Robust Federated Learning with Attack-Adaptive Aggregation

Ching Pui Wan , Qifeng Chen
The Hong Kong University of Science and Technology
{cpwan, cqf}@ust.hk

Abstract

Federated learning is vulnerable to various attacks, such as model poisoning and backdoor attacks, even if some existing defense strategies are used. To address this challenge, we propose an attack-adaptive aggregation strategy to defend against various attacks for robust federated learning. The proposed approach is based on training a neural network with an attention mechanism that learns the vulnerability of federated learning models from a set of plausible attacks. To the best of our knowledge, our aggregation strategy is the first one that can be adapted to defend against various attacks in a data-driven fashion. Our approach has achieved competitive performance in defending model poisoning and backdoor attacks in federated learning tasks on image and text datasets.

1 Introduction

Federated learning allows multiple clients to collectively train a neural network without directly sharing their own private data [McMahan *et al.*, 2017; Smith *et al.*, 2017]. The federated learning framework has been proposed for diverse applications such as mobile applications, healthcare, and financial assessment [Yang *et al.*, 2019]. Despite the large potential of federated learning in real-life applications, it is vulnerable to numerous attacks, including data poisoning and model poisoning [Blanchard *et al.*, 2017; Bagdasaryan *et al.*, 2020; Xie *et al.*, 2019a]. Can we design an attack-adaptive defense strategy for robust federated learning?

In federated learning,the attackers may control a fraction of clients and manipulate the local data and the model updates to inject a backdoor or to degrade the global model's performance. For example, with the backdoor attack [Bagdasaryan *et al.*, 2020], the attacker can locally assign a 'trash' label to the images of the automobiles manufactured by a certain brand and contaminate the global model. Therefore, it is important to defend the attacks for robust federated learning training.

From the server's perspective, the only clue for defending the adversarial attacks is the model updates submitted from the clients, in contrast to the attackers' large flexibility in designing attacks. Also, the heterogeneous data distribution (non-identically distributed data) in federated learning make the problem more challenging. Hence, the key of the defense is on designing a robust aggregation strategy for the model updates. In an aggregation strategy, we treat the model update as a vector, and we want to discard the corrupted update vectors from the attackers while keeping only the genuine update vectors from the benign clients.

Several aggregation rules, instead of FedAvg [McMahan *et al.*, 2017], have been proposed for defending the adversarial attacks. Classical robust estimators, such as Coordinate-wise median [Yin *et al.*, 2018; Chen *et al.*, 2020] and Geometric median (implemented as RFA in [Pillutla *et al.*, 2019]) have been proposed but their performance degraded due to the heterogeneous data distribution in federated learning. Residual-based reweighing [Fu *et al.*, 2019] extends the classical robust regression to the federated learning setting, but it has a low breakdown point. On the other hand, several similarity-based aggregation rules have been proposed. FoolsGold [Fung *et al.*, 2020] asserts the similarity of attackers, Krum and its variants [Blanchard *et al.*, 2017; Mhamdi *et al.*, 2018] assert the similarity of the benign clients (in terms of Euclidean distance), clustering-based approaches [Sattler *et al.*, 2020; Muñoz-González *et al.*, 2019] assert both with cosine similarity. Nevertheless, these similarity-based defenses can be bypassed by projecting the corrupted update vectors to the neighborhood of the genuine update vectors [Baruch *et al.*, 2019; Bagdasaryan *et al.*, 2020]. Recent works try to uncover more properties of the attacker. WeakDP [Sun *et al.*, 2019] tries to cancel the effect of the attacker by clipping and adding noise to the update vectors. However, the optimal size of noise is not well studied. In [Li *et al.*, 2020], their approach detects attackers by learning a variational autoencoder on randomly sampled coordinates of the unbiased model updates obtained in the centralized training setting. However, the optimal latent representation of the model updates is not well studied. Apart from their weakness, the former defense strategies may fail to detect edge-case backdoor [Wang *et al.*, 2020], where a very small region of the model updates are altered. It indicated the need of a tailor-made defense for challenging attacks.

In this work, we propose the first attack-adaptive aggregation mechanism for robust federated learning that learns to detect possible corrupted update vectors from challenging attacks. Our approach learns a low dimensional representa-

tion of the update vectors that allows detection of possible attacks. Specifically, we train an attention [Vaswani *et al.*, 2017] based neural network to explore the vulnerable regions of the model with respect to various attacks. We feed the update vectors to the attention module to obtain the alignment scores between the latent representations and reweigh the update vectors accordingly. We simulate the federated learning tasks under different attacks with the test set in the server and collect the update vectors to train our model in a self-supervised fashion. We show the approximation capacity of our neural network on similarity measures.

We compare our approach with existing aggregation rules on federated learning tasks: MNIST [LeCun *et al.*, 1998] classification, CIFAR-10 [Krizhevsky *et al.*, 2009] classification, Tiny-ImageNet classification, and IMDb [Maas *et al.*, 2011] sentiment analysis. Our approach outperforms prior work in defending model poisoning and backdoor attacks. Our approach can also generalize the defense to different datasets, numbers of clients, and attack parameters.

2 Related work

2.1 Attacks on federated learning.

Adversarial attacks can attack either the data or the model. [Blanchard *et al.*, 2017] suggested the omniscient attack, which multiplies the update vector by a negative constant, can reverse the direction of gradient descent and degrade the model performance. [Fung *et al.*, 2020] suggested that the label flipping attack, which changes the label of a certain class, can already be an effective attack if there are no defenses. [Bagdasaryan *et al.*, 2020] showed that the backdoor attack , which injects a certain pattern to the data and alters the label to the desired target, can mislead the global model while not affecting the standard accuracy. [Xie *et al.*, 2019a] proposed the distributed backdoor attack, which embeds similar but different patterns to the data, to bypass similarity-based defenses. Some works [Baruch *et al.*, 2019; Bagdasaryan *et al.*, 2020] show that certain defenses can be bypassed by projecting the corrupted update vectors to the neighborhood of the genuine ones. In our work, we will show the potential of a data-driven and attack-adaptive aggregation strategy in defending adversarial attacks.

2.2 Robust federated learning.

The defense on federated learning can be categorized in terms of robustness, privacy, and security. In terms of robustness, several approaches defend adversarial attacks by designing aggregation rules as introduced in Section 1. There are defenses relating to other aspects of federated learning. [Sun *et al.*, 2019] suggested that adding the differential privacy can improve robustness against certain attacks. [Pillutla *et al.*, 2019] proposed RFA, a secure implementation of the Geometric median aggregation rule. On the other hand, some approaches sacrifice certain level of privacy for robustness. Zeno [Xie *et al.*, 2019b] audits the local models' accuracy on a test set in the server. [Wang *et al.*, 2020] compares the update vectors with the unbiased model updates trained on the test set. In our work, we will focus on robustness, and we assume that the server has access to the update vectors and a test set that is disjoint from the local data.

3 Method

3.1 Formulation

Federated learning. In federated learning, the server distributes a global model $\boldsymbol{\theta}_{global}$ to each of the n clients. Then each client i trains its local model $\boldsymbol{\theta}^{(i)}_{client}$ with its own data and sends the update vector

$$\boldsymbol{x}_i = \boldsymbol{\theta}^{(i)}_{client} - \boldsymbol{\theta}_{global}$$

back to the server. The server aggregates the set of update vectors $\{\boldsymbol{x}_i\}$ by the aggregation strategy $g(\{\boldsymbol{x}_i\})$ and updates the global model as

$$\boldsymbol{\theta}_{global} \leftarrow \boldsymbol{\theta}_{global} + g(\{\boldsymbol{x}_i\}).$$

The new global model is then distributed to the clients for the next round of training.

Robust aggregation strategy. Federated learning can be vulnerable to adversarial attacks. The attackers can attack their local data or the update vectors directly. One the other hand, the server knows only the clients' update vectors but not their local training data or even the number of samples trained locally. Hence, a robust aggregation strategy is key to defend the attacks. Let $\mathbb{D}_{benign}$ to be a set of genuine update vectors from benign clients and $\mathbb{D}_{attack}$ to be a set of the corrupted update vectors from attackers. We denote the mean of only the genuine update vectors as the **robust mean**

$$\boldsymbol{\mu}_{robust} = \sum_{i=1}^{n} \frac{\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}}{\sum_{j=1}^{n} \mathbf{1}_{(\boldsymbol{x}_j \in \mathbb{D}_{benign})}} \boldsymbol{x}_i, \tag{1}$$

where $\mathbf{1}_{(condition)}$ is the indicator function which evaluates to 1 if the condition is true and 0 otherwise. A robust aggregation strategy $g(\cdot)$ aims to approximate the robust mean $\boldsymbol{\mu}_{robust}$, i.e. solving the minimization

$$\arg\min_{g} \left\| g\left(\{\boldsymbol{x}_i\}\right) - \boldsymbol{\mu}_{robust} \right\|. \tag{2}$$

The difficulty of designing a robust aggregation strategy is that the attackers can evolve their attacks to bypass the current defense. Hence we are interested in an aggregation strategy that can readily be adapted to defend the challenging attacks. In Section 3.2, we will propose a data-driven framework for attack-adaptive aggregation.

3.2 Attack-adaptive aggregation

This work provides a self-supervised way to detect attacks when aggregating update vectors in federated learning. We collect empirical data from federated learning tasks for training a data-driven model that detects corrupted update vectors. Our data-driven model is attack-adaptive because it can identify the vulnerable regions of the update vector with respect to different attacks. We simulate the federated learning tasks under different attack scenarios on the test set in the server. We collect the update vectors and their labels (corrupted or genuine). The data-driven model can then be trained with the

update vectors as input and a loss function that encourages the prediction to agree with the label. With such a data-driven model, we can defend against the attacks missed out by the previous methods and refine our defense readily against new attacks. We may update our defense model incrementally and serve the new defense model as a 'security patch'. From the clients' perspective, they could be informed of the anomaly in their local data or model, as well as which type of anomaly they are suspected of. The clients can then inspect the unintended contamination in their data accordingly.

To obtain such a data-driven model, we may parameterize the indicator function $\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}$ in Equation 1 by a neural network and retrain the neural network upon new attacks. However, the neural network would not work unless it can take arbitrary number of update vectors and arbitrary permutation of the clients since the order of the arrival of the update vectors is not fixed. Moreover, it should have the capacity to incorporate existing robust estimators, such as the Coordinate-wise median [Yin *et al.*, 2018], as prior knowledge. Hence we may parameterize instead the $p(\boldsymbol{x}_i \in \mathbb{D}_{benign} | \boldsymbol{q}_t)$, which is the probability of $\boldsymbol{x}_i$ being a genuine update vector from a benign client given a robust estimate $\boldsymbol{q}_t$. Then we may get the next estimate by reweighing the update vector $\boldsymbol{x}_i$ with the probability. Here we define $\boldsymbol{q}$ to be our estimator of the robust mean $\boldsymbol{\mu}_{robust}$. We can obtain the estimator by the iteration:

$$\begin{aligned} \boldsymbol{q}_0 &= \mathbf{med}(\{\boldsymbol{x}_i\}), \\ \boldsymbol{q}_{t+1} &= \sum_{i=1}^{n} \frac{p(\boldsymbol{x}_i \in \mathbb{D}_{benign} | \boldsymbol{q}_t)}{\sum_{j=1}^{n} p(\boldsymbol{x}_j \in \mathbb{D}_{benign} | \boldsymbol{q}_t)} \boldsymbol{x}_i, \end{aligned} \tag{3}$$

where $\mathbf{med}(\{\boldsymbol{x}_i\})$ is a function taking median coordinate-wisely on the update vectors $\{\boldsymbol{x}_i\}$. The $\mathbf{med}(\{\boldsymbol{x}_i\})$ serves as an initial guess and could be replaced with other robust estimators or simply the mean. In the iteration, the update vector $\boldsymbol{x}_i$ is reweighed with the probability $p(\boldsymbol{x}_i \in \mathbb{D}_{benign} | \boldsymbol{q}_t)$ which depends on $\boldsymbol{x}_i$ and $\boldsymbol{q}_t$. Such form of reweighing is very similar to the attention mechanism in neural network.

Attention Our model is described in Algorithm 1. Our model consists of multiple passes of an attention module. The update vectors' weights are updated in each pass, and a new estimate is obtained by reweighing the update vectors. The overview of our method is summarized in Figure 1.

In our approach, we use the attention mechanism for the parameterization of the likelihood $p(\boldsymbol{x}_i \in \mathbb{D}_{benign} | \boldsymbol{q}_t)$. We encode the robust estimate $\boldsymbol{q}_t$ by the query encoder Q, and the update vectors $\{\boldsymbol{x}_i\}$ by the key encoder K and the value encoder V. We fix the value encoder V to be the identity and train the key encoder K and query encoder Q such that the alignment score

$$s_i = \frac{Q(\boldsymbol{q}_t) \cdot K(\boldsymbol{x}_i)}{\|Q(\boldsymbol{q}_t)\| \|K(\boldsymbol{x}_i)\|} \tag{4}$$

is closed to $+1$ for genuine update vector $\boldsymbol{x}_i \in \mathbb{D}_{benign}$ and -1 for corrupted update vector $\boldsymbol{x}_i \in \mathbb{D}_{attack}$.

The original version of attention in [Vaswani *et al.*, 2017] does not fit our purpose of parameterizing the iteration in Equation 3 since e^{s_i} cannot cover the range from 0 to 1. Instead, we use the softmax with temperature [Guo *et al.*, 2017]

Algorithm 1 Attack-adaptive aggregation with attention

Input: update vectors $\{\boldsymbol{x}_i\}$, hyperparameters c, ε, T
Output: robust estimate $\boldsymbol{q}_T$, the weights of the update vectors $\{w_i\}$
$\boldsymbol{q}_0 = \mathbf{med}(\{\boldsymbol{x}_i\})$
for $t = 0$ **to** $T - 1$ **do**
  for $i = 1$ **to** n, **in parallel do**
    $s_i = \frac{Q(\boldsymbol{q}_t) \cdot K(\boldsymbol{x}_i)}{\|Q(\boldsymbol{q}_t)\| \|K(\boldsymbol{x}_i)\|}$
    $w_i = \exp(cs_i) / \sum_{j=1}^{n} \exp(cs_j)$
    $w_i = w_i \cdot \mathbf{1}_{(w_i \geq \varepsilon/n)}$
  end for
  $\boldsymbol{q}_{t+1} = \sum_{i=1}^{n} w_i \boldsymbol{x}_i$
end for
return $\boldsymbol{q}_T, \{w_i\}$

and the overall expression in one pass of the attention module is

$$\boldsymbol{q}_{t+1} = \frac{\sum_{i=1}^{n} e^{cs_i} \boldsymbol{x}_i}{\sum_{i=1}^{n} e^{cs_i}} = \frac{\sum_{i=1}^{n} (e^{cs_i}/e^{c})\, \boldsymbol{x}_i}{\sum_{i=1}^{n} (e^{cs_i}/e^{c})}, \tag{5}$$

where the scale factor $c = 1/\tau$ is the inverse of the temperature τ.

Here we can observe that the form in Equation 5 is very similar to that in the Equation 3. The only difference is that the probability term $p(\boldsymbol{x}_i \in \mathbb{D}_{benign} | \boldsymbol{q}_t)$ in Equation 3 is replaced with e^{cs_i}/e^{c}. The term e^{cs_i}/e^{c} has a range very close to $[0, 1]$ for a large c. Also, it contains the information of the update vectors $\boldsymbol{x}_i$ and the last estimate $\boldsymbol{q}_t$. Therefore, we can see that e^{cs_i}/e^{c} is a suitable representation of the probability term $p(\boldsymbol{x}_i \in \mathbb{D}_{benign} | \boldsymbol{q}_t)$. In another perspective, the corrupted update vectors are assigned a lower weight when we have a larger c.

In our algorithm, we further add a truncation step with the threshold ε/n after we compute the softmax values. It is done to eliminate the effect of any corrupted update vectors with a potentially large magnitude. The truncation step

$$w_i \leftarrow w_i \cdot \mathbf{1}_{(w_i \geq \varepsilon/n)} \tag{6}$$

zeroes out the attention weight if it is smaller than the threshold ε/n. It is necessary because the exponent e^{-c} can never reach 0, and it can be problematic if we have a corrupted update vector with an extremely large magnitude, for example, e^{2c}.

4 Implementation

4.1 Dimensionality reduction

One difficulty of training our model is that the dimension of the update vector is very large compared to the number of clients n. The model may overfit to the irrelevant regions. In fact, since we are concerning the relative deviation of the update vectors, we can operate on the low-rank approximation of the set of update vectors. By performing PCA and assuming the update vectors are already centered (since the update vectors represent changes), we get a low dimension representation of the update vectors. Moreover, the vulnerable regions

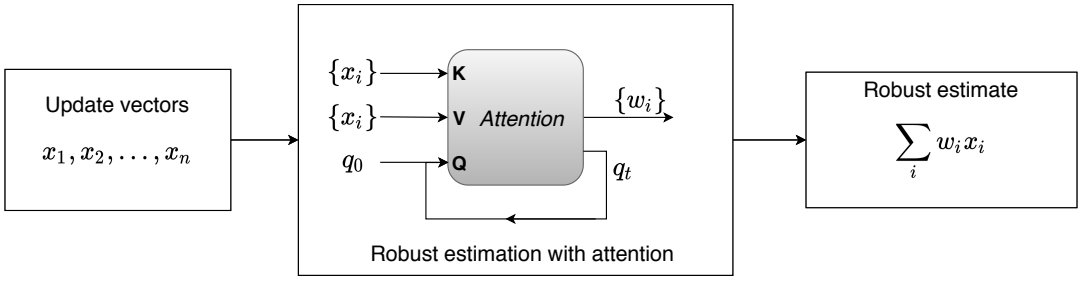


Figure 1: The overview of our approach for estimating the robust mean $\boldsymbol{\mu}_{robust}$ of the update vectors.

of the model in a federated learning task may reside in multiple layers. Hence, we perform PCA for each layer instead of performing it once for the whole update vectors. We keep all of the n principal components in each layer and the layer-wise PCA corresponds to a rotation in each layer.

After dimensionality reduction, we apply our model in Algorithm 1 to estimate the robust mean of the projected update vectors and their corresponding weights. The robust mean of the original update vectors can then be estimated by reweighing with the same weights.

A limitation of using PCA directly is that the attacker may hide its attack in multiple directions. For instance, the attacker in [Bhagoji *et al.*, 2019] adds a l_2-regularization on the distance to the previous benign updates. However, such l_2-regularized attack may not be stealthy in our case, where PCA is performed on each layer, and deviation at any layer may be flagged by our defense. Suppose the tolerable l_2-deviation is at most ε at each of the L layers and, as a result, the total deviation is at most $\sqrt{L}\varepsilon$. In this case, hiding the attack in a l_2-ball of size $\sqrt{L}\varepsilon$ is sufficient to bypass the plain PCA, but a size of ε is required to bypass our layer-wise version. Hence, the attacker needs to strengthen the l_2-regularization by a factor $\sqrt{L}$. Moreover, our attention module further suggests the vulnerable regions of the projected update vector. To hide the attack, the attacker needs to regularize further the cosine distance $\cos(\boldsymbol{v}(\boldsymbol{x}_{benign}), \boldsymbol{v}(\boldsymbol{x}_{attack}))$, where $\boldsymbol{v}$ is a projection to the vulnerable regions and $\boldsymbol{v}$ may not be known to the attacker. Our approach restricts the forgery at each layer and the vulnerable regions of the update vector. Hence, the attacker gets a worse trade-off between the stealthiness and effectiveness of its attack.

4.2 Training

We only need to consider the query encoder Q and the key encoder K to train our model. In our work, both encoders are 2-layer multi-layer perceptrons with ReLU activations. We perform the forward pass as described in Algorithm 1. We obtain the predicted estimate and compare it with the ground truth robust mean $\boldsymbol{\mu}_{robust}$ described in Equation 1. We use the L_1 loss and the Adam optimizer for the backpropagation. We train our model for 500 epochs with $T = 5$. For each attack, we run the federated learning tasks three times on the test set in the server to collect update vectors. Update vectors from two of the runs are used for training our model. The remaining run is served for validation.

4.3 Hyperparameter search

In our model, there are two major hyperparameters: c and ε. We perform a hyperparameter sweep to find a combination that yields a high validation accuracy on predicting attackers. We found that a set of moderate values around $c = 10, \varepsilon = 0.5$ is a good choice, and we use these values for our implementation. For the other hyperparameters, we found that they are also not sensitive. For instance, $T = 1$ is sufficient to reject the attackers, further passes to the attention module make the weights of the benign clients more uniform.

5 Experiments

5.1 Experimental setup

We compare our aggregation strategy with 6 prior works: FedAvg [McMahan *et al.*, 2017], Coordinate-wise median [Yin *et al.*, 2018], RFA [Pillutla *et al.*, 2019], Krum [Blanchard *et al.*, 2017], FoolsGold [Fung *et al.*, 2020], and Residual-based reweighing [Fu *et al.*, 2019]. We evaluate the performance of the aggregation strategies on four federated learning tasks under different attacks. To simulate a heterogeneous data distribution, we divide each dataset into disjoint partitions with the Dirichlet distribution with hyperparameter 0.9 as in [Bagdasaryan *et al.*, 2020]. Different from [Hsu *et al.*, 2019], we do not require the clients to have the same number of samples when generating the partitions. In each round, the clients train their local models on their data for one epoch. Then all clients, including possible attackers, are selected for the aggregation. Some prior works require hyperparameters. For Krum, we set $m = \lfloor \frac{n}{2} \rfloor - 2$. For FoolsGold, we set $\kappa = 1$. For Residual-based reweighing, we set $\lambda = 2, \delta = 0.1$.

5.2 Tasks

MNIST classification In this task, we use a LeNet [LeCun *et al.*, 1998] model with 10 clients. We evaluate the federated learning tasks under three types of attacks. (No attack) It simulates federated learning on heterogeneous data. (Omniscient) The attackers negate their update vectors by multiplying them by -1. (Backdoor) The attackers embed a pixel pattern to 50% of their image samples and alter their label to digit '2'. We run the task for 30 communication rounds.

CIFAR-10 classification In this task, we use a ResNet-18 [He *et al.*, 2016] model with 10 clients. We evaluate the federated learning tasks under three types of attacks: no attack, omniscient, backdoor. We run the task for 30 communication rounds.

Tiny-ImageNet classification We use the same model and the same attack scenarios as in CIFAR-10 classification. We run the task for 45 communication rounds.

IMDb sentiment analysis In this task, we use a Gated recurrent unit [Cho *et al.*, 2014] with FastText embedding [Joulin *et al.*, 2017] with 10 clients. We evaluate the federated learning tasks under three types of attacks. (No attack) It simulates the federated learning on heterogeneous data distribution. (Label flipping) The attackers swap the label of class 'Positive' and class 'Negative' in their local data. (Omniscient) The attackers negate their update vectors. We run the task for 10 communication rounds.

5.3 Metrics

We evaluate the performance of the aggregation strategies on the standard accuracy and the attack success rate. The accuracy (**ACC**) refers to the global model's standard accuracy on the test set. The attack success rate (**ASR**) is an evaluation of federated learning training against backdoor attacks. It measures how many backdoor-injected samples are classified as the target label of the attacker. If a backdoor-attacked sample is predicted to be the target label, we consider the attack on this sample is successful.

Our metrics are defined by

$$\begin{aligned} \textbf{ACC} &= \frac{\text{\# correct predictions}}{\text{\# samples}}, \\ \textbf{ASR} &= \frac{\text{\# successfully attacked samples}}{\text{\# attacked samples}}. \end{aligned} \tag{7}$$

The higher the accuracy, the better the aggregation strategy defends the attacks from interfering with the federated learning task. The lower the attack success rate, the better the aggregation strategy defends the backdoor attacks. An ideal aggregation strategy can achieve 100% accuracy and has the attack success rate as low as the fraction of attacked samples from the target class. We report the accuracy and the attack success rate in the final communication round. We take the average over three runs.

5.4 Evaluation

Visual tasks The results of the MNIST, CIFAR, Tiny-ImageNet tasks are summarized in Figure 2. The detailed results for backdoor attack can be found in the Appendix B.3. Under omniscient attack, most of the defense strategies failed when there were more attackers. It could be attributed to the curse of dimensionality, where negation of the vector does not alter the pairwise similarity much. In contrast, our approach was more resilient in all three visual tasks across different numbers of attackers since our approach operated on the projected update vectors where the negation became obvious. However, when there were 4 attackers in the most complicated Tiny-ImageNet task, the convergence was not good, even our approach outperformed others. The reason could be that the number of benign clients was too low to learn an effective global model in this complicated task. Nevertheless, our approach performed better than other approaches and had a good convergence when there was a moderate number of attackers.

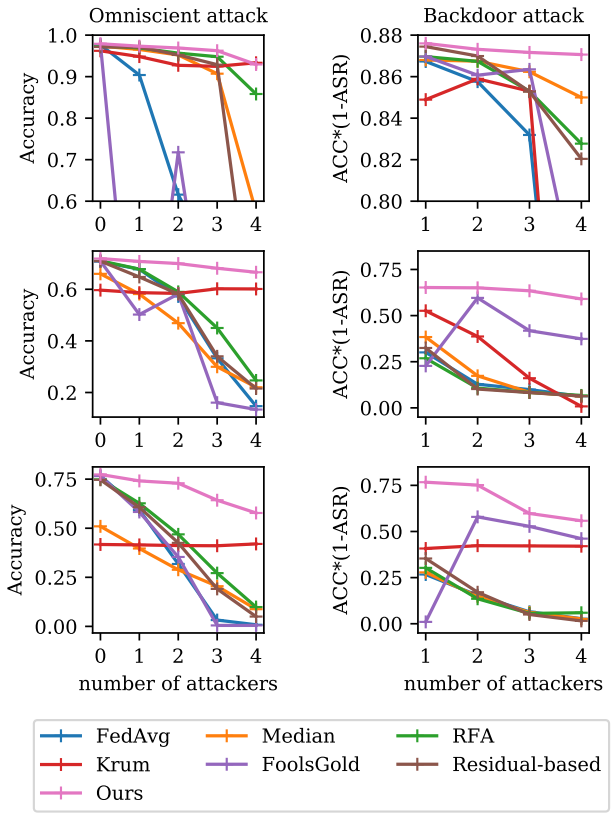


Figure 2: The performance of aggregation strategies in the MNIST (top), CIFAR(middle), ImageNet(bottom) classification task. The 'no attack' scenario is combined with the omniscient attack in the left plots.

Regarding the backdoor attack, our approach had the highest **ACC***(1-**ASR**) score in all visual tasks. It indicated that our approach had both a high global model accuracy and a low attack success rate. In the simplest MNIST task with abundant data, the backdoor was 'forgotten' when the global model became more mature over the communication rounds. Nevertheless, our approach removed the effect of the backdoor more effectively. In the CIFAR-10 and the Tiny-ImageNet task, the backdoor was not 'forgotten' since the ResNet-18 had more capacity to learn both the main task and the backdoor. Most approaches failed when there were 4 attackers and have $> 90\%$ **ASR**. In these tasks, the benign update vectors had a larger variance, and the corrupted update vectors from the backdoor attackers were relatively similar. It explained the higher resilience of FoolsGold when there were more attackers. Nevertheless, FoolsGold still had a $> 40\%$ **ASR** while our approach had a $< 15\%$ **ASR** when there were 4 attackers. It indicated that our approach could better identify the vulnerable regions of the update vectors. On the other hand, Krum learned the backdoor in CIFAR-10 task and did not learn both the backdoor and the main task well in the Tiny-ImageNet task since it aggregated only a small fraction of clients. In contrast, our approach aggregated most of the benign clients and achieved a high global accuracy and a low attack sucess rate.

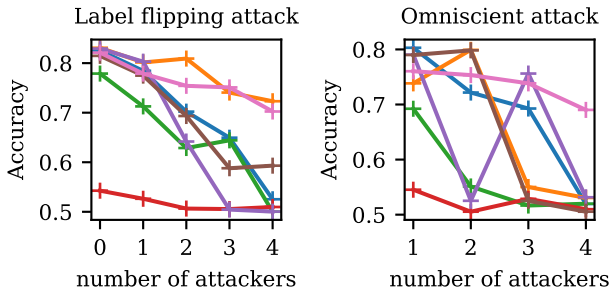


Figure 3: The performance of aggregation strategies in the IMDb sentiment analysis task. We use the same legend in Figure 2. Our approach is in pink.

	Attack Sucess Rate				
# of attackers	1	2	3	4	Average
FedAvg	57.37	81.87	86.03	91.02	79.07
MLP	54.35	79.92	86.25	89.60	77.53
w/o c	15.21	57.42	84.12	88.90	61.41
w/o ε	9.78	10.39	11.19	32.30	15.92
Ours	**6.78**	**5.64**	**6.29**	**13.03**	**7.94**

Table 1: Ablation study on the effect of removing attention (MLP), scaled softmax (w/o c), or the truncation step (w/o ε).

Textual task The Figure 3 shows the results on the IMDb sentiment analsysis task. most approaches degraded quickly with the number of attackers. Some of them were even worse than FedAvg. Seemingly, acquiring a benign client was more important than discarding an attacker in this task. Nevertheless, our approach managed to maintain a relatively high accuracy against the attacks. When there were 4 omniscient attackers, our approach was the only one that worked.

5.5 Ablation study

We performed an ablation study on various components of our model. We evaluated the effectiveness of the defenses on the CIFAR-10 task under the backdoor attack. The results are summarized in Table 1. The scale factor c in our softmax played an important role in defending attacks. The threshold factor ε controlled the trade-offs between robustness and convergence. Without attention, a plain multi-layer perceptron overfitted a certain permutation of the clients and could not distinguish the attackers when they arrived in a different order. Our attention-based model avoided this problem since it is permutation invariant.

5.6 Transferability of defense

As a practical data-driven aggregation mechanism, it is important to know how far our defense can be generalized to unseen scenarios. We study the transferability of our defense. Specifically, we trained our model on the CIFAR-10 task under backdoor attacks with a fixed backdoor pattern. Then we evaluated the defenses' performance under different scenarios: 1.) classification tasks on CIFAR-100 instead of CIFAR-10, 2.) 100 clients instead of 10 clients, and 3.) different backdoor patterns. The results is shown in Figure 4. Our approach generalized the defense better when there was a lower fraction of attackers. On the other hand, Figure 5 showed that our approach could generalize the defense to unseen backdoor patterns. In summary, our defense can be generalized to unseen attack scenarios when there are a lower fraction of attackers. It meets our expectations since our approach learns the similarity measure based on the attacks' traits of a set of plausible attack scenarios. With a new attack scenario, it may share part of the vulnerable regions and the deviation in these regions remains detectable when there are a lower fraction of attackers. On the other hand, our model is less confident to accuse a client when there are a high fraction of attackers. To address the issue, we need to provide additional supervision on the backdoor patterns that we want to defend.

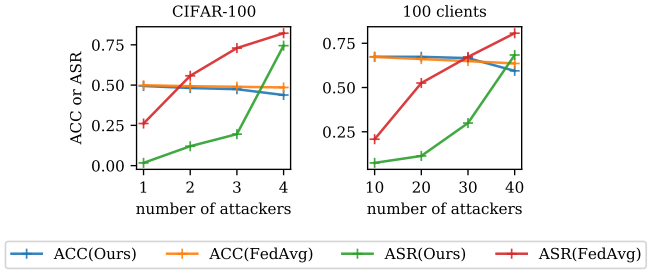


Figure 4: Performance of defense against backdoor attack across different scenarios. (Left) Transfering the defense to CIFAR-100 task. (Right) Transfering the defense on 100 clients in CIFAR-10 task.

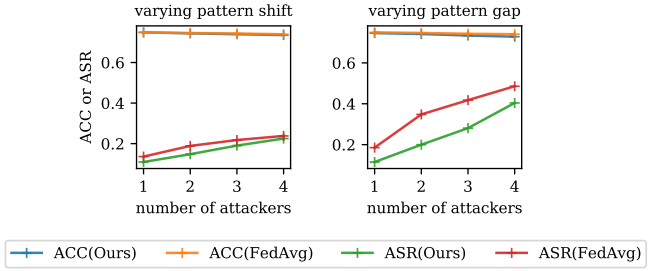


Figure 5: Performance of defense against backdoor attack with different backdoor patterns. We varied the values of the parameters of the backdoor patterns and reported the average result w.r.t. each parameter.

6 Conclusion

In this work, we presented a novel approach for robust federated learning using a deep neural network as the aggregation function. To the best of our knowledge, our aggregation strategy is the first one that is attack-adaptive and learns to defend against various attacks in a data-driven fashion. The attention mechanism in our designed network is effective in propagating contextual information to detect malicious attackers. We further demonstrate the transferability of our defense. We hope our attack-adaptive aggregation paradigm can inspire more work in this direction. Our source code is publicly available on https://github.com/cpwan/Attack-Adaptive-Aggregation.

References

[Bagdasaryan *et al.*, 2020] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *AISTATS*, 2020.

[Baruch *et al.*, 2019] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *NeurIPS*, 2019.

[Bhagoji *et al.*, 2019] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[Blanchard *et al.*, 2017] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NeurIPS*, 2017.

[Chen *et al.*, 2020] Xiangyi Chen, Tiancong Chen, Haoran Sun, Zhiwei Steven Wu, and Mingyi Hong. Distributed training with heterogeneous data: Bridging median-and mean-based algorithms. *NeurIPS*, 2020.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.

[Fu *et al.*, 2019] Shuhao Fu, Chulin Xie, Bo Li, and Qifeng Chen. Attack-resistant federated learning with residual-based reweighting. *RSEML-AAAI*, 2019.

[Fung *et al.*, 2020] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *RAID*, 2020.

[Guo *et al.*, 2017] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[Hsu *et al.*, 2019] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv:1909.06335*, 2019.

[Joulin *et al.*, 2017] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *EACL, Short Papers*, 2017.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

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

[Li *et al.*, 2020] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*, 2020.

[Maas *et al.*, 2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL-HLT*, 2011.

[McMahan *et al.*, 2017] 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.

[Mhamdi *et al.*, 2018] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *ICML*, 2018.

[Muñoz-González *et al.*, 2019] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv:1909.05125*, 2019.

[Okuno *et al.*, 2018] Akifumi Okuno, Tetsuya Hada, and Hidetoshi Shimodaira. A probabilistic framework for multi-view feature learning with many-to-many associations via neural networks. In *ICML*, 2018.

[Pillutla *et al.*, 2019] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *FL-ICML*, 2019.

[Sattler *et al.*, 2020] Felix Sattler, Klaus-Robert Müller, Thomas Wiegand, and Wojciech Samek. On the byzantine robustness of clustered federated learning. In *ICASSP*, 2020.

[Smith *et al.*, 2017] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *NeurIPS*, 2017.

[Sun *et al.*, 2019] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? In *FL-NeurRIPS*, 2019.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Wang *et al.*, 2020] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *NeurIPS*, 2020.

[Xie *et al.*, 2019a] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *ICLR*, 2019.

[Xie *et al.*, 2019b] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Zeno: Byzantine-suspicious stochastic gradient descent. In *ICML*, 2019.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM TIST*, 10(2):1–19, 2019.

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

A Illustrations of data and model

A.1 Generating heterogeneous data distribution

In our experiments, we generated heterogeneous data distribution with the Dirichlet distribution. An instance of the generated data distribution is shown in Figure 6. We used the Dirichlet distribution with the concentration parameter $\boldsymbol{\alpha} = (0.9, 0.9, \ldots, 0.9)$ to generate the fraction of samples to be drawn for each client. We performed the partition for each label and assigned the drawn samples to the clients. It was different from some previous works, which forced the clients to have the same number of data and drawn the data samples with replacement. In our data distribution generation, the data were divided into **disjoint** partitions with varying sizes. In this way, the federated learning experiments could be run on different data distributions. It allowed us to collect update vectors from diverse federated learning settings, which provided higher-quality data for training our model.

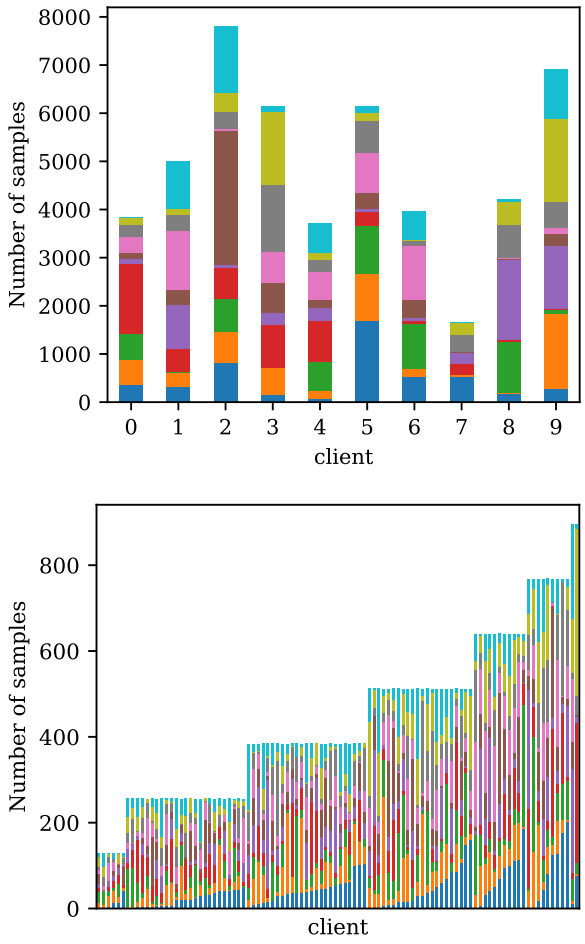


Figure 6: An illustration of the data distribution across clients. Each color corresponds to a class. (Left) 10 clients. (Right) 100 clients. The clients are sorted solely for visualization purposes. The clients appear in random order when we run the experiments. We used a batch size of 128 in the local training. Hence the total number of samples in each client is a multiple of 128.

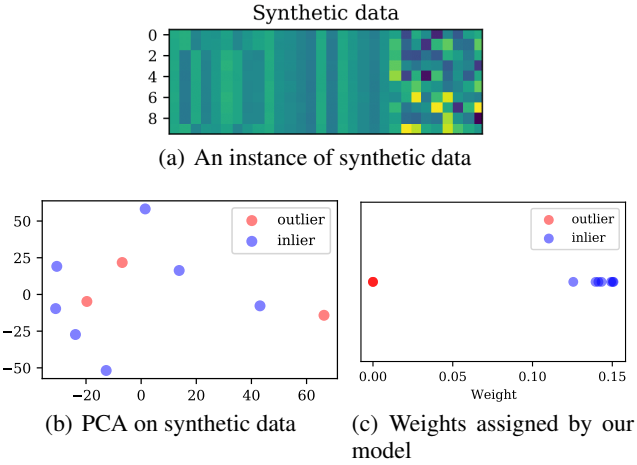
A.2 Demonstrations on synthetic data

We will demonstrate how our model works for robust estimation on synthetic data. Each instance of the synthetic data is a set of 10 samples drawn from a 30-dimensional multivariate normal distribution. The last one-third of features consist of noise, while the outliers had their first one-third of features modified. We generated a training set of size 2048. Figure 7(a) illustrated an instance of the synthetic data in the validation set, in which the sample 0, 1, 9 are the outliers. Figure 7(b) showed no apparent distinctions between inliers and outliers under PCA. We trained our model on the 2048 instances of synthetic data. The results in Figure 7(c) illustrated that our model successfully distinguished the outliers.

Apart from estimating each individual set of samples, we also estimate the feature importance by measuring the magnitude of the weight W in the first layer of our key encoder for the i-th feature:

$$\text{imptz}(i) = \sum_{j=0}^{h} |W_{i,j}|\,, \tag{8}$$

where h is the dimension of the hidden layer. Figure 8 shows that our model indeed captured the features that distinguish outliers from inliers. We may employ similar visualization to explore which part of the model is more vulnerable to adversarial attacks in federated learning.



(a) An instance of synthetic data

(b) PCA on synthetic data

(c) Weights assigned by our model

Figure 7: Illustration of outlier detection on synthetic data. The samples 0, 1, 9 are the outliers.

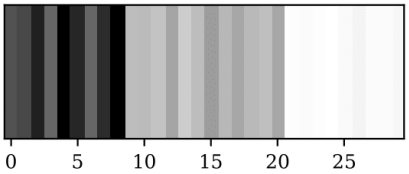


Figure 8: The feature importance on the 30 features of the synthetic data, estimated from our model (the darker, the more important).

B More experiments

B.1 Capturing the traits of attacks

We may analyze the feature importance on the attack-adaptive aggregation model we trained similarly as in section A.2 . We will use the same definition from Equation 8. In this section, we will analyze our attack-adaptive aggregation model. The model were trained to defend against backdoor attack in the CIFAR-10 tasks with 10 clients. We summed up the feature importance for each layer (with weight and bias separately counted). The most prominent layers are

- `layer3.0.downsample.1.weight`,
- `layer3.0.downsample.0.weight`,
- `layer4.1.bn2.bias`,
- `layer4.0.downsample.1.weight`

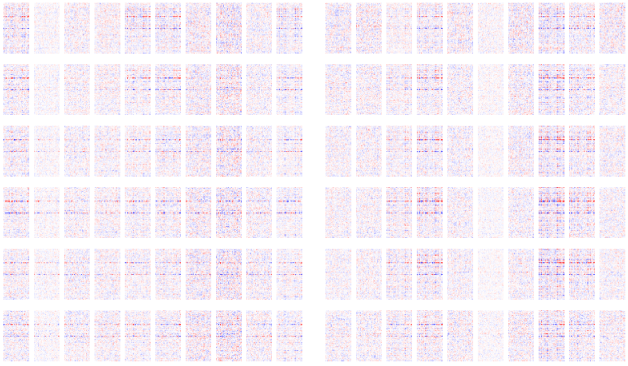
The result is reasonable. Since the backdoor attacker attempts to recognize the backdoor pattern, the attacker's local model needs to extract features in its residual blocks. Although the receptive field is sufficient to cover the pattern (with a shape of 3 by 7) in the earlier blocks, the later blocks seem responsible for learning the pattern. Besides, it seems that the downsampling layers are more sensitive to the backdoor pattern. It may be due to that the attacker favors some channels during downsampling. Indeed, when we look at the `layer3.0.downsample.0.weight` layer of the update vectors in Figure 9, the two horizontal strips of the attackers indicate that the corresponding two channels are given a higher weight during downsampling.

Based on the literature on neural network architecture, we may already have some ideas on the neural network's vulnerability. However, it is not immediately trivial for one to decide which layers may contain the attack's traits. On the other hand, our attack-adaptive aggregation model can readily discover the attack's traits and offer a defense at the same time. It is illustrated in Figure 9 that the traits of the attack we found indeed affected the training. For FedAvg, the update vectors of the benign clients were progressively contaminated by the two horizontal strips. It implied that the attackers had successfully led the gradient direction to a malicious objective. In contrast, with our attack-adaptive aggregation, the benign clients could deliver uncontaminated update vectors and continued the global training with an appropriate gradient direction. The result confirms the ability of our attack-adaptive aggregation to identify the traits of attack.

In summary, our approach serves as a defense strategy and provides a way to analyze the adversarial attack's traits. In contrast, some previous approaches did not consider the contributions of different regions of the update vector. In this regard, our approach is an interpretable defense strategy.

B.2 Transferability of defense across backdoor patterns

In the main text, we summarized the average results of transferring the defense to different backdoor patterns. In this section, we will give the results in detail. During the training of our attack-adaptive aggregation, we used the pixel pattern shown in Figure 10. The backdoor pattern occupies a 3 (pixels) by 7 (pixels) region in the top-left corner of the image.



(a) FedAvg (b) Ours

Figure 9: The `layer3.0.downsample.0.weight` layer in the (0,2,4,6,8,10)-th communication round in the CIFAR-10 task under backdoor attack with FedAvg (Left) and our attack-adaptive aggregation (Right). The columns correspond to the clients. The blue color means positive, the red color means negative. The darker is the color, the larger is the magnetuide. The model parameters are reshaped for illustration purposes. For the task with FedAvg, the client 0, 4, 5, 9 is the attacker. For the task with our attack-adaptive aggregation, the client 2, 3, 7, 8 is the attacker.

It consists of four 1 by 3 horizontal bars arranged in two columns, with a 1-pixel gap. The backdoor attacker modified the red channel of these 12 pixels to the largest intensity. In each experiment on transferability, we letted the attackers to use a backdoor pattern with different $\textit{shift}_y$, $\textit{shift}_x$, and *gap* parameters. The $\textit{shift}_y$ and $\textit{shift}_x$ parameters control the shift of the pattern along and vertical and horizontal direction respectively. The *gap* parameter controls how wide the gap is between the bars in addition to the default 1-pixel gap. In the evaluation, we varied only one parameter and kept the other parameter fixed. In additional, for experiments on $\textit{shift}_y$ and $\textit{shift}_x$, we set $\textit{gap} = 1$ to introduce two-pixels gaps between the bars in the backdoor pattern (instead of only one). It was done to ensure that the new backdoor pattern is different from the one we used in training.

The effects of $\textit{shift}_y, \textit{shift}_x$, *gap* parameters are shown in Figure 11. For the $\textit{shift}_y, \textit{shift}_x$ parameters, the backdoor attack was generally weaker when the backdoor pattern was further away from the top-left corner. It agrees to the previous research that the local model may 'forget' the backdoor pattern in the middle of the image. On the other hand, it is shown that our attack-adaptive aggregation was able to generalize the defense across different backdoor pattern parameters when there were 1 to 2 attackers. Nevertheless, our attack-adaptive aggregation did not generalize well enough to defend 3 to 4 attackers. Since the attack's traits in the update vectors could be different when a different backdoor pattern is used, our attack-adaptive aggregation may treat some of the new traits of the different attack as a naturally occurred variance. For instance, if the new backdoor pattern is on the top-right corner instead, then the backdoor attack may leave a different trait in the update vector. If there is only one attacker, such a trait may still be noticeable by our model.

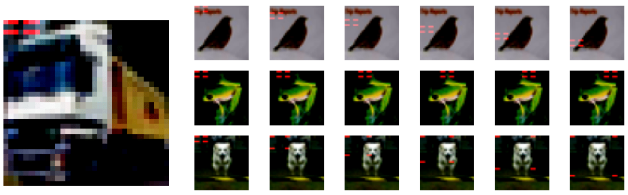
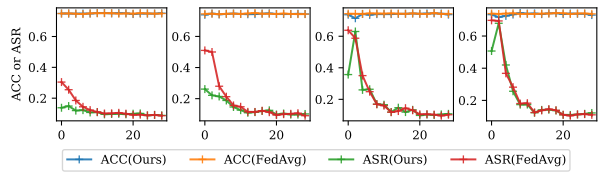
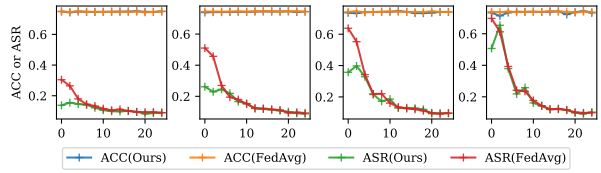


Figure 10: Backdoor patterns. (Left) The red pixel pattern is the backdoor pattern used in training our attack-adaptive aggregation. (Right) From top to bottom, each row demonstrates how the backdoor pattern looks like in different $shift_y$, $shift_x$, and *gap*. The $shift_y$ and $shift_x$ parameters control the shift of the pattern along and vertical and horizontal direction respectively. The *gap* parameter controls how many gap to be injected between the bars in additional to the default 1 pixel gap.

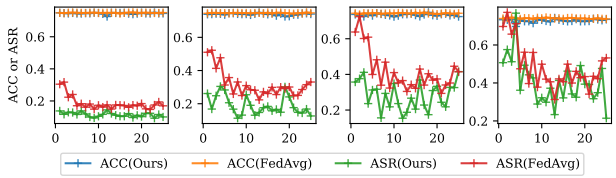
However, if there are multiple attackers, our model cannot decide whether such a trait results from an attack or images containing an object in the top-right corner. As a result, our attack-adaptive aggregation could not be confident enough to accuse the attackers. To address the issue, we need to train a new attack-adaptive aggregation model with update vectors collected under different attack parameters.



(a) Varying $shift_y$.



(b) Varying $shift_x$.



(c) Varying *gap*.

Figure 11: Effects of backdoor patterns on the transferability of our attack-adaptive aggregation. From left to right, the plots show the performance of the defense strategies under 1 to 4 attackers. **ACC** stands for the standard accuracy (higher the better). **ASR** stands for the attack success rate (lower the better).

B.3 Detailed results of the backdoor attacks

Table 2,3,4 show the detailed results of the visual tasks under backdoor attacks described in Section 5.4.

B.4 Training with multiple attacks

We evaluated the effect of training our defense on multiple types of attacks. We compared the performance of our defense against the attacks when the defense was trained on 1.) only the backdoor attack scenarios, 2.) only the omniscient attack scenarios, or 3.) both the backdoor attack and the omniscient attack scenarios. The results of our defenses against backdoor attack is summarized in Table 5. Interestingly, even we trained our defense only on the omniscient attack scenarios but not the backdoor attack scenarios, the defense can still defend the backdoor attack. This is reasonable because the omniscient attack negates the update vector and every coordinate in the update vector could be considered vulnerable under the omniscient attack. Therefore, our attention module takes every coordinate of the update vector into account, including the coordinates involved with the backdoor attack. So, the anomaly in these coordinates could still be detected. On the other hand, when we train the defense with the backdoor attack, our attention module learned the vulnerable regions better and had a better defense when there were more attackers. Similarly, we can defend the omniscient attack even we trained our defense only on the backdoor attack scenarios, as shown in Table 6. This is again due to the overlapping vulnerable regions with respect to the two attacks. When there were 4 attackers, our defense was again stronger if we trained our defense on the same attack.

We observed that we can defend a different type of attack if the attack shares the vulnerable region with the attack scenario that we trained on. However, when we train our defense on multiple attacks, our defense may not perform as good as training alone on a single attack, especially in the cases of higher fraction of attackers. It may due to that our attention module learned a suboptimal vulnerable regions when multiple attacks were involved in the training. When our defense tries to increase the detection rate of an attack, it may also raise the chance of false alarm in another attack scenario. Therefore, our defense may become more conservative and do not work as good when there are higher fraction of attackers. Nevertheless, the performance of our defense trained on multiple attacks was still competitive when compared with other aggregation strategies.

# of attackers	Accuracy 1	2	3	4	Average	Attack Sucess Rate 1	2	3	4	Average
FedAvg	97.09	96.76	96.43	96.02	96.58	10.67	11.37	13.73	38.84	18.65
Median	96.77	96.75	96.46	95.61	96.40	10.30	10.34	10.60	11.10	10.58
RFA	97.08	96.88	96.39	96.37	96.68	10.42	10.47	11.51	14.11	11.63
Krum	94.99	95.46	94.98	95.26	95.17	10.63	10.02	10.19	42.66	18.38
FoolsGold	97.23	96.47	96.85	96.25	96.70	10.55	10.78	10.84	22.34	13.63
Residual-based	97.39	97.09	96.34	95.55	96.59	10.21	10.40	11.48	14.14	11.56
Ours	97.40	96.90	96.83	96.79	96.98	**10.06**	**9.89**	**9.98**	**10.05**	**10.00**

Table 2: Performance on MNIST task under backdoor attack.

# of attackers	Accuracy 1	2	3	4	Average	Attack Sucess Rate 1	2	3	4	Average
FedAvg	70.51	70.61	70.85	69.13	70.28	57.37	81.87	86.03	91.02	79.07
Median	64.31	66.60	65.39	64.82	65.28	40.33	73.96	87.13	90.19	72.90
RFA	71.03	70.42	70.10	69.42	70.24	62.17	84.83	88.07	90.28	81.34
Krum	58.31	60.76	57.23	58.71	58.75	9.78	36.34	71.94	98.64	54.18
FoolsGold	70.52	69.54	68.81	67.69	69.14	67.75	14.30	39.24	44.77	41.52
Residual-based	70.72	70.04	69.61	69.40	69.94	54.10	85.45	88.09	90.78	79.60
Ours	69.97	68.93	67.70	67.86	68.62	**6.78**	**5.64**	**6.29**	**13.03**	**7.94**

Table 3: Performance on CIFAR-10 task under backdoor attack.

# of attackers	Accuracy 1	2	3	4	Average	Attack Sucess Rate 1	2	3	4	Average
FedAvg	62.79	53.43	41.56	32.75	47.63	57.50	71.21	84.45	91.60	76.19
Median	41.73	32.99	25.00	17.88	29.40	33.30	55.35	75.44	85.39	62.37
RFA	64.17	49.68	41.37	29.24	46.12	52.81	72.77	86.46	79.62	72.91
Krum	40.93	42.50	42.42	42.24	42.02	**0.38**	**0.46**	**0.57**	**0.48**	**0.47**
FoolsGold	16.02	70.03	67.82	60.91	53.70	93.73	17.30	22.18	24.28	39.37
Residual-based	63.91	49.96	40.50	31.64	46.50	44.67	65.83	87.64	95.10	73.31
Ours	77.06	75.54	60.18	59.24	68.01	0.45	0.53	0.67	5.80	1.86

Table 4: Performance on ImageNet task under backdoor attack. Note that Krum has a poor accuracy even it achieves the lowest attack success rate. Our approach achieves both a low attack success rate and a high accuracy.

# of attackers	Accuracy 1	2	3	4	Average	Attack Sucess Rate 1	2	3	4	Average
Backdoor (B)	69.97	68.93	67.70	67.86	68.62	**6.78**	**5.64**	**6.29**	**13.03**	**7.94**
Omniscient (O)	70.06	68.92	69.02	66.36	68.59	8.70	10.12	9.44	34.23	15.62
B+O	69.85	69.13	68.23	67.87	68.77	7.56	8.16	13.05	38.02	16.70

Table 5: Performance of our approach on CIFAR-10 task under **backdoor attack** when trained with different attack scenarios.

# of attackers	Accuracy 1	2	3	4	Average
Backdoor (B)	70.46	**68.47**	67.41	59.78	66.53
Omniscient (O)	70.79	67.97	**68.14**	**66.59**	**68.87**
B+O	**70.85**	68.46	66.69	44.73	62.68

Table 6: Performance of our approach on CIFAR-10 task under **omniscient attack** when trained with different attack scenarios.

C Theoretical Analysis

C.1 Universal approximation property

We use the attention module in our attack-adaptive aggregation. The query encoder Q and the key encoder K in the attention module are 2-layer multi-layer perceptrons with ReLU activation. We will show in Theorem 1 that given large enough hidden units and large enough latent space in the last layer of Q and K, the dot product $Q(\cdot) \cdot K(\cdot)$ can approximate any similarity measure and the alignment score function $\frac{Q(\cdot)\cdot K(\cdot)}{\|Q(\cdot)\|\|K(\cdot)\|}$ is a projection of such a similarity measure to $[-1, 1]$.

Theorem 1. *Let $f_1, f_2 : [-M, M]^{d_j} \to [-M, M]^{D'}$ be continuous functions, $h : [-M, M]^{D'} \to \mathbb{R}$ be a symmetric, continuous positive definite kernel function, $\sigma(\cdot)$ be ReLU. Then, for arbitrary $\varepsilon' > 0$, by specifying sufficiently large $D, T \in \mathbb{N}$, there exist $\boldsymbol{A} \in \mathbb{R}^{D\times T}, \boldsymbol{B} \in \mathbb{R}^{T\times d_j}, \boldsymbol{c} \in \mathbb{R}^T$ such that*

$$\left| h\left(f_1(\boldsymbol{x}), f_2\left(\boldsymbol{x}'\right)\right) - \left\langle f_1^{\psi}(\boldsymbol{x}), f_2^{\psi}\left(\boldsymbol{x}'\right)\right\rangle \right| < \varepsilon'$$

for all $(\boldsymbol{x}, \boldsymbol{x}') \in [-M, M]^{d_1+d_2}$ where $f_i^{\psi}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{\sigma}\left(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}\right)$ are two-layer neural networks with T hidden units, D dimension output layer and $\boldsymbol{\sigma}(\boldsymbol{x})$ is element-wise $\sigma(\cdot)$ function.

The Theorem 1 is a special case of the Theorem 5.1 in [Okuno *et al.*, 2018]. We apply their result for our theorem. The theorem implies that the dot product of two neural networks can approximate any similarity measure. Suppose we feed the robust mean to the query encoder Q and the sets of update vectors to the key encoder K. In that case, the theorem implies that the encoders have the approximation ability such that the alignment score $\frac{Q(\cdot)\cdot K(\cdot)}{\|Q(\cdot)\|\|K(\cdot)\|}$ is close to $+1$ for genuine update vectors and is close to -1 for corrupted update vectors.

C.2 Robust mean estimation as an optimization problem

In each step t of our algorithm,

$$\begin{aligned}
&\|g\left(\{\boldsymbol{x}_i\}\right) - \boldsymbol{\mu}_{robust}\| = \|\boldsymbol{q}_t - \boldsymbol{\mu}_{robust}\| \\
&= \left\| \sum_{i=1}^{n} tr\left(\frac{\left(e^{cs_i}/e^{c}\right)}{\sum_{k=1}^{n}\left(e^{cs_k}/e^{c}\right)}\right)\boldsymbol{x}_i - \sum_{i=1}^{n}\frac{\mathbf{1}_{(\boldsymbol{x}_i\in\mathbb{D}_{benign})}}{\sum_{k=1}^{n}\mathbf{1}_{(\boldsymbol{x}_k\in\mathbb{D}_{benign})}}\boldsymbol{x}_i \right\|
\end{aligned}$$

where $tr(*)$ is a truncation function that yields zero if $* < \varepsilon/n$

When training our model, we used the L_1 loss and $T = 5$, as stated in Section 4.2. **This is exactly minimizing $\|g\left(\{\boldsymbol{x}_i\}\right) - \boldsymbol{\mu}_{robust}\|$ w.r.t. the minimizer s_i at the last time step $T = 5$.** Hence, the quality of s_i decides the quality of our algorithm for the minimization. In our algorithm, s_i is the result of the dot product of the encoders. That is why we need Theorem 1 to show that the minimizer s_i can be sufficiently optimal.

C.3 Error bound of the robust mean estimation

Here, we attempt to give an error bound for estimating the robust mean with our attack-adaptive aggregation model. Suppose $\boldsymbol{\mu}_{robust} \in \mathbb{R}^k$ is the robust estimate. We want a similarity measure $h'(k, q)$ such that for $\boldsymbol{q}'$ in the neighborhood of $\boldsymbol{\mu}_{robust}$, $h'(\boldsymbol{x}_i, \boldsymbol{q}') = 1$ for the genuine update vector $\boldsymbol{x}_i$ and $h'(\boldsymbol{x}_j, \boldsymbol{q}') = -1$ for the corrupted update vector $\boldsymbol{x}_j$. In the rest of this section, $\| * \|$ stands for the l_1 norm.

We denote $s(k, q) = \frac{K(k)\cdot Q(q)}{\|K(k)\|\|Q(q\|)}$ for the attention score between k and q. Theorem 1 suggests that we can train a neural network such that $s(k, q)$ approximates $h'(k, q)$. That is, for $\boldsymbol{q}_{t-1}$ in δ-neighborhood of $\boldsymbol{\mu}_{robust}$ relative to the update vectors where $\frac{\|\boldsymbol{q}_{t-1}-\boldsymbol{\mu}_{robust}\|}{\max_l\left(\|\boldsymbol{x}_l\|\right)} \leq \delta$,

$$\begin{aligned}
\|1 - s(\boldsymbol{x}_i, \boldsymbol{q}_{t-1})\| < \varepsilon' \quad \forall \boldsymbol{x}_i \in \mathbb{D}_{benign} \\
\| - 1 - s(\boldsymbol{x}_j, \boldsymbol{q}_{t-1})\| < \varepsilon' \quad \forall \boldsymbol{x}_j \in \mathbb{D}_{attack}.
\end{aligned} \tag{9}$$

for some small $\varepsilon' > 0$. We will use this definition in Lemma 2.

Lemma 2. *Suppose $s(k, q)$ approximates $h'(k, q)$ for $\boldsymbol{q}_{t-1}$ in the δ-neighborhood of $\boldsymbol{\mu}_{robust}$ relative to the update vectors with an error bounded by ε'. Then the next estimate $\boldsymbol{q}_t$ in our algorithm would have an error bounded by*

$$\max\left(e^{c\varepsilon'} - 1, \frac{n}{n-m}e^{c\varepsilon'}e^{-2c}\right)\max_{l, w_l\geq\varepsilon/n}\left(\|\boldsymbol{x}_l\|\right), \tag{10}$$

where c is the scale factor, ε is the threshold factor defined in our algorithm, and w_l is the attention score of update vector x_l.

Moreover, the error bound improves by a rate γ if

$$\begin{aligned}
\varepsilon' &\leq \min\left(\frac{1}{c}\ln(\gamma\delta + 1), 2 - \frac{1}{c}\ln\left(\frac{\gamma^{-1}\delta^{-1} - m/n}{1 - m/n}\right)\right) \\
&= O(\gamma\delta).
\end{aligned} \tag{11}$$

Proof. Since the attention score is normalized, we have

$$\begin{aligned}
s(\boldsymbol{x}_i, \boldsymbol{q}_{t-1}) > 1 - \varepsilon' \quad \forall \boldsymbol{x}_i \in \mathbb{D}_{benign} \\
s(\boldsymbol{x}_j, \boldsymbol{q}_{t-1}) < -1 + \varepsilon' \quad \forall \boldsymbol{x}_j \in \mathbb{D}_{attack}.
\end{aligned}$$

Suppose we have m attackers and $n - m$ benign clients, then

$$\begin{aligned}
(n-m)e^{c(1-\varepsilon')} + me^{-c} &< \sum_k e^{cs(x_k, \boldsymbol{q}_{t-1})} \\
&< (n-m)e^{c} + me^{-c(1-\varepsilon')}.
\end{aligned}$$

Hence, $\forall \boldsymbol{x}_i \in \mathbb{D}_{benign}$,

$$\begin{aligned}
\frac{e^{c}}{(n-m)e^{c(1-\varepsilon')} + me^{-c}} &> \frac{e^{cs(\boldsymbol{x}_i, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} \\
&> \frac{e^{c(1-\varepsilon')}}{(n-m)e^{c} + me^{-c(1-\varepsilon')}}, \\
\frac{1}{(n-m)e^{-c\varepsilon'} + me^{-2c}} &> \frac{e^{cs(\boldsymbol{x}_i, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} \\
&> \frac{e^{-c\varepsilon'}}{(n-m)e^{-c\varepsilon'} + me^{-2c}}
\end{aligned}$$

and thus,

$$\begin{aligned}
&\left\| w_i - \frac{\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \right\| \\
&= \left\| \frac{e^{cs(\boldsymbol{x}_i, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} - \frac{\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \right\| \\
&= \left\| \frac{e^{cs(\boldsymbol{x}_i, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} - \frac{1}{n-m} \right\| \\
&\leq \frac{\max\left(1 - e^{-c\varepsilon'} - \frac{m}{n-m} e^{-2c}, e^{-2c}\right)}{(n-m)e^{-c\varepsilon'} + me^{-2c}} \\
&= \max\left(\frac{e^{2c} - e^{c(2-\varepsilon')} - \frac{m}{n-m}}{(n-m)e^{c(2-\varepsilon')} + m}, \frac{1}{(n-m)e^{c(2-\varepsilon')} + m}\right)
\end{aligned} \tag{12}$$

On the other hand, $\forall \boldsymbol{x}_j \in \mathbb{D}_{attack}$,

$$\begin{aligned}
\frac{e^{c(-1+\varepsilon')}}{(n-m)e^{c(1-\varepsilon')} + me^{-c}} &> \frac{e^{cs(\boldsymbol{x}_j, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} \\
&> \frac{e^{c(-1)}}{(n-m)e^{c} + me^{-c(1-\varepsilon')}}, \\
\frac{e^{c\varepsilon'}}{(n-m)e^{2c} + me^{c\varepsilon'}} &> \frac{e^{cs(\boldsymbol{x}_j, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} \\
&> \frac{1}{(n-m)e^{2c} + me^{c\varepsilon'}}
\end{aligned}$$

and thus,

$$\begin{aligned}
&\left\| w_j - \frac{\mathbf{1}_{(\boldsymbol{x}_j \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \right\| \\
&= \left\| \frac{e^{cs(\boldsymbol{x}_j, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_k, \boldsymbol{q}_{t-1})}} - \frac{\mathbf{1}_{(\boldsymbol{x}_j \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \right\| \\
&= \left\| \frac{e^{cs(\boldsymbol{x}_i j, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_i k, \boldsymbol{q}_{t-1})}} - \frac{0}{n-m} \right\| \\
&\leq \frac{e^{c\varepsilon'}}{(n-m)e^{2c} + me^{c\varepsilon'}} \\
&= \frac{1}{(n-m)e^{c(2-\varepsilon')} + m}
\end{aligned} \tag{13}$$

Combining the results for the benign clients and the attackers, we have the error bound for estimating the robust mean:

$$\begin{aligned}
&\|\boldsymbol{q}_t - \boldsymbol{\mu}_{robust}\| \\
&= \left\| \sum_l w_l \cdot \mathbf{1}_{(w_i \geq \varepsilon/n)} \boldsymbol{x}_l - \boldsymbol{\mu}_{robust} \right\| \\
&= \left\| \sum_l tr\left(\frac{e^{cs(\boldsymbol{x}_i l, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_i k, \boldsymbol{q}_{t-1})}} \right) \boldsymbol{x}_l - \sum_l \frac{\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \boldsymbol{x}_l \right\| \\
&\leq \sum_i \left\| \frac{e^{cs(\boldsymbol{x}_i i, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_i k, \boldsymbol{q}_{t-1})}} - \frac{\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \right\| \|\boldsymbol{x}_i\| \\
&\quad + \sum_{j, w_j \geq \varepsilon/n} \left\| \frac{e^{cs(\boldsymbol{x}_i j, \boldsymbol{q}_{t-1})}}{\sum_k e^{cs(\boldsymbol{x}_i k, \boldsymbol{q}_{t-1})}} - \frac{\mathbf{1}_{(\boldsymbol{x}_i \in \mathbb{D}_{benign})}}{\sum_{k=1}^{n} \mathbf{1}_{(\boldsymbol{x}_k \in \mathbb{D}_{benign})}} \right\| \|\boldsymbol{x}_j\| \\
&\leq \left((n-m) \max\left(\frac{e^{2c} - e^{c(2-\varepsilon')} - \frac{m}{n-m}}{(n-m)e^{c(2-\varepsilon')} + m}, \frac{1}{(n-m)e^{c(2-\varepsilon')} + m} \right) \right. \\
&\quad \left. + \frac{m}{(n-m)e^{c(2-\varepsilon')} + m} \right) \max_{l, w_l \geq \varepsilon/n} (\|\boldsymbol{x}_l\|) \\
&= \frac{\max\left((n-m)e^{c(2-\varepsilon')}(e^{c\varepsilon'} - 1), n \right)}{(n-m)e^{c(2-\varepsilon')} + m} \max_{l, w_l \geq \varepsilon/n} (\|\boldsymbol{x}_l\|) \\
&\leq \max\left(e^{c\varepsilon'} - 1, \frac{n}{(n-m)e^{c(2-\varepsilon')} + m} \right) \max_{l, w_l \geq \varepsilon/n} (\|\boldsymbol{x}_l\|)
\end{aligned} \tag{14}$$

The first equality is due to the design of the algorithm. In the second equality, we denote $tr(*)$ to be the truncation function that yields zero if $* < \varepsilon/n$. The second term of the second equality is due to the definition of robust mean μ_{robust}. The third line is obtained by applying triangle inequality and splitting the terms for $\boldsymbol{x}_i$ from benign clients and $\boldsymbol{x}_j$ from attackers. Some terms for attackers are zeroed out if they have a weight smaller than ε/n. We can observe from Equation 13 that the weight w_j for a attacker is less than ε/n if $\varepsilon' < 2 - \frac{1}{c} \ln\left(\frac{1/\varepsilon - m/n}{1 - m/n}\right)$. Hence, the error term due to some attackers can be dropped. The fourth line is obtained by dropping the error term due to the attacker and plugging in Equation 12. The fifth line is simplification. The sixth line drops the m term in the denominator of the previous line. Note that the effect of dropping the m term would be minimal if we have a large c.

Here we have finished the proof on the error bound of the robust estimate. The bound can be tighter if we do not drop the m term in the last line. However, we present the current looser bound for readability. Next, we are going to prove the condition for improving the robust estimate.

In the best case where we have an arbitrarily small ε', our algorithm can at best achieve an error bounded of

$$\begin{aligned}
&\frac{n}{(n-m)e^{c(2-\varepsilon')} + m} \max_{l, w_l \geq \varepsilon/n} (\|\boldsymbol{x}_l\|) \\
&\xrightarrow{\varepsilon' \to 0} \quad \frac{1}{(1 - m/n)e^{2c} + m/n} \max_{l, w_l \geq \varepsilon/n} (\|\boldsymbol{x}_l\|).
\end{aligned} \tag{15}$$

The higher the fraction of attackers, the larger the best error bound we can achieve. If there are less than 50% attackers, the upper bound of the relative error $\frac{\|\boldsymbol{q}_t - \boldsymbol{\mu}_{robust}\|}{\max_i \|\boldsymbol{x}_i\|}$ is controlled by $2e^{-2c}$, which is about 10^{-9} for $c = 10$.

In general, if we want our algorithm to give a better approximation than the previous iteration (with a rate $\gamma < 1$), then we require

$$\begin{aligned}
&\max\left(e^{c\varepsilon'}-1, \frac{n}{(n-m)e^{c(2-\varepsilon')}+m}\right)\max_{l,w_l\geq\varepsilon/n}(\|\boldsymbol{x}_l\|)\\
&\quad\leq \gamma\|\boldsymbol{q}_{t-1}-\boldsymbol{\mu}_{robust}\|\\
&\quad\leq \gamma\delta\max_l(\|\boldsymbol{x}_l\|).
\end{aligned}$$

The second inequality is the condition on q_{t-1} in Equation 9.

Hence, we require

$$\begin{aligned}
e^{c\varepsilon'}-1 &\leq \gamma\delta\\
\varepsilon' &\leq \frac{1}{c}\ln(\gamma\delta+1) = \frac{\gamma\delta}{c}+O(\gamma^2\delta^2)
\end{aligned} \tag{16}$$

and

$$\begin{aligned}
\frac{n}{(n-m)e^{c(2-\varepsilon')}+m} &\leq \gamma\delta\\
\varepsilon' &\leq 2-\frac{1}{c}\ln\left(\frac{\gamma^{-1}\delta^{-1}-m/n}{1-m/n}\right)
\end{aligned} \tag{17}$$

The equality in (16) is due to Taylor expansion.

It means that we require our attention module to approximate the similarity measure $h'(k,q)$ with an error of at most $\frac{\gamma\delta}{c}$ if we have a large c. For example, we require $\varepsilon' \leq 1\%$ if we want to bound the relative error to 10% in one iteration. □

In practice, the true similarity measure $h_*(k,q)$ that identifies the attackers may not be the same as the $h'(k,q)$ we approximate based on the training data. In this case, the error term ε' may not be small. The estimation could have a larger error according to the bound in the first part of Lemma 2 and it may not improve throughout iterations since it may violate the condition in the second part of Lemma 2. Moreover, the higher the fraction of attackers, the worse we can do in the best case according to the bound. Fortunately, the first part of Lemma 2 suggests that using a smaller scale factor c may still give a good approximation even the similarity measure was not approximated well. In the extreme case when $c = 0$, our approximation reduces to simple mean. In other words, when the attackers behave differently from what we simulated in the training (such as a new attack), using a conservative value of the scale factor c could prevent false detection of attacker and false rejection of benign client.