ATTACK-RESISTANT FEDERATED LEARNING WITH RESIDUAL-BASED REWEIGHTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning has a variety of applications in multiple domains by utilizing private training data stored on different devices. However, the aggregation process in federated learning is highly vulnerable to adversarial attacks so that the global model may behave abnormally under attacks. To tackle this challenge, we present a novel aggregation algorithm with residual-based reweighting to defend federated learning. Our aggregation algorithm combines repeated median regression with the reweighting scheme in iteratively reweighted least squares. Our experiments show that our aggression algorithm outperforms other alternative algorithms in the presence of label-flipping, backdoor, and Gaussian noise attacks. We also provide theoretical guarantees for our aggregation algorithm.

1 INTRODUCTION

Federated learning is a machine learning methodology for training a global model with decentralized data stored on multiple or even millions of devices (McMahan et al., 2017). In federated learning, private data is stored locally in isolated devices and will not be revealed to other parties during training. Federated learning can enable numerous real-world machine learning applications by utilizing massive training data that are privacy-sensitive and scattered on different devices (Bonawitz et al., 2017). For instance, multiple hospitals can collaborate to train a global model for classifying diseases using X-ray images without compromising patient privacy. Note that these hospitals may possess X-ray images in different quantities and varieties, resulting in the non-IID (independent and identically distributed) data distribution. Federated learning is different from distributed learning in the sense that the training data is often non-IID and we have no control over data distribution in federated learning.

The default federated learning aggregation algorithm *FedAvg* (McMahan et al., 2017) that takes the average of locally updated models is vulnerable to various attacks. We find that federated learning suffers from label-flipping, backdoor, and Gaussian noise attacks in our experiments. When a local model is poisoned, the aggregated global model can also be poisoned and fail to behave correctly. A label-flipping attack (Biggio et al., 2012) happens where an attacker assigns incorrect labels to some data. For example, an attacker can train a local model with cat images mislabelled as dogs and then share the poisoned local model for aggregation.

Mitigating attacks in federated learning or distributed learning has been explored in recent research (Chen et al., 2017; Yin et al., 2018; Fung et al., 2018; Blanchard et al., 2017). Although the median or trimmed mean aggregation algorithms (Yin et al., 2018) may seem plausible in distributed learning, their performance degrades in federated learning when data is non-IID. FoolsGold (Fung et al., 2018) is a defense algorithm that identifies participants with similar models as attackers but this strategy may not work when some harmless participants have similar local data. To make federated learning more attack-resistant, we develop an aggregation algorithm that is robust against label-flipping, backdoor, and Gaussian noise attacks in a general non-IID setting. We derive our aggregation algorithm by adopting the repeated median estimator (Siegel, 1982) and the reweighting scheme in iteratively reweighted least squares (IRLS) (Holland and Welsch, 1977; Rand R, 1997). We estimate the confidence of each parameter in the local models and then the weight of each local model can be computed by heuristically accumulating all the parameter confidence in each local model. Our algorithm is straightforward to implement (less than 100 lines). Furthermore, we provide theoretical guarantees for our aggregation algorithm.

We compare our proposed algorithm to several baselines by conducting experiments on four datasets, the MNIST dataset (LeCun et al., 1998), CIFAR-10 dataset (Krizhevsky et al., 2009), Amazon Reviews dataset (Ruining and Julian, 2016) and the Lending Club loan dataset (Kan, 2019). Our proposed aggregation significantly mitigates the impact of attacked models in non-IID federated learning and outperforms other baselines in our evaluation.

2 Related Work

Adversarial attacks on federated learning. Several attacks have been studied against federated learning (Wang et al., 2018; Biggio et al., 2012; Fung et al., 2018; Hayes et al., 2019; Hitaj et al., 2017; Melis et al., 2019). The label-flipping attack (Biggio et al., 2012) is shown to have great harm to a federated system even with a very small number of attackers (Fung et al., 2018). In this attack, the attacker flips the labels of training data in one class to another class and trains the model accordingly. Bagdasaryan et al. (2018) propose a backdoor attack so that the global model behaves incorrectly on adversarial targeted input. In our work, we mainly focus on defending against label-flipping attack, backdoor, and Gaussian noise attacks. Note that an attacker can perform any type of attacks, such as modifying any model values and training the local model on poisoned data for arbitrary epochs.

Robust distributed learning. Statistical methods have been studied and applied in robust distributed learning where data is IID (Feng et al., 2014; Blanchard et al., 2017; Chen et al., 2017; Yin et al., 2018; Alistarh et al., 2018). The median method and the trimmed mean method (Yin et al., 2018) are effective approaches in robust distributed learning, but may not be attack-resistant in federated learning where data distribution is non-IID. To tackle the challenge in robust federated learning, we propose a reweighted aggregation algorithm that dynamically assigns weights to the local model based on the residual to a regression line estimated by the repeated median estimator (Siegel, 1982).

Defending federated learning. Recently, some researchers have proposed some defense strategies for robust federated learning (Fung et al., 2018; Blanchard et al., 2017). FoolsGold (Fung et al., 2018) is a defense mechanism against Sybil attacks by adjusting the learning rates of local models based on contribution similarity. The algorithm identifies grouped actions as Sybil attacks and promotes the diversity of local model update. However, FoolsGold may identify harmless participants as attackers when these participants have similar local data. Gu et al. (2018) proposed a model, CalTrain, that represents data with fingerprints to identify poisoned data and models. Konstantinov and Lampert (2019) propose to maintain a small reference dataset to justify the quality and accountability of models. While this method is effective, it requires a lot of time to evaluate each model in every single round. Our algorithm does not need an additional reference dataset before or after each aggregation process.

Some researchers proposed to improve the privacy preservation of federated learning (Bonawitz et al., 2017; Geyer et al., 2017; Truex et al., 2018; Thakkar et al., 2019). Bonawitz et al. (2017) propose a privacy-preserving protocol for model aggregation in federated learning. Geyer et al. (2017) introduce differential privacy into federated learning. Instead of enhancing privacy preservation, we focus on the robustness of federated learning so that the global model should behave correctly even when there is a large portion of malicious participants.

3 Our Algorithm

In federated learning, there are multiple rounds of communication between participants and a central server for learning a global model. In each round, the global model is shared among the K participants and a local model on each device is trained on its local private data with the shared global model as initialization. Then all the K local models are sent to the central server to update the global model with an aggregation algorithm. The original aggregation scheme uses a simple averaging algorithm to aggregate all the local models (McMahan et al., 2017). Suppose the participant k has a local model $\boldsymbol{M^{(k)}}$, and we can update the global model $\boldsymbol{M_{global}}$ by taking the (weighted) average of all the K local models.

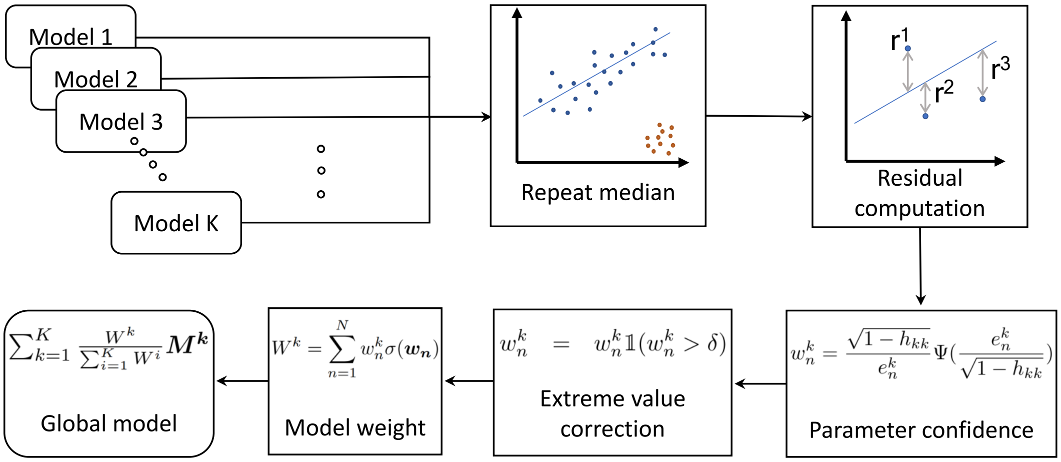

Model 1
Model 2
Model 3
Model K
Repeat median
r^1
r^2
r^3
Residual computation
$w_n^k = \frac{\sqrt{1-h_{kk}}}{e_n^k}\Psi(\frac{e_n^k}{\sqrt{1-h_{kk}}})$
Parameter confidence
$w_n^k = w_n^k \mathbb{1}(w_n^k > \delta)$
Extreme value correction
$W^k = \sum_{n=1}^{N} w_n^k \sigma(\boldsymbol{w_n})$
Model weight
$\sum_{k=1}^{K} \frac{W^k}{\sum_{i=1}^{K} W^i} \boldsymbol{M^k}$
Global model

Figure 1: The overview of our aggregation algorithm for attack-resistant federated learning.

3.1 Aggregation Algorithm

Median is a robust estimator widely used in statistics. However, when the data distribution is non-IID, median neglects a significant amount of information by merely taking a single median value. Hence, in our aggregation algorithm, the global model is designed to be a reweighted average of all the local models where the model weights are estimated robustly.

Algorithm 1 summaries our aggregation algorithm, and a detailed step-by-step description is provided below. We perform a weighted average of all the local models at the model level by assigning a weight to each local model. The weight of each local model is computed by accumulating the parameter confidence in the local model. The parameter confidence is computed based on the residual to a regression line estimated by the repeated median estimator (Siegel, 1982; Rand R, 1997). Inspired by the reweighting scheme in IRLS (Rand R, 1997), we reweight each parameter by its vertical distance (residual) to a robust regression line. For the robust regression line estimation, we use the repeated median estimator (Siegel, 1982) since it has a high breakdown point of 50%.

Let $y_n^{(k)}$ be the n-th parameter of the k-th local model. We use $\boldsymbol{y_n}$ to indicate the list of n-th parameters in all the local models. Let $\boldsymbol{x_n}$ be the indices of $\boldsymbol{y_n}$ sorted in an ascending order. Then $(\boldsymbol{x_n}, \boldsymbol{y_n})$ is a point set in 2D with increasing values in the y direction.

Repeated median. We use the repeated median estimator (Siegel, 1982) to estimate a linear regression line $y = \beta_{n0} + \beta_{n1}x$. The slope β_{n1} and intercept β_{n0} are estimated as follow,

$$\beta_{n1} = \underset{i}{\text{median}} \underset{i \neq j}{\text{median}} \frac{y_n^{(j)} - y_n^{(i)}}{x_n^{(j)} - x_n^{(i)}}, \quad i, j \in \{1, 2, ..., K\} \tag{1}$$

$$\beta_{n0} = \underset{i}{\text{median}} \underset{i \neq j}{\text{median}} \frac{x_n^{(j)} y_n^{(i)} - x_n^{(i)} y_n^{(j)}}{x_n^{(j)} - x_n^{(i)}}, \quad i, j \in \{1, 2, ..., K\} \tag{2}$$

Residual computation. We can calculate the residuals of the n-th parameters in all the local models:

$$\boldsymbol{r_n} = \boldsymbol{y_n} - \beta_{n0} - \beta_{n1}\boldsymbol{x_n}.$$

Since $\boldsymbol{r_n}$ can be very different in magnitude for different parameters, we can normalize $\boldsymbol{r_n}$ similar to the reweighting scheme in IRLS (Rand R, 1997):

$$\tau_n = \gamma \widetilde{|r_n|}(1 + \frac{5}{K-1}), \tag{3}$$

$$\widetilde{|r_n|} = \text{median}(|\boldsymbol{r_n}|). \tag{4}$$

where γ is a constant. We set $\gamma = 1.48$ recommended by Wilcox et al. (Rand R, 1997). Then the normalized residuals become

$$e_n^{(k)} = \frac{r_n^{(k)}}{\tau_n}. \tag{5}$$

Algorithm 1 Our aggregation algorithm

```
  Input: Models $\boldsymbol{M^{(1)}}, \boldsymbol{M^{(2)}}, ..., \boldsymbol{M^{(K)}}$, with parameters $y_n^{(1)}, y_n^{(2)}, ..., y_n^{(K)}$
  Output: The global model $\boldsymbol{M_{global}}$
for $n$-th parameter where $n = 1 \rightarrow N$, and let $\boldsymbol{y_n} = [y_n^{(1)}, y_n^{(2)}, ..., y_n^{(K)}]^T$ do
    $\boldsymbol{x_n} \leftarrow$ indices of $\boldsymbol{y_n}$ sorted in an ascending order    ▷ assign indices
    $\beta_{n0}, \beta_{n1} \leftarrow RepeatedMedian(\boldsymbol{x_n}, \boldsymbol{y_n})$    ▷ get robust line estimation
    $\boldsymbol{r_n} \leftarrow \boldsymbol{y_n} - \beta_{n0} - \beta_{n1}\boldsymbol{x_n}$    ▷ compute residual
    $\widetilde{|r_n|} \leftarrow Median|\boldsymbol{r_n}|$
    $\tau_n \leftarrow \gamma\widetilde{|r_n|}(1 + \frac{5}{K-1})$    ▷ normalize residuals
    $\boldsymbol{e_n} \leftarrow \frac{\boldsymbol{r_n}}{\tau_n}$
    $\boldsymbol{H_n} \leftarrow \boldsymbol{x_n}(\boldsymbol{x_n^T x_n})^{-1}\boldsymbol{x_n^T}$    ▷ compute Hat matrix
    $\boldsymbol{w_n} \leftarrow \frac{\sqrt{\boldsymbol{1}-diag(\boldsymbol{H_n})}}{\boldsymbol{e_n}}\Psi(\frac{\boldsymbol{e_n}}{\sqrt{\boldsymbol{1}-diag(\boldsymbol{H_n})}})$    ▷ compute parameter confidence
    $\boldsymbol{y_n}, \boldsymbol{w_n} \leftarrow$ CorrectExtremeValue($\boldsymbol{y_n}, \boldsymbol{w_n}$)
    $\boldsymbol{w_n} \leftarrow \boldsymbol{w_n}\sigma(\boldsymbol{w_n})$    ▷ reweight confidence by its standard deviation
for each $k = 1 \rightarrow K$ do
    $W^{(k)} \leftarrow \sum_{n=1}^{N} \boldsymbol{w_n^{(k)}}$    ▷ accumulate weights
$\boldsymbol{M_{global}} \leftarrow \sum_{k=1}^{K} \frac{W^{(k)}}{\sum_{i=1}^{K} W^{(i)}} \boldsymbol{M^{(k)}}$    ▷ the global model aggregation
```

Parameter confidence. After obtaining the normalized residuals, the parameter confidence can be determined accordingly (Rand R, 1997):

$$w_n^{(k)} = \frac{\sqrt{1-h_{kk}}}{e_n^{(k)}}\Psi(\frac{e_n^{(k)}}{\sqrt{1-h_{kk}}}), \tag{6}$$

where $w_n^{(k)}$ is the confidence of the n-th parameter in $M^{(k)}$, $\Psi(x) = max\{-Z, min(Z, x)\}$ with $Z = \lambda\sqrt{2/K}$ and λ is a hyperparameter. We use $\lambda = 2$ in our experiments. Ψ here acts as a trusted interval and we can expand or shrink the interval by tuning λ. h_{kk} is the k-th diagonal of matrix in H_n:

$$\boldsymbol{H_n} = \boldsymbol{x_n}(\boldsymbol{x_n^T x_n})^{-1}\boldsymbol{x_n^T}, \tag{7}$$

where $\boldsymbol{x_n} = [x_n^{(1)}\ x_n^{(2)}\ \ldots\ x_n^{(K)}]^T$.

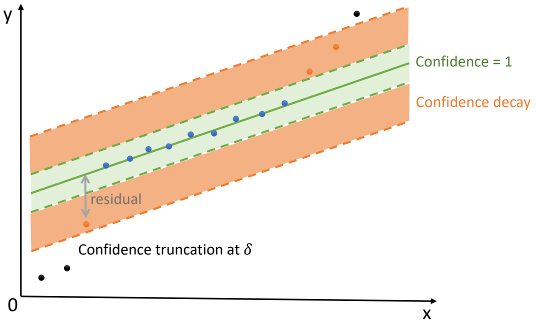


Figure 2: Parameter confidence assignment based on the residual which is the distance from a point to the regression line. In the green area, the parameter confidence is 1; in the orange area, the confidence decays from 1 to δ; in other areas, the confidence is set to 0.

Extreme value correction. Extremely large values, even multiplied with a small weight in model aggregation, can damage the global model. We address this issue by involving a a threshold δ. If a parameter has a confidence value lower than δ, then it should be corrected as follows,

$$w_n^{(k)} = w_n^{(k)}\mathbb{1}(w_n^{(k)} > \delta), \tag{8}$$

$$y_n^{(k)} = y_n^{(k)}\mathbb{1}(w_n^{(k)} > \delta) + (\beta_{n0} + \beta_{n1}x_n^{(k)})\mathbb{1}(w_n^{(k)} \leq \delta). \tag{9}$$

The process of selecting proper δ is discussed in the appendix.

Model weight. To obtain the weight of each local model, we can simply aggregate the parameter confidence in the local model but this is not ideal. Imagine an attacker trains a model honestly, but then alters only 10% of the parameters to some extremely large values. This adversary model still receives about 90% of the parameter confidence. To address this problem, we measure the importance of a parameter by the standard deviation of $\boldsymbol{w_n}$. A confidence assignment with a large standard deviation indicates a great disagreement among this parameter in all models and should be more critical when being accumulated towards model weights:

$$W^{(k)} = \sum_{n=1}^{N} w_n^{(k)}\sigma(\boldsymbol{w_n}), \tag{10}$$

where $W^{(k)}$ is the weight for model k, N is the number of parameters, $\boldsymbol{w_n} = [w_n^{(1)} w_n^{(2)} \dots w_n^{(K)}]^T$.

Global model. Finally, we can obtain the updated global model by

$$\boldsymbol{M_{global}} = \sum_{k=1}^{K} \frac{W^{(k)}}{\sum_{i=1}^{K} W^{(i)}} \boldsymbol{M^{(k)}}. \tag{11}$$

3.2 Theoretical Guarantee

For simplicity, we consider the bound of the error rate for training a single-parameter model on K devices, each storing S IID samples of data. We adopt the same set of assumptions from Yin et al. (2018), assuming that the loss function is L-smooth and the derivatives of the loss function are v-sub-exponential. Suppose that there are K devices and U of them are corrupted. We denote the set of adversarial devices as $\mathcal{B}$ and the corruption ratio $\alpha = \frac{U}{K}$. We define the parameter of the local model i as $\hat{y}^{(i)}$ and let μ be the expected value of the global model.

Based on our algorithm, the residuals can be simplified as $(\hat{y}^{(i)} - \tilde{y})$, where $\tilde{y}$ is the median of $\{\hat{y}^{(i)}\}$ for $i = 1, 2, ..., K$. Let $\widetilde{|r|}$ be the median of absolute residuals, i.e., $\widetilde{|r|} = \text{median}(|\hat{y}^{(i)} - \tilde{y}|)$. Then, the normalized residual can be expressed as

$$e^{(i)} = \frac{\hat{y}^{(i)} - \tilde{y}}{\gamma \widetilde{|r|}(1 + \frac{5}{K-1})}, \tag{12}$$

and the parameter confidence is defined as

$$z^{(i)} = \begin{cases} 1, & \text{if } \left|e^{(i)}\right| \leq \dfrac{\sqrt{2}\lambda}{\sqrt{K}} \\ \left|\dfrac{\sqrt{2}\lambda}{\sqrt{K} e^{(i)}}\right|, & \text{if } \dfrac{\sqrt{2}\lambda}{\sqrt{K}} < \left|e^{(i)}\right| \leq \left|\dfrac{\sqrt{2}\lambda}{\sqrt{K}\delta}\right| \\ 0, & \text{if } \left|e^{(i)}\right| > \left|\dfrac{\sqrt{2}\lambda}{\sqrt{K}\delta}\right| \end{cases} \tag{13}$$

Then we will prove that the error of the global model $\boldsymbol{M_{global}} = \frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i=1}^{K} z^{(i)} \hat{y}^{(i)}$ is bounded:

$$\left| \frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i=1}^{K} z^{(i)} \hat{y}^{(i)} - \mu \right| = \tilde{\mathcal{O}}(\frac{\alpha}{\sqrt{S}} + \frac{1}{\sqrt{SK}} + \frac{1}{S} + \frac{1}{\sqrt{K}\delta}). \tag{14}$$

The details of the proof is presented in the appendix.

4 Experiments

We compare our approach with other aggregation algorithms, including FedAvg (McMahan et al., 2017), the coordinate-wise median method (Yin et al., 2018), the coordinate-wise trimmed mean method (Yin et al., 2018), FoolsGold (Fung et al., 2018), and a coordinate-wise repeated median approach we adopt from (Siegel, 1982). We perform experiments on the MNIST handwritten digit dataset (LeCun et al., 1998), the Amazon Reviews dataset (Ruining and Julian, 2016), the CIFAR-10 dataset (Krizhevsky et al., 2009), and the Lending Club loan dataset (Kan, 2019). We implement attack strategies and defense algorithms in PyTorch (Paszke et al., 2017).

We use a two-layer convolutional neural network (CNN) for our MNIST experiments and the network architecture is shown in the appendix. With this simple CNN model, our goal is to evaluate different aggregation algorithms for defending federated learning in the presence of attacks. On the CIFAR-10 dataset, we use ResNet-18 (He et al., 2015) for image classification. The text classification model *FastText* (Joulin et al., 2016) is adopted for evaluation on the Amazon Reviews dataset. It is a two-layer deep neural network where the first layer is an embedding layer, and the second layer is a fully connected layer. For the Lending Club loan dataset, we use a simple neural network with three fully-connected layers to classify loan status. We use the last two models to demonstrate that our algorithm can be generalized to a natural language processing task and to a real-work financial problem. All the evaluation results are the average of running the same experiments 3 times.

4.1 Datasets and Experimental

MNIST dataset. The MNIST dataset contains 70,000 real-world handwritten images with digits from 0 to 9. We evaluate different methods by learning a global model on these training images distributed on multiple devices in a non-IID setting with adversarial attacks.

CIFAR-10 dataset. The CIFAR-10 dataset contains 60,000 natural images in ten object classes. The experimental setup is also non-IID on CIFAR-10.

Amazon Reviews dataset. The Amazon Reviews dataset (Ruining and Julian, 2016) contains product reviews and ratings collected from the Amazon website. Every review is paired with a sentiment rating from 1 to 5. We categorize comments with rating 1 or 2 as negative and comments with rating 4 or 5 as positive. We discard reviews with rating 3, and we only train a binary classifier. We only use the book reviews from the Kindle Store. 20% of the reviews are used for testing, while the rest is for training. We obtain a training set of 86,164 reviews and a test set of 13,260 reviews.

Lending Club Loan dataset. The Lending club dataset LOAN (Kan, 2019) contains financial information such as credit scores and the number of finance inquiries for loan status prediction ("Current", "Late", or "Fully Paid"). There are 1,808,534 data samples in 9 types of loan status. We divide them by US states to simulate the federated learning scenarios with non-IID data distribution where each state represents a participant.

4.2 Results on Label-flipping Attacks

We evaluate the overall classification performance of different aggregation methods on three datasets under label-flipping attacks, the MNIST dataset (Biggio et al., 2012), the CIFAR-10 dataset (Krizhevsky et al., 2009), and Amazon Reviews dataset (Ruining and Julian, 2016). In label-flipping attacks, attackers flip the labels of training examples in the source class to a target class and train their models accordingly.

In the MNIST experiment, we simulate federated learning with 100 participants, within which 0 to 10 of them are attackers. Each participant contains images of two random digits. The attackers are chosen to be some participants with images of digit 1 and another random digit since they are flipping the label of 1 to 7. We run 200 synchronization rounds with δ set to 0.01. In each round of federated learning, each participant is supposed to train the local model for 5 epochs, but the attackers can train for arbitrary epochs. The results are shown in Figure 3 where attackers train the models with 5 more epochs to enhance the attacks. Our algorithm outperforms all other methods and is robust when the number of attackers increases. Median methods (Median and Repeated Median) have relatively low accuracy due to discarding most of the information in model aggregation. Our algorithm, on the other hand, takes the reweighted average of all local models and thus gathers more information in an unsupervised way. We also compare our algorithm with the state-of-the-art algorithm FoolsGold (Fung et al., 2018). Though their algorithm also maintains a low attack success rate, our algorithm is more stable and surpasses FoolsGold by 6% on average.

For the CIFAR-10 experiment, following Bagdasaryan et al. (2018), we adopt a Dirichlet distribution (Minka, 2000) with a hyperparameter 0.9 to generate non-IID data distribution for totally 10 participants. The experimental setup is the same for the CIFAR-10 and MNIST experiments under backdoor attacks in Section 4.3. The attackers flip the label of "cat" to "dog" since they are the most similar classes in CIFAR-10. The experiment results can be found in Table 1. FoolsGold (Fung et al., 2018) fails in this experiment because honest participants can be classified as attackers when two or more honest participants have similar data distribution.

In the experiment on Amazon Reviews, there are 10 participants, where 0 to 4 participants are attackers who flip all their labels. Attackers also train for 5 more epochs. We run 10 synchronization rounds in the experiment. The result is summarized in Table 2. Our algorithm achieves comparable state-of-the-art results, and our performance does not degrade when there are less than 50% attackers.

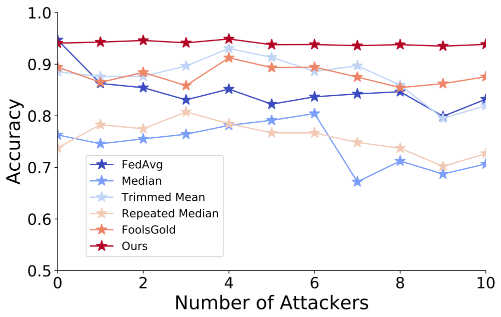


Figure 3: Results of label-flipping attacks on the MNIST dataset. The number of participants is 100, within which 0 to 10 of them are attackers.

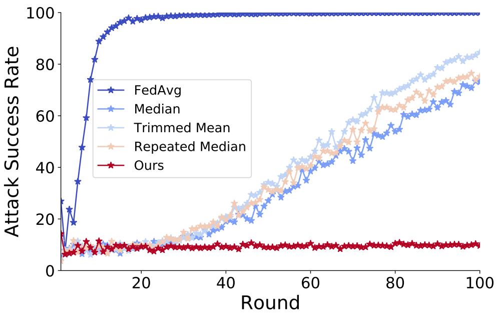


Figure 4: Result of backdoor attack success rate on CIFAR-10 under different aggregation algorithms. Ours outperforms other baselines.

# of attackers	0	1	2	3	4	Average
FedAvg (McMahan et al., 2017)	88.96%	85.74%	82.49%	82.35%	82.11%	84.33%
Median (Yin et al., 2018)	88.11%	87.69%	87.15%	85.85%	82.01%	86.16%
Trimmed Mean (Yin et al., 2018)	88.70%	88.52%	**87.44%**	85.36%	82.35%	86.47%
Repeated Median (Siegel, 1982)	88.60%	87.76%	86.97%	85.77%	81.82%	86.19%
FoolsGold (Fung et al., 2018)	9.70%	9.57%	10.72%	11.42%	9.98%	10.28%
Ours	**89.17%**	**88.60%**	86.66%	**86.09%**	**85.81%**	**87.27%**

Table 1: Results of label-flipping attacks on CIFAR-10 dataset with different numbers of attackers. The total number of participants is 10.

# of attackers	0	1	2	3	4	Average
FedAvg (McMahan et al., 2017)	91.81%	86.91%	24.97%	12.52%	9.78%	45.20%
Median (Yin et al., 2018)	91.73%	91.87%	91.79%	91.43%	91.17%	91.60%
Trimmed Mean (Yin et al., 2018)	91.81%	91.82%	91.82%	91.49%	91.26%	91.64%
Repeated Median (Siegel, 1982)	91.55%	88.41%	23.22%	11.70%	9.62%	44.90%
FoolsGold (Fung et al., 2018)	50.79%	49.45%	47.44%	49.71%	49.95%	49.47%
Ours	91.71%	91.79%	91.76%	91.67%	91.38%	**91.66%**

Table 2: Results of label-flipping attacks on Amazon Reviews dataset with different numbers of attackers. The total number of participants is 10.

4.3 Results on Backdoor Attacks

For pixel-pattern backdoor attacks (Gu et al., 2019) in federated learning(Bagdasaryan et al., 2018), attackers manipulate their local models so that the learned global model predicts some backdoor target label for any image embedded with certain patterns. An example is shown in Figure 5 in the Appendix. The global model can behave normally for clean data. We choose "bird" in CIFAR-10 and "2" in MNIST as the backdoor target labels. Similarly, for the preprocessed LOAN dataset that contains 92 features, we manipulate 6 features by assigning certain large values to them and change the labels of manipulated data to "Does not meet the credit policy. Status:Fully Paid." The training data is mixed with manipulated data and clean data to fit both the backdoor task and the main task. We compare the performance of aggregation algorithms under two backdoor attack scenarios, which are called the naive approach and the model replacement in Bagdasaryan et al. (2018). For the naive approach, an attacker poisons its local model and submits the malicious update in every round. For the model replacement, an attacker only poisons in one round to embed some patterns into the global model, so the attacker needs to scale up its malicious update before submission. In our experiment, the malicious participant attacks in round 6 and scales up its update by 100. We run 200 rounds for MNIST and 100 rounds for CIFAR and LOAN.

Table 3 summarizes the results of backdoor attacks. Our method is the highest in terms of accuracy on MNIST under both backdoor attack scenarios. Moreover, on the more challenging CIFAR-10 dataset,

	MNIST				CIFAR-10			
Dataset	Naive approach		Model replacement		Naive approach		Model replacement	
	Acc.	A.S.R	Acc.	A.S.R	Acc.	A.S.R	Acc.	A.S.R
FedAvg	99.08%	99.71%	98.75%	17.85%	87.44%	99.91%	69.72%	38.59%
Median	98.91%	10.34%	98.87%	10.35%	88.58%	73.06%	87.22%	10.01%
Trimmed Mean	98.97%	10.34%	98.81%	10.34%	88.38%	84.56%	87.30%	9.85%
Repeated Median	98.96%	10.36%	98.82%	10.32%	88.22%	75.25%	87.57%	9.79%
FoolsGold	96.20%	12.51%	97.96%	10.27%	10.00%	0.00%	10.00%	0.00%
Our	**98.97%**	10.35%	**98.88%**	10.31%	**88.89%**	**9.65%**	87.43%	9.56%

Table 3: Results of backdoor attacks on MNIST and CIFAR-10. There are 10 participants, 1 of whom is an attacker. We denote the accuracy as Acc. and the attack success rate as A.S.R.

Standard deviation	0%	100%	200%	300%	Average
FedAvg (McMahan et al., 2017)	**93.17%**	78.80%	53.74%	9.26%	58.74%
Median (Yin et al., 2018)	73.92%	64.35%	71.54%	63.24%	68.26%
Trimmed Mean (Yin et al., 2018)	87.59%	74.87%	73.76%	75.74%	77.99%
Repeated Median (Siegel, 1982)	61.86%	75.53%	67.76%	73.63%	69.69%
FoolsGold (Fung et al., 2018)	86.72%	90.31%	86.96%	89.94%	88.48%
Ours	90.06%	**90.91%**	**89.06%**	**92.28%**	**90.58%**

Table 4: Results of Gaussian noise attacks on the MNIST dataset when ϵ is sampled from a Gaussian distribution with different standard deviations. There are 100 participants where 10 are attackers.

our algorithm is the only one that can defend the naive approach backdoor attack. In Figure 4, we plot the attack success rates over time under different aggregation algorithms except for FoolsGold because it completely fails in both main and backdoor tasks. Intuitively, backdoor attacks can easily succeed under FedAvg, and other baselines slow down the process but still reach high attack success rate into over 70% within 100 rounds. Our algorithm effectively defends the attack and remains stable with 9.65% attack success rate when being attacked continuously for 100 rounds. On the LOAN dataset, our method achieves higher accuracy 94.50% (FedAvg 93.65%) and 0.00% attack success rate (FedAvg 99.71%) under the naive approach attack after 100 rounds. Similarly, our method has 95.06% accuracy (FedAvg 94.11%) and 0.00% attack success rate (FedAvg 98.96%) under model replacement attacks after 100 rounds. Other baselines also have 0.00% attack success rate in two attacking scenarios expect FoolsGold, that of which is 99.96% under the naive approach attack.

4.4 Results on Gaussian Noise Attacks

In Gaussian noise attacks, We try to simulate real-world model corruption with 10 corrupted users at different scales of noise. Gaussian noise attacks are performed by multiplying each parameter by a scale ϵ sampled from a Gaussian distribution with a mean of 1. In the experiment on MNIST, we have 100 participants where some may be attackers. Each participant has 300 images from a random digit and other 300 images from another random digit. Table 4 summarizes the results when ϵ is sampled from a Gaussian distribution with different standard deviations and there are 10 attackers. Our aggregation algorithm outperforms other approaches under Gaussian noise at multiple scales.

5 Conclusion

Federated learning utilizes private data on multiple devices to train a global model, but the simple aggregation algorithm in federated learning is vulnerable to malicious attacks. To tackle this problem, we present a novel aggregation algorithm with residual reweighting. Our experiments on computer vision, natural language processing, and financial data show that our approach is robust to label-flipping, Gaussian noise, and backdoor attacks while prior aggregation methods are not. Our algorithm is easy to implement and readily incorporated into existing federated learning frameworks. We hope our proposed aggregation algorithm can make federated learning more practical and robust in the future. Our source code will be made public.

REFERENCES

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *POMACS*, 1(2):44:1–44:25, 2017.

Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.

Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

Andrew F Siegel. Robust regression using repeated medians. *Biometrika*, 69(1):242–244, 1982.

Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977.

Wilcox Rand R. *Introduction to Robust Estimation and Hypothesis Testing*. Elsevier, 1997.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

He Ruining and McAuley Julian. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. 2016.

Wendy Kan. Lending club loan data, Mar 2019. URL `https://www.kaggle.com/wendykan/lending-club-loan-data`.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. *arXiv preprint arXiv:1812.00535*, 2018.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. *Proceedings on Privacy Enhancing Technologies*, 2019(1):133–152, 2019.

Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*, 2017.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy*, 2019.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

Jiashi Feng, Huan Xu, and Shie Mannor. Distributed robust learning. *CoRR*, abs/1409.5937, 2014.

Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 2018.

Zhongshu Gu, Hani Jamjoom, Dong Su, Heqing Huang, Jialong Zhang, Tengfei Ma, Dimitrios Pendarakis, and Ian Molloy. Reaching data confidentiality and model accountability on the caltrain. *arXiv preprint arXiv:1812.03230*, 2018.

Nikola Konstantinov and Christoph Lampert. Robust learning from untrusted sources. *arXiv preprint arXiv:1901.10310*, 2019.

Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, and Rui Zhang. A hybrid approach to privacy-preserving federated learning. *arXiv preprint arXiv:1812.03224*, 2018.

Om Thakkar, Galen Andrew, and H. Brendan McMahan. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Thomas Minka. Estimating a dirichlet distribution, 2000.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.pdf*, 2014.

A Appendix

A.1 Proof

Proof. We can separate Equation 14 into two sets with adversarial participant $\mathcal{B}$ and normal users $[K] \setminus \mathcal{B}$ where $[K] = \{1, 2, \dots, K\}$.

$$\begin{aligned}
\left| \frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i=1}^{K} z^{(i)} \hat{y}^{(i)} - \mu \right| &\leq \frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i=1}^{K} z^{(i)} \left| \hat{y}^{(i)} - \mu \right| \\
&\leq \frac{1}{\sum_{j=1}^{K} z^{(j)}} \left(\sum_{i \in [K] \setminus \mathcal{B}} z^{(i)} \left| \hat{y}^{(i)} - \mu \right| + \sum_{i \in \mathcal{B}} z^{(i)} \left| \hat{y}^{(i)} - \tilde{y} + \tilde{y} - \mu \right| \right) \\
&\leq \max_{i \in [K] \setminus \mathcal{B}} \{ \left| \hat{y}^{(i)} - \mu \right| \} + \frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i \in \mathcal{B}} z^{(i)} \left| \hat{y}^{(i)} - \tilde{y} \right| + \left| \tilde{y} - \mu \right| .
\end{aligned} \tag{15}$$

Yin et al. (2018) prove that the first term of Equation 15, $\max_{i \in [K] \setminus \mathcal{B}} \{ \left| \hat{y}^{(i)} - \mu \right| \} = \tilde{\mathcal{O}}(\frac{\alpha}{\sqrt{S}} + \frac{1}{\sqrt{SK}})$ by Bernstein's inequality and union bound. They also prove the third term $|\tilde{y} - \mu|$ is bounded:

$$|\tilde{y} - \mu| = \tilde{\mathcal{O}}(\frac{\alpha}{\sqrt{S}} + \frac{1}{\sqrt{SK}} + \frac{1}{S}). \tag{16}$$

Now let us consider $\frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i \in \mathcal{B}} z^{(i)} \left| \hat{y}^{(i)} - \tilde{y} \right|$. For an attacker $a \in \mathcal{B}$, there are two cases:

- $|e^a| \leq \frac{\sqrt{2}\lambda}{\sqrt{K}}$. From Equation 12 and $|e^a| \leq \frac{\sqrt{2}\lambda}{\sqrt{K}}$, we have $\left| \hat{y}^{(i)} - \tilde{y} \right| \leq \frac{C}{\sqrt{K}}$, where $C = \gamma \sqrt{2} \lambda \widetilde{|r|} (1 + \frac{5}{K-1})$.
- $|e^a| > \frac{\sqrt{2}\lambda}{\sqrt{K}}$. If $|e^a| > \left| \frac{\sqrt{2}\lambda}{\sqrt{K}\delta} \right|$ then it can be eliminated from Equation 13. From Equation 12 and $|e^a| > \frac{\sqrt{2}\lambda}{\sqrt{K}}$, we have $\frac{C}{\sqrt{K}} < \left| \hat{y}^{(i)} - \tilde{y} \right| \leq \frac{C}{\sqrt{K}\delta}$.

Combine two cases, we know that $\left| \hat{y}^{(i)} - \tilde{y} \right| \leq \frac{C}{\sqrt{K}\delta}$, and we have the following inequality,

$$\frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i \in \mathcal{B}} z^{(i)} \left| \hat{y}^{(i)} - \tilde{y} \right| \leq \sup_{i \in \mathcal{B}} \left| \hat{y}^{(i)} - \tilde{y} \right| \leq \frac{C}{\sqrt{K}\delta}. \tag{17}$$

Therefore, we prove that

$$\left| \frac{1}{\sum_{j=1}^{K} z^{(j)}} \sum_{i=1}^{K} z^{(i)} \hat{y}^{(i)} - \mu \right| = \tilde{\mathcal{O}}(\frac{\alpha}{\sqrt{S}} + \frac{1}{\sqrt{SK}} + \frac{1}{S} + \frac{1}{\sqrt{K}\delta}). \tag{18}$$

□

A.2 Model Details

The models we use in the experiments are described here. The CNN model used for classifying MNIST images (LeCun et al., 1998) can be found in Table 5. CIFAR-10 is trained with Resnet-18(He et al., 2015). The learning rate for both MNIST classifier and CIFAR-10 classifier is 0.01 with a SGD optimizer. We train MNIST classifier for 200 synchronous rounds and CIFAR-10 classifier for 100 rounds, with batch size 128 and 64 respectively. The Amazon Reviews (Ruining and Julian, 2016) classifier FastText (Joulin et al., 2016) is a two-layer deep neural network where the first layer is an embedding layer of dimension 100 and the second layer is a fully connected layer of dimension 1. We use binary cross entropy loss to train the model with Adam optimizer (Kingma and Ba, 2014) and learning rate 0.05. The input dimension (vocabulary size) of the model is 25,000. For the preprocessed LOAN dataset, the input data consists of 91 features. We use a simple neural network with three fully-connected layers whose feature channels are 46, 23, and 9 respectively.

Layer	# of Channels	Kernel size	Image size
Conv1	4	5x5	24x24
Max-pool	4		12x12
Conv1	8	5x5	8x8
Max-pool	8		4x4
fc1	16	16	
fc2	10	10	

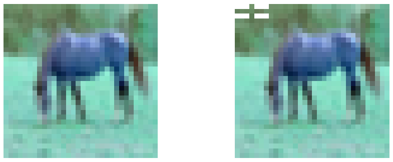
Table 5: The CNN architecture for MNIST classification.

A.3 Hyperparameters

Our algorithm has two hyperparameters, λ and δ. λ is used to control the confidence interval, and δ is used to constrain the lower bound of the confidence. λ is insensitive to changes. We use $= 2$ in all the experiments. δ, however, is more sensitive to the data distribution. While the data is close to IID (on the Amazon Reviews dataset), a large δ (such as 0.1) yields satisfactory results as the parameters among models are more similar. If the data distribution is non-IID (on the MNIST dataset), a smaller δ (such as 0.01) is preferable.

A.4 Backdoor Experimental Setup

Experiment setup of backdoor attacks is provided here. There are 10 total participants each round where one of them is the attacker. In each round of federated learning, benign participants train their local models for 5 epochs and attackers train for 10 epochs. Benign participants always use a learning rate of 0.01. In MNIST and CIFAR, the attacker uses a learning rate of 0.1, and in LOAN, the attacker uses 0.002. The batch size is 64 for all participants. The attacker poisons 20 samples per batch in the training process. An example of backdoor patterns we use for image datasets is shown in Figure 5. For the LOAN dataset, we assign certain values that are slightly larger than their maximum value to the six features (num_tl_120dpd_2m, num_tl_90g_dpd_24m, pub_rec_bankruptcies, pub_rec, acc_now_delinq, tax_liens).



(a) Clean image (b) Backdoored image

Figure 5: A backdoored image. There is a white color pattern in the left upper corner.