Defending against Backdoors in Federated Learning with Robust Learning Rate

Mustafa Safa Ozdayi, Murat Kantarcioglu, Yulia Gel
The University of Texas at Dallas
{mustafa.ozdayi, muratk, ygl}@utdallas.edu [*]

Abstract

Federated learning (FL) allows a set of agents to collaboratively train a model without sharing their potentially sensitive data. This makes FL suitable for privacy-preserving applications. At the same time, FL is susceptible to adversarial attacks due to decentralized and unvetted data. One important line of attacks against FL is the backdoor attacks. In a backdoor attack, an adversary tries to embed a backdoor functionality to the model during training that can later be activated to cause a desired misclassification. To prevent backdoor attacks, we propose a lightweight defense that requires minimal change to the FL protocol. At a high level, our defense is based on carefully adjusting the aggregation server's learning rate, *per dimension* and *per round*, based on the sign information of agents' updates. We first conjecture the necessary steps to carry a successful backdoor attack in FL setting, and then, explicitly formulate the defense based on our conjecture. Through experiments, we provide empirical evidence that supports our conjecture, and we test our defense against backdoor attacks under different settings. We observe that either backdoor is completely eliminated, or its accuracy is significantly reduced. Overall, our experiments suggest that our defense significantly outperforms some of the recently proposed defenses in the literature. We achieve this by having minimal influence over the accuracy of the trained models. In addition, we also provide convergence rate analysis for our proposed scheme.

1 Introduction

Federated learning (FL) (McMahan et al. 2016) has been introduced as a distributed machine learning protocol. Through FL, a set of agents can collaboratively train a model without sharing their data with each other, or any other third party. This makes FL suitable to settings where data privacy is desired. In this regard, FL differs from the traditional distributed learning setting in which data is first centralized at a place, and then distributed to the agents (Dean et al. 2012; Li et al. 2014).

At the same time, FL has been shown to be susceptible to *backdoor attacks* (Bhagoji et al. 2019; Bagdasaryan et al. 2020). In a backdoor attack, an adversary disturbs the training process to make the model learn a *targeted misclassification functionality* (Chen et al. 2017; Shafahi et al. 2018; Liu et al. 2018). In centralized setting, this is typically done by *data poisoning*. For example, in a classification task involving cars and planes, the adversary could label all blue cars in the training data as plane in an attempt to make the model to classify blue cars as plane at the inference/test phase. In FL, since the data is decentralized, it is unlikely that an adversary could access all the training data. Thus, backdoor attacks are typically carried through *model poisoning* in the FL context (Bhagoji et al. 2019; Bagdasaryan et al. 2020; Sun et al. 2019). That is, the adversary tries to construct a malicious update that encodes the backdoor in a way such that, when it is aggregated with other agents' updates, the aggregated model exhibits the backdoor.

In this work, we study backdoor attacks against deep neural networks in FL setting, and formulate a defense. Our solution is based on carefully adjusting the learning rate of the aggregation server during the training. Through experiments, we illustrate that our defense can deter backdoor attacks significantly. Further, this achieved with only minimal degradation in the trained model's accuracy in both i.i.d. and non-i.i.d. settings. We provide empirical evidence justifying the effectiveness of our defense, and also theoretically analyze its convergence properties. In summary, our work significantly outperforms some of the existing defenses in the literature, and succeeds in scenarios where they fail.

The rest of the paper is organized as follows. In Section 2, we provide the necessary background information. In Section 3, we explain our defense, and in Section 4, we illustrate the performance of our defense under different experimental settings. In Section 5, we discuss and elaborate upon the results of our experiments, and finally, in Section 6, we provide a few concluding remarks.

2 Background

2.1 Federated Learning (FL)

At a high level, FL is multi-round protocol between an aggregation server and a set of agents in which agents jointly train a model. Formally, participating agents try to minimize

[*]The research reported herein was supported in part by NIH award 1R01HG006844, NSF awards, CNS-1633331, CNS-1837627, OAC-1828467, IIS-1939728, DMS-1925346, CNS-2029661 and ARO award W911NF-17-1-0356
Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

the average of their loss functions

$$\arg\min_{w\in R^d} f(w) = \frac{1}{K}\sum_{k=1}^{K} f_k(w),$$

where f_k is the loss function of kth agent. For example, for neural networks, f_k is typically empirical risk minimization under a loss function L such as cross-entropy, i.e.,

$$f_k(w) = \frac{1}{n_k}\sum_{j=1}^{n_k} L(x_j, y_j; w),$$

with n_k being the total number of samples in agent's dataset and (x_j, y_j) being the jth sample.

Concretely, FL protocol is executed as follows: at round t, server samples a subset of agents S_t, and sends them w_t, the model weights for the current round. Upon receiving w_t, kth agent initializes his model with the received weight, and trains for some number of iterations, e.g., via stochastic gradient descent (SGD), and ends up with weights w_t^k. The agent then computes his update as $\Delta_t^k = w_t^k - w_t$, and sends it back to the server. Upon receiving the update of every agent in S_t, server computes the weights for the next round by aggregating the updates with an aggregation function $\mathbf{g}\colon R^{|S_t|\times d} \to R^d$ and adding the result to w_t. That is, $w_{t+1} = w_t + \eta \cdot \mathbf{g}(\{\Delta_t\})$ where $\{\Delta_t\} = \cup_{k\in S_t}\Delta_t^k$, and η is the server's learning rate. For example, original FL paper (McMahan et al. 2016) and many subsequent papers on FL (Bhagoji et al. 2019; Bagdasaryan et al. 2020; Sun et al. 2019; Bonawitz et al. 2017; Geyer, Klein, and Nabi 2017) consider weighted averaging to aggregate updates. In this context, this aggregation is referred as Federated Averaging (FedAvg), and yields the following update rule,

$$w_{t+1} = w_t + \eta\frac{\sum_{k\in S_t} n_k \cdot \Delta_t^k}{\sum_{k\in S_t} n_k}. \tag{1}$$

In practice, rounds can go on indefinitely, as new agents can keep joining the protocol, or until the model reaches some desired performance metric (e.g., accuracy) on a validation dataset maintained by the server.

2.2 Backdoor Attacks and Model Poisoning

Training time attacks against machine learning models can roughly be classified into two categories: targeted (Bhagoji et al. 2019; Bagdasaryan et al. 2020; Chen et al. 2017; Liu et al. 2018), and untargeted attacks (Blanchard et al. 2017; Bernstein et al. 2018). In untargeted attacks, the adversarial task is to make the model converge to a sub-optimal minima or to make the model completely diverge. Such attacks have been also referred as *convergence attacks*, and to some extend, they are easily detectable by observing the model's accuracy on a validation data.

On the other hand, in targeted attacks, adversary wants the model to misclassify only a set of chosen samples with minimally affecting its performance on the main task. Such targeted attacks are also known as *backdoor attacks*. A prominent way of carrying backdoor attacks is through *trojans* (Chen et al. 2017; Liu et al. 2018). A trojan is a carefully crafted pattern that is leveraged to cause the desired misclassification. For example, consider a classification task over cars and planes and let the adversarial task be making the model classify blue cars as plane. Then, adversary could craft a brand logo, put it on *some* of the blue car samples in the training dataset, and only mislabel those as plane. Then, potentially, model would learn to classify blue cars with the brand logo as plane. At the inference time, adversary can present a blue car sample with the logo to the model to activate the backdoor. Ideally, since the model would behave correctly on blue cars that do not have the trojan, it would not be possible to detect the backdoor on a clean validation dataset.

In FL, the training data is decentralized and the aggregation server is only exposed to model updates. Given that, backdoor attacks are typically carried by constructing malicious updates. That is, adversary tries to create an update that encodes the backdoor in a way such that, when it is aggregated with other updates, the aggregated model exhibits the backdoor. This has been referred as *model poisoning* attack (Bhagoji et al. 2019; Bagdasaryan et al. 2020; Sun et al. 2019). For example, an adversary could control some of the participating agents in a FL instance and train their local models on trojaned datasets to construct malicious updates.

2.3 Robust Aggregation Methods

Several works have explored using techniques from robust statistics to deter attacks in FL. At a high level, these works tried replacing averaging with robust estimators[1] such as coordinate-wise median, geometric median, α-trimmed mean, or a variant/combination of such techniques (Yin et al. 2018; Pillutla, Kakade, and Harchaoui 2019; Blanchard et al. 2017; Mhamdi, Guerraoui, and Rouault 2018). However, to the best of our knowledge, the primary aim of these defenses are to deter convergence attacks.

In contrast, a recent work (Sun et al. 2019) has shown FedAvg can be made robust against backdoor attacks in some settings when it is coupled with weight-clipping and noise addition as introduced in (Geyer, Klein, and Nabi 2017). Concretely, server inspects updates, and if the L_2 norm of an update exceeds a threshold M, server clips the update by dividing it with an appropriate scalar. Server then aggregates clipped updates and adds Gaussian noise to the aggregation. In this case, the update rule can be written as,

$$w_{t+1} = w_t + \eta\left(\frac{\sum_{k\in S_t} n_k \cdot \frac{\Delta_t^k}{max(1, \|\Delta_t^k\|_2/M)}}{\sum_{k\in S_t} n_k} + \mathcal{N}(0, \sigma^2 M^2)\right). \tag{2}$$

Another recent work (Fung, Yoon, and Beschastnikh 2020) tries to make FL robust by introducing a per-client learning rate rather than having a single learning rate at the server side, yielding the following update rule,

$$w_{t+1} = w_t + \frac{\sum_{k\in S_t} \alpha_k^t \cdot n_k \cdot \Delta_t^k}{\sum_{k\in S_t} n_k}. \tag{3}$$

where $\alpha_k^t \in [0, 1]$ is the kth agent's learning rate for the tth round. The exact details of how learning rates are computed

[1]Informally, a statistical estimator is said to be robust if it cannot be skewed arbitrarily in presence of outliers (Huber et al. 1972).

can be found in Algorithm 1 of the respective paper. Though, at a high level, the algorithm tries to assign lower learning rates to updates whose directions are similar, as given by cosine similarity. The rationale of this defense is that, assuming adversary's agents share the common backdoor task, their updates will be more similar among themselves than honest updates. Under this assumption, the algorithm will assign lower learning rates to malicious updates, and reduce their effects. For example, if there are two identical updates, the algorithm assigns 0 as learning rate to both updates. However, as we observe experimentally in Section 4, their assumption does not hold in some realistic settings for FL. That is, if local data distributions of honest agents exhibit some similarity, algorithm cannot distinguish the adversarial agents and end up assigning everyone either the same, or very similar learning rates throughout the training process.

Finally in (Bernstein et al. 2018), authors develop a communication efficient, distributed SGD protocol in which agents only communicate the signs of their gradients. In this case, server aggregates the received signs and returns the sign of aggregation to the agents who locally update their models using it. We refer their aggregation technique as *sign aggregation*, and in FL setting, it yields the following update rule,

$$w_{t+1} = w_t + \eta\big(\operatorname{sgn} \sum_{k \in S_t} \operatorname{sgn}(\Delta_t^k)\big), \tag{4}$$

where sgn is the element-wise sign operation. Although authors show their approach is robust against certain adversaries who carry convergence attacks, e.g., by sending random signs, or by negating the signs of their gradients, in Section 4, we show that it is susceptible against backdoors attacks.

3 Robust Learning Rate

Backdoor Task vs Main Task Let $\Delta_{adv}, \Delta_{hon}$, be the aggregated updates of adversarial, and honest agents respectively. Ideally, Δ_{adv} should steer the parameters of the model to w_{adv}, which ideally minimizes the loss on both the main, and the backdoor attack task. At the same time, Δ_{hon} would want to move the model parameters towards w_{hon} that only minimizes the loss on main task. Our main conjecture is that, assuming w_{hon} and w_{adv} are different points, Δ_{adv} and Δ_{hon} will most likely differ in the directions they specify at least for some dimensions. As we show next, assuming a bound on the number of adversarial agents, we can ensure the model moves away from w_{adv}, and moves toward w_{hon}, by tuning the server's learning rate based on sign information of updates.

Robust learning rate (RLR) Following the above insight, we construct a defense which we denote as *robust learning rate* (RLR) by extending the approach proposed in (Bernstein et al. 2018). In order to move the model towards a particular direction, for each dimension, we require a sufficient number of votes, in form of signs of the updates. Concretely, we introduce a hyperparameter called *learning threshold* θ at the server-side. For every dimension where the sum of signs of updates is less than θ, the learning rate is multiplied by -1. This is *to maximize the loss on that dimension rather than minimizing it*. That is, with a learning threshold of θ, the learning rate for the i^th dimension is given by,

$$\eta_{\theta,i} = \begin{cases} \eta & \left|\sum_{k \in S_t} \operatorname{sgn}(\Delta_{t,i}^k)\right| \geq \theta, \\ -\eta & \text{otherwise.} \end{cases} \tag{5}$$

For example, consider FedAvg and let η_θ denote the learning rate vector over all dimensions, i.e., $[\eta_{\theta,1}, \eta_{\theta,2}, \ldots, \eta_{\theta,d}]^\top$. Then, the update rule with a robust learning rate takes the form,

$$w_{t+1} = w_t + \eta_\theta \odot \frac{\sum_{k \in S_t} n_k \cdot \Delta_t^k}{\sum_{k \in S_t} n_k}, \tag{6}$$

where $\odot$ is the element-wise product operation. Note that, since we only adjust the learning rate, the approach is agnostic to the aggregation function. For example, we can trivially combine it with update clipping and noise addition as in (2).

To illustrate how this might help to maximize adversary's loss, we consider a simple example where the local training consists of a single epoch of full-batch gradient descent. In this case, update of k^th agent is just the negative of his gradients, i.e., $\Delta_t^k = w_t^k - w_t = (w_t - \nabla f_k(w_t)) - w_t = -\nabla f_k(w_t)$. Then, aggregated updates is just the average of negative of agents' gradients, i.e., $-g_{avg}$. Therefore, if sum of the signs at a dimension i is below θ, that dimension is updated as $w_{t,i} = w_{t,i} + \eta \cdot g_{avg,i}$. Otherwise, it is updated as $w_{t,i} = w_{t,i} - \eta \cdot g_{avg,i}$. So we see that, for dimensions where the sum of signs is below θ, we are moving towards the direction of gradient, and hence, attempting to maximize loss. For other dimensions, we are moving towards the negative of gradient and attempting to minimize the loss as usual. Therefore, assuming number of adversarial agents is sufficiently below θ, the model would try to move away from w_{adv}, and would try to move towards w_{hon}.

Convergence Rate We now turn to deriving the convergence rate for full-batch FedAvg with RLR. Let $f_k(w) = \mathbb{E}_{D_k}[f_k(w, \xi_k)]$ be the loss function of k^th agent, where D_k is its distribution[2] and ξ_k is randomness caused by the local batch variability. We use $\mathbb{E}$ to denote expectation in respect to all random variables. Let g_k be the gradient of the k^th agent at the t^th rounds, i.e. $g_k^t = \nabla f_k(w_{t-1}^k, \xi_k^t)$, and $\mathbb{E}_{D_k}(g_k^t | \mathcal{F}_t) = \nabla f_k(w_{t-1}^k)$ where $\mathcal{F}_t$ is a filtration generated by all random variables at step t, i.e. a sequence of increasing σ-algebras $\mathcal{F}_s \subseteq \mathcal{F}_t$ for all $s < t$. Finally, following Bernstein et al. (2018), we assume that for all $t, k \in \mathbb{Z}$ each component of the stochastic gradient vector g_k^t has a unimodal distribution that satisfies population weighted symmetry (Wolfe 1974). In particular, let W be a random variable symmetric around zero, i.e., $Pr(W \leq -w) = Pr(W \geq w)$ for each $w > 0$. We now consider a family of asymmetric distributions which are constructed by distorting an arbitrary symmetric distribution with a scalar parameter $\beta > 0$ such that $Pr(W_\beta = 0) = Pr(W = 0)$ and for all $w > 0$ $Pr(W_\beta \leq -w) = 2Pr(W \geq w)/(1+\beta)$

[2]Note that D_i and D_j are not necessarily identical for two different agents i and j

and $Pr(W_\beta \geq w) = 2\beta Pr(W \geq w)/(1+\beta)$, or equivalently for all $w > 0$

$$Pr(W_\beta \geq w) = \beta Pr(W_\beta \leq -w). \tag{7}$$

Condition (7) is referred to as population weighted symmetry (Wolfe 1974). For a case of $\beta = 1$, (7) reduces to a standard symmetric distribution and corresponds to the assumption 4 of Bernstein et al. (2018). For $\beta \neq 1$ (7) describes a class of asymmetric distributions (Rosenbaum and Silber 2009). As such, (7) allows us to consider a broader class of distributions than distributions which are symmetric around the mean as in the case of Bernstein et al. (2018).

Assumption 1 Gradient is Lipschitz continuous for each agent $k = 1, \dots K$ and $L > 0$

$$||\nabla f_k(x) - \nabla f_k(y)|| \leq L||x-y||, \quad \forall x, y \in \mathbb{R}^d.$$

Assumption 2 Variance for each agent $k = 1, \dots, K$ is bounded,

$$\mathbb{E}_{D_k}||\nabla f_k(x, \xi_k^t) - \nabla f_k(x)|| \leq \sigma^2, \quad \forall x \in \mathbb{R}^d, \forall k \in \mathbb{Z}^+$$

Assumption 3 Random variables ξ_k^t are independent for all $k, t \in \mathbb{Z}^+$.

Theorem 1 (Convergence Rate) Let for all $i, k, t \in \mathbb{Z}^+$, $0 \leq Pr(1 - I_{|\sum_{k \in S_t} \text{sgn}(\Delta_{t,i}^k)| \geq \theta} | \mathcal{F}_t) \leq p_0 < 0.25, 0 < \nu \leq (1-p_0)/L$ and $E||w_t^k|| < M$, where $M > 0$ is a universal clipping upper bound. Then under Assumptions 1-3, we have the following convergence rate for our robust learning rate scheme

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}||\nabla f(\hat{w}_t)||^2 \leq \frac{2}{\eta T}(f(\hat{w}_0) - f^*) + L^2M^2 + \frac{L\eta\sigma^2}{n},$$

where $\hat{w}_t = 1/n\sum_{k=1}^n w_t^k$. See Appendix[3] for the proof of the theorem.

4 Experiments

In this section, we first illustrate the performance of our defense, and then provide some empirical justification for its effectiveness via experimental evaluation. Our implementation is done using PyTorch (Paszke et al. 2019), and the code is available at *https://github.com/TinfoilHat0/Defending-Against-Backdoors-with-Robust-Learning-Rate*.

The general setting of our experiments are as follows: we simulate FL for R rounds among K agents where F fraction of them are corrupt. The backdoor task is to make the model misclassify instances from a *base class* as *target class* by using trojan patterns. That is, a model having the backdoor classifies instances from base class with trojan pattern as target class (see Figure 1). To do so, we assume an adversary who corrupts the local datasets of corrupt agents by adding a trojan pattern to P fraction of base class instances and re-labeling them as target class. Other than that, adversary cannot view and modify updates of honest agents, or cannot influence the computation done by honest agents and the aggregation server. At each round, the server uniformly samples $C \cdot K$ agents for training where $C \leq 1$. These agents locally train for E epochs with a batch size of B before sending their updates. Upon receiving and aggregating updates, we measure three key performance metrics of the model on validation data: validation accuracy, base class accuracy and backdoor accuracy. Validation and base class accuracies are computed on the validation data that comes with the used datasets, and the backdoor accuracy is computed on a poisoned validation data that is constructed by (i) extracting all base class instances from the original validation data, and (ii) adding them the trojan pattern and re-labeling them as the target class. We measure the performance of the following five aggregation methods: (i) FedAvg (equation 1), (ii) FedAvg with *our proposed robust learning rate scheme:* RLR (equation 6), (iii) coordinate-wise median (comed), (iv) FoolsGold (equation 3), and (v) sign aggregation (equation 4). We also measure the performance of these aggregations under the proposed defense in (Sun et al. 2019), i.e., combining aggregations with weight-clipping and noise addition, to see if these techniques provide any robustness for each aggregation under our attack setting. Furthermore, in Appendix, we provide results when comed and sign aggregation are combined with RLR.

When there is a L_2 clipping threshold M on updates, we assume M is public and every agent runs projected gradient descent to minimize their losses under this restriction, i.e., an agent ensures his update's L_2 norm is bounded by M by monitoring the L_2 norm of his model during training and clips its weights appropriately. Finally we use the same model as in (Sun et al. 2019), a 5-layer convolutional neural network consisting of about 1.2M parameters with the following architecture: two layers of convolution, followed by a layer of max-pooling, followed by two fully-connected layers with dropout. Hyperparameters used in all experiments can be found in Appendix.

4.1 IID Setting

We start with a setting where data is distributed in i.i.d. fashion among agents. Concretely, we use the Fashion MNIST (Xiao, Rasul, and Vollgraf 2017) dataset, and give each agent an equal number of samples from the training data via uniform sampling. In Figure 2, we plot the training curves of FedAvg, and FedAvg with RLR, and report the final accuracies reached in each setting in Table 1. Results reported in Table 1 shows that, compared to baselines, our proposed RLR scheme provides significant protection against the backdoor attacks.

4.2 Non-IID Setting

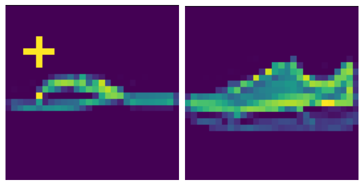
We now move on to a more realistic setting for FL in which data is distributed in non-i.i.d. fashion among agents. For this setting, we use the Federated EMNIST dataset from the LEAF benchmark (Caldas et al. 2018). In this dataset, digits 0-9 are distributed across 3383 users and each user has possibly a different distribution over digits. Similar to the i.i.d. case, we plot training curves for FedAvg and FedAvg with RLR (Figure 3). Table 1 reports the final accuracy results for each setting. Again the results reported indicate that our

[3]For the Appendix, please refer to the full version of our paper at *https://arxiv.org/abs/2007.03767*

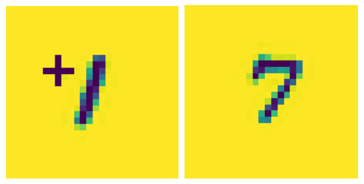
Aggregation	M	σ	Backdoor (%)	Validation (%)	Base (%)
FedAvg-*No Attack*	0	0	1	**93.5**	98.5
FedAvg	0	0	100	93.4	98.5
FedAvg	4	1e-3	100	93.2	**99.1**
FoolsGold	0	0	100	93.1	98.9
FoolsGold	4	1e-3	100	93.3	98.5
Comed	0	0	100	92.8	99.0
Comed	4	1e-3	99.5	92.8	98.4
Sign	0	0	100	92.9	98.7
Sign	4	1e-3	99.7	93.1	98.6
FedAvg with RLR	0	0	**0**	92.9	98.3
FedAvg with RLR	4	1e-3	0.5	92.2	97.4

Aggregation	M	σ	Backdoor (%)	Validation (%)	Base (%)
FedAvg*-*No Attack*	0	0	21.1	**98.6**	**99.1**
FedAvg	0	0	99.3	98.5	99.0
FedAvg	0.5	1e-3	99.2	98.0	98.7
FoolsGold	0	0	98.5	98.9	99.5
FoolsGold	0.5	1e-3	99.1	97.9	98.6
Comed	0	0	82.3	96.3	98.4
Comed	0.5	1e-3	95.2	95.5	98.1
Sign	0	0	99.8	97.6	98.7
Sign	0.5	1e-3	99.7	97.8	98.5
FedAvg with RLR	0	0	3.4	94.8	97.6
FedAvg with RLR	0.5	1e-3	**0.4**	93.2	97.7

Table 1: Final backdoor, validation and base class accuracies for different aggregations in i.i.d. (left) and non-i.i.d. (right) settings. Lowest backdoor, highest validation and base class accuracies are highlighted in **bold**. FedAvg-No Attack corresponds to our baseline where we use FedAvg with no attackers. See Appendix for additional experiments under different combinations of M and σ, and our justification for the chosen values.



(a)



(b)

Figure 1: Samples from trojaned base classes and corresponding target classes. Trojan pattern is a 5-by-5 plus sign that is put to the top-left of objects. For i.i.d. case (a), backdoor task is to make model classify trojaned sandals as sneakers. For non-i.i.d. case (b), it is to make model classify trojaned digit 1s as digit 7s. Note that original images are in grayscale, these figures are normalized as they appear in training/validation dataset. We also repeat the experiments we present here under *three additional trojan patterns* and report the results in Appendix.

defense provides the best protection against the attack with minimal degradation on the validation accuracy.

4.3 Removing Backdoor During Training

During experiments, we observed that, FedAvg with RLR rate performs substantially better than other methods in terms of preventing the backdoor task, but it also reduces convergence speed. Therefore, we wonder if one can start without RLR, and then switch to RLR at some point during the training, e.g., when the model is about to converge, to clean any possible backdoors from the model. Our experiments indicate that this is the case. In the interest of space, we provide results in Appendix, however they suggest that one can start without RLR and later switch to RLR when the model is about to converge, and/or a backdoor attack is suspected, to clean the model of backdoor during training. Overall, this improves the time to convergence when compared to using RLR right from the beginning.

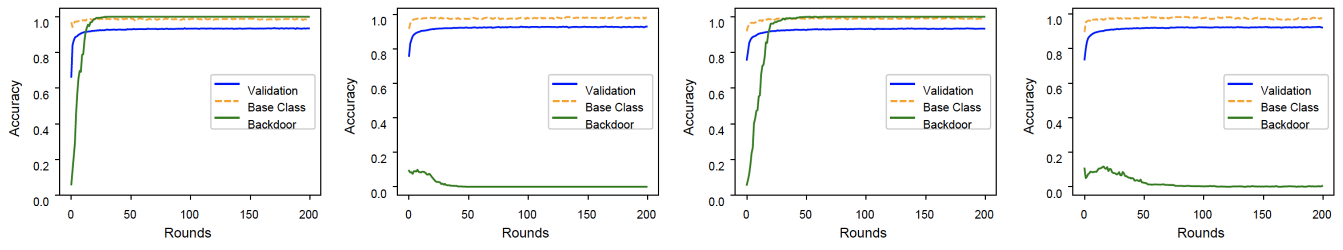
4.4 Analyzing Our Defense via Parameter and Feature Attributions

We now aim to explain why our defense works and provide some empirical justification for its effectiveness. First, recall our conjecture from Section 3 where we basically argue that the adversary has to overcome the influence of honest agents to embed the backdoor to model. More concretely, in our scenario, adversary tries to map the base class instances with trojan pattern to the target class (adversarial mapping) where as honest agents try to map them to the base class (honest mapping). If we had a way to quantify the influence of agents on the model, regarding the mapping of trojaned inputs, we would expect the model to exhibit the backdoor if the influence of adversary is greater than of the total influence of honest agents. Given that, we designed a simple experiment to quantify the influence of agents, and test this conjecture empirically. In the interest of space, we defer the details of this experiment to Appendix, but we note that it is mainly based on doing *parameter attribution* on the model to find out which parameters are most important to adversarial/honest mapping, and then tracking how they are updated over the rounds. In Figure 4, we can see that with RLR, honest agents' influence overcome the adversarial agents' for the backdoor task.

Second, we do a *feature attribution* experiment which is concerned with discovering features of an input that are important to a model's prediction. Particularly, we pick an arbitrary sample from our poisoned validation set that is correctly classified (as base class) by the model when it is trained with FedAvg with RLR, but incorrectly classified (as target class) when it is trained FedAvg. Figure 5 illustrates that, resulting feature maps on no attack and with our defense scenario are similar. This shows, our defense successfully prevents the model from focusing on the trojan pattern.

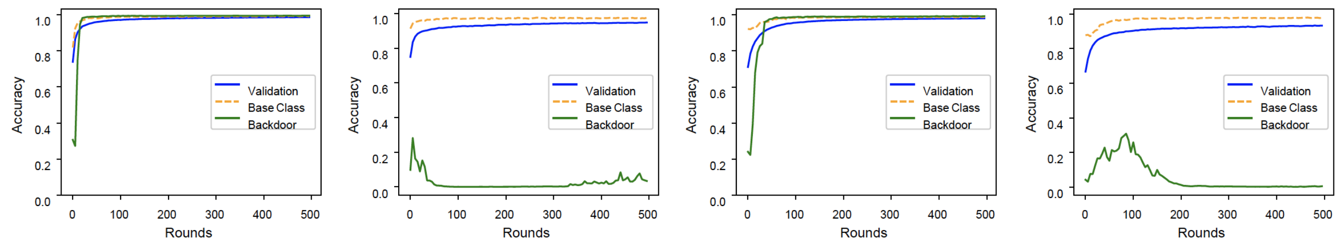
4.5 Distributed Backdoor Attacks

Finally, we briefly test our defense against a recent, novel type of backdoor attack introduced in (Xie et al. 2019). The main idea of this attack is to partition the pixels of a trojan between the agents of the adversary, and through that, ensuring the resulting malicious updates to be less different than honest' updates to make attack more stealthy. For example, if adversary has four agents, the plus pattern can be partitioned accross these four agents such that, each adversar-



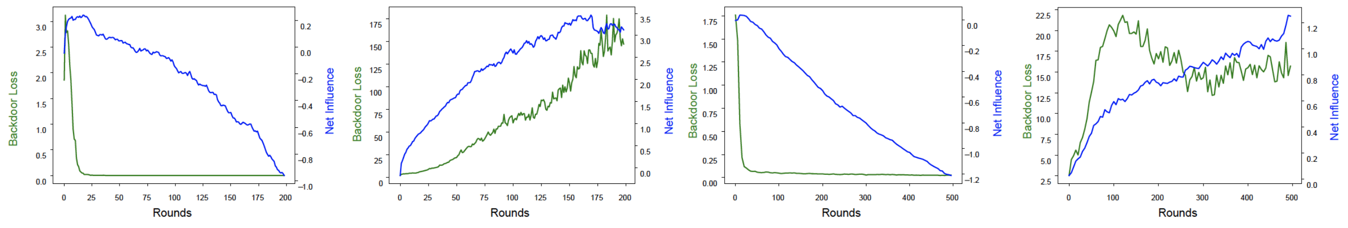
(a) (b) (c) (d)

Figure 2: Training curves for FedAvg and FedAvg with RLR in i.i.d. setting. From left-to-right: (a) FedAvg, (b) FedAvg with RLR, (c) FedAvg under clipping&noise, (d) FedAvg with RLR under clipping&noise. As can be seen, FedAvg is weak against the attack even with clipping&noise. On the other hand, FedAvg with RLR prevents the backdoor with or without clipping&noise. Using clipping and noise addition could be a desirable property in contexts where differential privacy is applied, or against attackers who try to make the model diverge by sending arbitrarily large values.



(a) (b) (c) (d)

Figure 3: Plots for FedAvg and FedAvg with RLR in non-i.i.d. setting. From left-to-right: (a) FedAvg, (b) FedAvg with RLR, (c) FedAvg under clipping&noise, (d) FedAvg with RLR under clipping&noise.



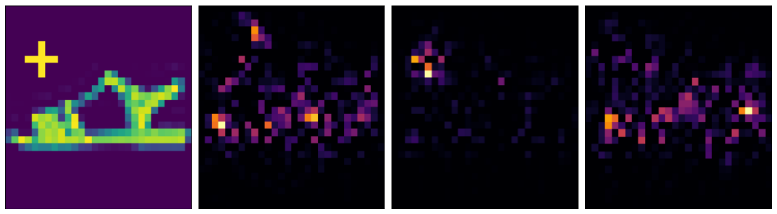
(a) (b) (c) (d)

Figure 4: Results of parameter attribution experiments. From left-to-right: (a) FedAvg, (b) FedAvg with RLR in i.i.d. setting, and (c) FedAvg, (d) FedAvg with RLR in non-i.i.d. setting. Net influence is the cumulative sum of differences between the influences of honest agents and the adversarial agents for the mapping of trojaned samples. As can be seen, net influence is loosely correlated with the backdoor loss. With RLR, net influence is positive, indicating that honest agents' influence is greater than adversarial agents. This causes backdoor loss to increase, and hence, preventing the backdoor. On the other hand, without RLR, net influence quickly becomes negative and backdoor loss decreases. This results in a successful backdoor attack.

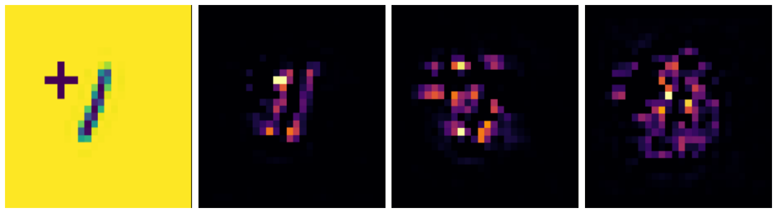
ial agent applies only a vertical/horizontal part of the plus. In case the backdoor is successful, the model would still misclassify the samples with the complete plus pattern. We test this attack only against FedAvg with RLR, as other defenses already fail on default backdoor attacks, on CIFAR10 dataset (Krizhevsky, Nair, and Hinton 2009). Table 2 indicates our defense performs well against distributed backdoor attacks too.

5 Discussion

Our experiments show that our approach significantly reduces the effectiveness of trojan pattern backdoor attacks. One can wonder that, how it performs with respect to the so-called semantic backdoors (a.k.a label-flipping) attacks. In these attacks, the adversary simply flips the label of the base class instances to a desired target label without adding a trojan pattern. In FL setting, it has been shown that successfully carrying such attacks require *boosting* (Bhagoji et al. 2019). That is, after training on a poisoned dataset, adversary has to multiply the resulting update with a large constant to overcome the effect of honest agents. Naturally, this results in adversarial updates having a large norm, and as shown in (Sun et al. 2019), weight-clipping and noise addition significantly deters these attacks. Since our defense is compatible with clipping and noise addition, it can also deter such attacks. In fact, our experiment show that, trojan backdoors are strictly more powerful than semantic backdoors in FL context as an adversary does not need to use boosting with them.



(a) IID setting



(b) Non-IID setting

Figure 5: Feature maps (FM) for i.i.d. and non-i.i.d. settings on a trojaned sample given by Gradient SHAP (Lundberg and Lee 2017). Leftmost image is the sample input from poisoned validation data, and to its right we present FMs in the following order: FM of model trained using FedAvg without any attack, FM of model trained using FedAvg under attack, FM of model trained using FedAvg with RLR under attack. For no attack case, important pixels are either on or around the actual objects. For i.i.d. setting, model predicts the sample correctly as sandals with 100% confidence, and for non-i.i.d., model predicts the digit 1 with 99.2% confidence. For no defense scenario, we can see that model's attention has shifted towards the trojan pattern. This is especially very visible for i.i.d. setting where the model almost completely focuses on the trojan. In i.i.d. case, model predicts the sample as sneakers with 100% confidence, and in non-i.i.d. case, model predicts the digit as 7 with 91.2% confidence. Finally, we see that with robust learning rate, the model's attention has been shifted back to the actual objects to some extent. Now, model predicts the sample as sandals with 100% confidence in i.i.d. case, and it predicts the digit as 1 with 91.2% confidence in non-i.i.d. case.

Aggregation	Backdoor (%)	Validation (%)	Base (%)
FedAvg-*No Attack*	6.6	79.0	89.4
FedAvg	88.6	79.4	87.5
FedAvg with RLR	9.0	77.5	87.8

Aggregation	Backdoor (%)	Validation (%)	Base (%)
FedAvg-*No Attack*	6.3	76.6	87.7
FedAvg	61.7	76.6	78.2
FedAvg with RLR	8.5	71.8	83.3

Table 2: Backdoor attack on i.i.d.-partitioned CIFAR10. Backdoor task is to classify dogs (base class) with plus pattern as horses (target class). Left table is for regular backdoor attack, and right table is for distributed backdoor attack where plus pattern is partitioned to 4 adversarial agents out of 40 agents. See Appendix for details.

Finally, we ask if an adversary who knows our defense scheme can devise a clever attack. At a high level, as long as the θ parameter is set appropriately (see Appendix for a discussion on hyperparameters), and adversary's local loss function differs from the honest against, the scheme will try to move the model from the directions the adversarial update specifies. Adversary could try to make his loss function more in-line with honest agents' via some modification, but then this will likely result in his attack losing effectiveness. We emphasize that our approach does not "magically" finds the adversary, and negates his update by multiplying it with $-\eta$. Therefore, the adversary cannot by-pass our defense just by negating his loss function (see Appendix for an experimental evaluation of this attack).

6 Conclusion

In this work, we studied FL from an adversarial perspective, and constructed a simple defense mechanism, particularly against backdoor attacks. The key idea behind our defense was adjusting the aggregation server's learning rate, per dimension and per round, based on the sign information of agents' updates. Through experiments we present above and in Appendix, we illustrate that our defense reduces backdoor accuracy substantially with a minimal degradation in the overall validation accuracy. Overall, it outperforms some of the recently proposed defenses in the literature. As a final comment, we believe the insights behind our defense are also related to training in non-i.i.d. setting, even in the presence of no adversaries. Because, the differences in local distributions can cause updates coming from different agents to steer the model towards different directions over the loss surface. As a future work, we plan to analyze how RLR influences performance of models trained in different non-i.i.d. settings.

References

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

Bernstein, J.; Zhao, J.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291* .

Bhagoji, A. N.; Chakraborty, S.; Mittal, P.; and Calo, S. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, 634–643.

Blanchard, P.; Guerraoui, R.; Stainer, J.; et al. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 119–129.

Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H. B.; Patel, S.; Ramage, D.; Segal, A.; and Seth, K. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 1175–1191.

Caldas, S.; Wu, P.; Li, T.; Konečnỳ, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* .

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* .

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*, 1223–1231.

Fung, C.; Yoon, C. J. M.; and Beschastnikh, I. 2020. Mitigating Sybils in Federated Learning Poisoning. *arXiv preprint arXiv:1808.04866* .

Geyer, R. C.; Klein, T.; and Nabi, M. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557* .

Huber, P. J.; et al. 1972. The 1972 wald lecture robust statistics: A review. *The Annals of Mathematical Statistics* 43(4): 1041–1067.

Krizhevsky, A.; Nair, V.; and Hinton, G. 2009. CIFAR-10 (Canadian Institute for Advanced Research) URL http://www.cs.toronto.edu/~kriz/cifar.html.

Li, M.; Andersen, D. G.; Park, J. W.; Smola, A. J.; Ahmed, A.; Josifovski, V.; Long, J.; Shekita, E. J.; and Su, B.-Y. 2014. Scaling distributed machine learning with the parameter server. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 583–598.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2018. Trojaning Attack on Neural Networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, 4765–4774.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; et al. 2016. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629* .

Mhamdi, E. M. E.; Guerraoui, R.; and Rouault, S. 2018. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927* .

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pillutla, K.; Kakade, S. M.; and Harchaoui, Z. 2019. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445* .

Rosenbaum, P. R.; and Silber, J. H. 2009. Amplification of sensitivity analysis in matched observational studies. *Journal of the American Statistical Association* 104(488): 1398–1405.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, 6103–6113.

Sun, Z.; Kairouz, P.; Suresh, A. T.; and McMahan, H. B. 2019. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963* .

Wolfe, D. A. 1974. A characterization of population weighted-symmetry and related results. *Journal of the American Statistical Association* 69(347): 819–822.

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

Xie, C.; Huang, K.; Chen, P.-Y.; and Li, B. 2019. DBA: Distributed Backdoor Attacks against Federated Learning. In *International Conference on Learning Representations*.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *International Conference on Machine Learning*, 5650–5659.

Appendices

Please refer to the full version of our paper at *https://arxiv.org/abs/2007.03767*