, and defense. Use the experimental results to verify the effectiveness of our defense methods.

Fig.8(a) shows a comparison of the model's classification success rate for global tasks under three conditions: normal federated learning training, model poison attack, and defense. Under the condition of normal federated learning training, the classification accuracy of the model for global tasks is about 15%.In the case of model poisoning attack, the model's classification accuracy of global tasks is about 10%.In the case of using our defense strategy, the classification accuracy of the model for global tasks is about 13%.The results show that compared with the case of no defense, our defense method improves the accuracy of the model on the global task, although it is not as high as the normal federation learning and training, but it is close. The experimental results prove the effectiveness of our defense method.

Fig.8(b) shows the comparison of the model's classification success rate for subtasks under three conditions: normal federated learning training, model poison attack, and defense. Under normal federated learning and training conditions, the classification success rate of subtasks reached 70%.In the case of model poisoning attack, the model's classification success rate of subtasks was reduced to 10%.With our defense approach, the model's classification success rate for subtasks went back up to 50%.Compared with the non-defense case, our defense



(a) Comparison of global task classification accuracy.

(b)Comparison of subtask classification accuracy.

Fig. 8: Comparison of the classification accuracy of the global task and the classification success rate of the subtask under federated normal learning training, model poisoning attack and defense.

method improves the classification success rate of subtasks by 40%, which proves the effectiveness of our method.

5 Related work

In order to make federated learning more robust and practical, some researchers have come up with some defense strategies against poisoning attacks. For instance, Zhao et al. use client-side cross-validation to protect against poison attacks [17]. They randomly selected a small group of clients that participate in federated learning and asked them to evaluate each model update with their own local data, which was used to detect abnormal updates. Specifically, the server randomly divides K updates into D parts. Each part is then aggregated into sub-models, and each sub-model is randomly assigned to the selected client, and each client evaluates the sub-model. Next, the server adopts a majority voting strategy, and the client participating in the evaluation needs to submit a binary matrix to declare whether the corresponding category of data samples has been properly classified. Although both this paper and ours detect abnormal updates to prevent poisoning attacks, the strategy of this paper is quite different from ours. Firstly, our defense strategy is to validate each model update with a validation set. And the data in our validation set is brand new data rather than local private data from the client participating in federated learning. Our settings reduce the risk of user privacy disclosure. Secondly, in order to defend against backdoor attacks, we have verified the classification accuracy of each update on the validation set for the main tasks of the model, as well as the classification accuracy of each update for each type of data. In this way, the attacker can be detected even if he performs well on the model's main task. After validating the

model updates, we label each client and the server determines whether or not to aggregate the updates from a particular client. Our defenses are effective against poisoning attacks and are low in computing and communication costs and easy to deploy.

Tolpegin et al. propose an automated defense strategy that identifies the relevant parameter subset and study participant updates using dimensionality reduction (PCA) against label ipping attacks [15]. Because updates from malicious participants belong to a distinct cluster from those from benign participants. Based on this, they use gradient clustering to identify malicious actors before the servers aggregating updates from various clients. After the aggregator identifies the malicious actors, it will ignore their updates and eventually converge to a high-utility model. This defense strategy is completely different from our defense strategy. First of all, their defense strategy is against label ipping attacks in data poisoning attacks, while ours is against backdoor attacks in model poisoning attacks. Secondly, their strategy is to identify malicious actors through gradient clustering, while our defense strategy is to identify malicious actors by validating each client's model update through validation sets.

Also, to protect against backdoor attacks, Ozdayi et al. proposed a lightweight defense [11], which does not change the federated learning structure. Their defense strategy adjusts the learning rate during each round of training and move the model towards a particular direction to maximizes the loss of the attacker. The core idea behind their defense was to adjust the learning rate according to the sign information of updates from the agents. Compared to their defense strategy, our defense strategy changed the structure of federated learning, and we validated each model update with a validation set before the server aggregated. We were equally effective in defending against a backdoor attack.

In addition, there are many studies on the defense of deep learning. Such as Cretu et al. extend the training phase of anomaly sensors with a new sanitization phase that uses the novel micromodels in a voting scheme to eliminate attacks and anomalies from training data [4]. While not all of these methods of defense are intended for federal learning, they will help us continue to study the defense of federal learning in the future.

6 Conclusions and future work

In summary, federated learning is vulnerable to model poisoning attack. The challenge is caused by the fact that the centralized server cannot observe clients′ training data and is difficult to perform the validation of clients′ updates. To address the above challenges, we have presented a new defense method against model poisoning attack in federated learning in this paper. Our method validates every client′s training results using a validation dataset, to evaluate if these client′s updates are valid for aggregation. We have considered two cases in federated learning settings: all clients′ training data have the similar distribution and all clients′ training data have the different distribution. The experiment re-

sults demonstrate that our method can hugely reduce model poisoning attacks′ success rate in both cases, and prevent misclassification on the poisoned class.

There is still much work we have to do in the future.We will continue to study how to defend against attacks in federated learning, such as data poisoning attacks, inference attacks, and so on.We know that there are obvious vulnerabilities in federated learning protocol designs , and these vulnerabilities pose signicant threats to federated learning.In the future we may be able to improve the federated learning design to protect against attacks,making federated learning safer and more practical.

References

1. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. CoRR **abs/1807.00459** (2018), http://arxiv.org/abs/1807.00459
2. Cao, D., Chang, S., Lin, Z., Liu, G., Sun, D.: Understanding distributed poisoning attack in federated learning. In: 25th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2019, Tianjin, China, December 4-6, 2019. pp. 233–239. IEEE (2019). https://doi.org/10.1109/ICPADS47876.2019.00042, https://doi.org/10.1109/ICPADS47876.2019.00042
3. Fang, M., Cao, X., Jia, J., Gong, N.Z.: Local model poisoning attacks to byzantine-robust federated learning. CoRR **abs/1911.11815** (2019), http://arxiv.org/abs/1911.11815
4. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R.G.L., Rouayheb, S.E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P.B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S.U., Sun, Z., Suresh, A.T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F.X., Yu, H., Zhao, S.: Advances and open problems in federated learning. CoRR **abs/1912.04977** (2019), http://arxiv.org/abs/1912.04977
5. Konecný, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. CoRR **abs/1610.02527** (2016), http://arxiv.org/abs/1610.02527
6. Konecný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. CoRR **abs/1610.05492** (2016), http://arxiv.org/abs/1610.05492
7. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Process. Mag. **37**(3), 50–60 (2020). https://doi.org/10.1109/MSP.2020.2975749, https://doi.org/10.1109/MSP.2020.2975749
8. Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y., Yang, Q., Niyato, D., Miao, C.: Federated learning in mobile edge networks: A comprehensive survey. IEEE Commun. Surv. Tutorials **22**(3), 2031–2063 (2020). https://doi.org/10.1109/COMST.2020.2986024, https://doi.org/10.1109/COMST.2020.2986024
9. Lyu, L., Yu, H., Yang, Q.: Threats to federated learning: A survey. CoRR **abs/2003.02133** (2020), https://arxiv.org/abs/2003.02133

10. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., 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. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (2017), http://proceedings.mlr.press/v54/mcmahan17a.html
11. Özdayi, M.S., Kantarcioglu, M., Gel, Y.R.: Defending against backdoors in federated learning with robust learning rate. CoRR **abs/2007.03767** (2020), https://arxiv.org/abs/2007.03767
12. Pillutla, V.K., Kakade, S.M., Harchaoui, Z.: Robust aggregation for federated learning. CoRR **abs/1912.13445** (2019), http://arxiv.org/abs/1912.13445
13. Shen, S., Zhu, T., Wu, D., Wang, W., Zhou, W.: From distributed machine learning to federated learning: In the view of data privacy and security. CoRR **abs/2010.09258** (2020), https://arxiv.org/abs/2010.09258
14. Suya, F., Mahloujifar, S., Evans, D., Tian, Y.: Model-targeted poisoning attacks: Provable convergence and certified bounds. CoRR **abs/2006.16469** (2020), https://arxiv.org/abs/2006.16469
15. Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L.: Data poisoning attacks against federated learning systems. CoRR **abs/2007.08432** (2020), https://arxiv.org/abs/2007.08432
16. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 12:1–12:19 (2019). https://doi.org/10.1145/3298981, https://doi.org/10.1145/3298981
17. Zhao, L., Hu, S., Wang, Q., Jiang, J., Shen, C., Luo, X.: Shielding collaborative learning: Mitigating poisoning attacks through client-side detection. CoRR **abs/1910.13111** (2019), http://arxiv.org/abs/1910.13111
18. Zhu, T., Li, G., Zhou, W., Yu, P.S.: Differentially private data publishing and analysis: A survey. IEEE Trans. Knowl. Data Eng. **29**(8), 1619–1638 (2017). https://doi.org/10.1109/TKDE.2017.2697856, https://doi.org/10.1109/TKDE.2017.2697856
19. Zhu, T., Ye, D., Wang, W., Zhou, W., Yu, P.S.: More than privacy: Applying differential privacy in key areas of artificial intelligence. CoRR **abs/2008.01916** (2020), https://arxiv.org/abs/2008.01916