Preventing Backdoors in Federated Learning by Adjusting Server-side Learning Rate

Mustafa Safa Ozdayi
The University of Texas at Dallas
mustafa.ozdayi@utdallas.edu

Murat Kantarcioglu
The University of Texas at Dallas
muratk@utdallas.edu

Yulia R. Gel
The University of Texas at Dallas
ygl@utdallas.edu

Abstract

Federated Learning (FL) allows a set of agents to collaboratively train a model without sharing their data. This makes FL suitable for privacy sensitive settings. However, since the data is decentralized and unvetted, FL is particularly susceptible to adversarial attacks. One important line of attacks against FL is the backdoor attacks. In a backdoor attack, an adversary tries to embed a backdoor to the model during training. The backdoor can then be activated to cause a desired misclassification during inference. To prevent backdoor attacks, we propose a lightweight defense that requires minimal change to the FL structure. 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. In this work, we first conjecture the necessary steps to carry a successful backdoor attack in FL setting, and then, formulate our defense. We test our defense against backdoor attacks in different settings, and observe that it either completely eliminates the backdoors, or significantly reduces their accuracy. We also provide empirical justification for the effectiveness of our defense. Overall, our experiments demonstrate we outperform some of the recently proposed defenses in the literature. This is achieved by having a minimal degradation on the accuracy of the trained models.

1 Introduction

Federated Learning (FL) [1] is a distributed machine learning protocol. Through FL, a set of agents can collaboratively train a model without sharing their data with another party. This makes FL suitable to settings where data privacy is desired. At the same time, FL is susceptible to backdoor attacks [2, 3]. In a backdoor attack, an adversary disturbs the training process to make the model learn a targeted misclassification functionality [4, 5, 6]. In FL context, backdoor attacks are typically carried through *model poisoning* [2, 3, 7]. That is, the adversary tries to constructs an update that encodes the backdoor in a way such that, when the malicious update is aggregated with benign updates, the aggregated model exhibits the backdoor.

In this work, we study backdoor attacks in FL setting, and formulate a defense. Our defense is based on carefully adjusting the learning rate of the aggregation server, per round and per dimension, based on the sign information of agents' updates. Through experiments, we show our defense performs better than existing defenses over a various different settings. Furthermore, this is achieved with only minimal degradation in the trained model's accuracy. We also provide empirical justification for our defense via experiments. In summary, our work significantly outperforms the existing defenses in the literature, and succeeds in settings where they fail.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

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

2 Background

2.1 Federated Learning (FL)

FL is a multi-round machine learning protocol between an aggregation server, and a set of agents in which agents jointly train a model. Concretely, FL runs 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, k$^{\text{th}}$ 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 $\mathbf{g}\colon \mathbb{R}^{|S_t| \times d} \to \mathbb{R}^d$, and adding the result to w_t. That is, $w_{t+1} = w_t + \eta \cdot \mathbf{g}(\{\Delta_t\})$ where $\{\Delta_t\} = \bigcup_{k \in S_t} \Delta_t^k$, and η is the server's learning rate. For example, original FL paper [1] and many subsequent papers on FL [2, 3, 7, 8, 9] 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 [2, 3, 4, 6], and untargeted attacks [10, 11]. 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 are also referred as *convergence attacks*, and to some extend, they are easily detectable by observing the model's accuracy on a validation dataset.

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* [4, 6]. 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 [2, 3, 7]. 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 [13, 14, 10, 15]. However, to the best of our knowledge, the primary aim of these defenses are to deter convergence attacks.

[1] Informally, a statistical estimator is said to be robust if it cannot be skewed arbitrarily in presence of outliers [12].

In contrast, a recent work [7] has shown FedAvg can be made robust against backdoor attacks in some settings when it is coupled with differential privacy as introduced in [9]. 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,

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

Another recent work [16] tries to make FL robust to backdoor attacks 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 k[th] agent/update's learning rate for the t[th] round. The exact details of how learning rates are computed 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 with respect to honest updates. Under this assumption, the algorithm will assign lower learning rates to malicious updates, and reduce their effectiveness. 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. In fact, for our experimental setting, it did not provide any backdoor attack protection over FedAvg.

Finally in [11], 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 signs 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 While carrying the backdoor attack, the adversarial agents try to steer the parameters of the model to w_{adv}, which ideally minimizes the loss on both the main and the backdoor attack tasks. At the same time, the honest agents would try to move the model parameters, collectively, 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, the updates that are sent by honest agents and adversarial agents will most likely differ in the directions they specify. 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 [11]. In order to move the model in a particular direction, we require a sufficient number of votes, in form of updates' sign, for each dimension. Concretely, we introduce a hyperparameter named *learning threshold* θ, and for every dimension where the sum of signs of updates is less than θ, we multiply the learning rate by -1. This is *to maximize the loss on that dimension rather than minimizing it*. In short, the learning rate for the i[th] dimension is given by,

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

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 RLR gives becomes,

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

where $\odot$ is the element-wise product operation. Note that, since we only adjust the server's learning rate, our 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 update 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}.

4 Experiments

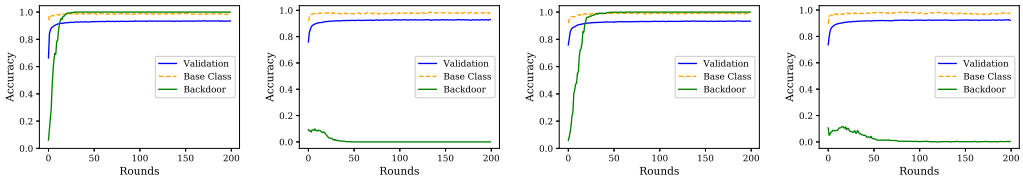
We now illustrate the performance of our defense via experiments. The general setting of our experiments are as follows: we simulate FL for R rounds among K agents where F fraction of them are corrupt. The backdoor task is to make the model misclassify instances from a *base class* as *target class* 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 $C \leq 1$. Those 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 a 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) and (iv) FoolsGold (equation 3) (v) sign aggregation (equation 4). We also measure the performance of these aggregations under the proposed defense in [7], 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 B, 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 in [7], 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 A.

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 [17] 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. Our results reported in Table 1 show that, compared to other aggregations, RLR provides significant protection against the backdoor attack.

(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 more different trojan patterns* and report the results in Appendix B.

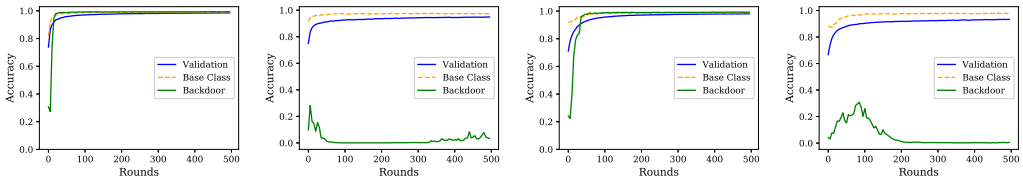


(a) (b) (c) (d)

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.

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



(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.

4.3 Distributed Backdoor Attacks

Finally, we briefly test our defense against a recent, novel type of backdoor attack introduced in [19]. 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 adversarial 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 against the default backdoor attack, on CIFAR10 dataset [20]. Table 2 indicates our defense performs well against distributed backdoor attacks too.

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 B for additional experiments under different combinations of M and σ, and our justification for the chosen values.

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 A for details of this experiment.

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* [2]. 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 [7], 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. We also 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 A 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 B for an experimental evaluation of this attack.

Finally, we note that we provide empirical justification for the effectiveness of our defense via *parameter*, and *feature* attribution methods in Appendix C. We also illustrate our defense can *clean* a successfully inserted backdoor from a model during the training in Appendix B.1.

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 server's learning rate, per dimension and per round, based on the sign information of updates coming from the agents. Through experiments we present above and in Appendix B, 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. This is 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 non-i.i.d. settings.

Acknowledgements

The research reported herein was supported in part by NIH award 1R01HG006844, NSF awards CICI-1547324, IIS-1633331, CNS-1837627, OAC-1828467, IIS-1939728 and ARO award W911NF-17-1-0356.

References

[1] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[2] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643, 2019.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948, 2020.

[4] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[5] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

[6] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 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, 2018.

[7] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[8] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

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

[10] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[11] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291*, 2018.

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

[13] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659, 2018.

[14] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[15] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. *arXiv preprint arXiv:1802.07927*, 2018.

[16] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2020.

[17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[18] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[19] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.

[20] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 2009.

[21] Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

[22] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

Appendices

A Hyperparameters of Experiments

We remind the notation we introduced at the beginning of Section 4 and report the hyperparameters of our experiments. We also briefly discuss our choices.

- R: Number of rounds
- K: Total number of agents
- F: Fraction of corrupt agents
- P: Fraction of trojaned samples in a corrupt agent's dataset
- C: Fraction of selected agents for training in a round
- E: Number of epochs in local training
- B: Batch size of local training
- η: Server's learning rate
- θ: Threshold for RLR (see equation 5)

R	K	F	P	C	E	B
200	10	0.1	0.5	1	2	256

R	K	F	P	C	E	B
500	3383	1	0.5	0.01	10	64

Table 3: Hyperparameters for all i.i.d. (left) and non-i.i.d. (right) experiments. In addition to what is presented in table, we set η to 1e-3 when sign aggregation is used, and to 1 otherwise. Finally, θ is set to 4 in i.i.d. case, and is set to 7 in non-i.i.d. case when RLR used.

In both cases, we set E and B to some values that loosely help us to run as many experiments as quickly as possible in our system. F was arbitrarily fixed to 0.1 so as the values for C. We set P to 0.5 after trying different values and observing that the backdoor accuracy rises the quickest under that value to simulate a strong adversary. Setting the value of θ is non-trivial. Technically, it could be any value between $K.F + 1$, $K - K.F$. In our experiments, setting it to 4 in i.i.d. setting seemed to provide us the best trade-off between backdoor prevention and the drop in validation accuracy. For non-i.i.d. setting, in expectation, we had 3 corrupt agents per round, and setting θ to 7 gave us a similar trade-off as in i.i.d. case.

Finally, hyperparameters for distributed backdoor attack experiment on CIFAR10 (see Figure 2) is given in Table 4.

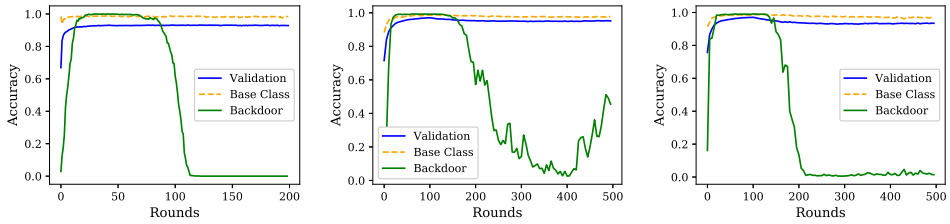
R	K	F	P	C	E	B
100	40	0.1	0.5	1	2	256

Table 4: Hyperparameters for all CIFAR10 experiments. In addition to what is presented in table, we set η to 1 Finally, θ is set to 8 when RLR used. Under distributed backdoor attack, four lines of the plus pattern is partitioned across four adversarial agents.

B Extra Experiments

B.1 Cleaning 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. 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 as seen in Figure 4. Overall, this improves the time to convergence when compared to using RLR right from the beginning.



(a) IID setting (b) Non-IID setting, $\theta = 7$ (c) Non-IID setting, $\theta = 8$

Figure 4: Cleaning the backdoor during training by activating the RLR when the model is about to converge. For i.i.d. setting, we activate the RLR when the model's validation accuracy is above 93%. This occurs at round 41, and at that round, backdoor accuracy is at 100%. At round 124, the backdoor accuracy is 0% with a validation accuracy of 93.1%. Final validation and base class accuracies are 92.8% and 98.5% respectively. On the other hand, for non-i.i.d. setting, we activate RLR when the validation accuracy is above 97%. However in (b), we observe that using the same RLR threshold (7) that we used in Section 4.2 fails to prevent the backdoor now. Backdoor accuracy falls to 2% from 99% and then rises up to 45% by round 500. Yet, if we increase the threshold to 8, we observe it performs substantially better in (c). In this case, final backdoor accuracy is 1.4%, and final validation and base class accuracies are 93.4% and 97%, respectively.

B.2 Negating Loss Function Attack

We briefly show that an adversary cannot by-pass our defense simply by negating the sign of his loss function. We explained, at an intuitive level, why such an attack would fail in the second paragraph of Section 5, and in Table 5, we provide the experimental confirmation.

Setting	Backdoor (%)	Validation (%)	Base (%)
IID	0.7	90.8	98.4
Non-IID	3	92.2	98.0

Table 5: Results of what happens when adversary tries to by-pass RLR by negating his loss function. The aggregation is FedAvg with RLR. We can see this attack does not give anything useful to adversary for backdoor task when results are compared with Table 1. However, since adversary tries to maximize his original loss function due to negation, this causes norms of his updates to be very large. So, we had to use clipping at the server-side with values of $M = 4$ for i.i.d. setting, and $M = 0.5$ for non-i.i.d. setting to prevent model from diverging.

B.3 Experiments for all M, σ combinations and Trojan Patterns

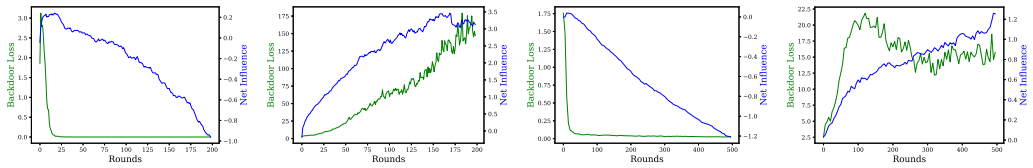
Regarding our choice of M, σ in i.i.d. case, we observed L_2 norm of updates of honest agents during training in baseline which happened to be floating around 6. So, we ran experiments for $M = 6, 4, 2$. For σ, we tried $\sigma = 1e-4, 1e-3, 5e-3$ and stopped increasing it after observing that training becomes imbalanced at $5e-3$. For non-i.i.d. case, L_2 norm of updates of honest agents were about $1.5-2$, so we have chosen $M = 1, 0.5, 0.25$ and used the same σ values from values from i.i.d. case. In addition to the plus trojan pattern we used in the main body, we repeated our experiments under three more trojans which are the same as in [6]: a square, a copyright logo, an Apple logo, placed to the bottom-right of objects (see Figure 7). We provide explicit results for all possible settings for FedAvg in Tables 8 and 9. For other aggregations, we provide the configuration for the setting where the backdoor accuracy is the lowest, for each trojan, for sake of brevity in Tables 6 and 7. Results indicate FedAvg combined with RLR outperforms the other techniques. However, comed and sign also perform well especially under square,copyright and Apple logo trojans in non-i.i.d. setting(Table 7).Another point to note is, sign aggregation seems to be performing well with RLR while comed sometimes performs better without it.

C Explaining Effectiveness of RLR 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 by quantifying influences of corrupt/honest agents via *parameter attribution*. Concretely, after each round, we find the 100 most important parameters for adversarial, and honest mapping by computing the empirical Fisher Information Matrix (FIM) as done in [21]. Particularly, we compute the diagonal of FIM on the trojaned samples, labeled as target class, and take top 100 values for adversarial mapping. We do the same by computing FIM on trojaned samples labeled as base class to find out the most influential parameters for honest mapping. Then, due to RLR, some of these top 100 parameters are updated in a way to minimize the loss, and some of them are updated in a way to maximize the loss. Let S_1, S_2 be those which are updated to minimize the loss for adversarial and honest mapping, respectively. Also let S_3, S_4 be those which are updated to maximize the loss for adversarial and honest mapping, respectively. Then, we quantified the net adversarial influence as $I_{adv} = \|S_1 \setminus S_2\|_2$ - $\|S_3 \setminus S_4\|_2$ and the net honest influence as $I_{hon} = \|S_2 \setminus S_1\|_2$ - $\|S_4 \setminus S_3\|_2$. Finally, the net influence is then given by $I_{hon} - I_{adv}$ which is plotted in blue in Figure 5.

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 6 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.



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

Figure 5: 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.

Table 6: Results for comed/sign (left/right) in i.i.d. setting under different trojans.

Trojan Pattern	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
Plus	6	5e-3	No	10.5	85.8	94.0
Square	4	5e-3	No	1.9	88.7	95.9
Copyright	6	5e-3	No	5.7	85.5	95.5
Apple	4	5e-3	No	4.4	88.6	95.9

Trojan Pattern	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
Plus	6	5e-3	Yes	92.1	92.4	98.1
Square	0	1e-3	Yes	4.9	98.2	92.6
Copyright	6	5e-3	Yes	47.7	92.5	98.1
Apple	0	1e-4	Yes	8.5	92.6	98.9

Table 7: Results for comed/sign (left/right) in non-i.i.d. setting under different trojans.

Trojan Pattern	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
Plus	0.5	5e-3	Yes	10.5	94.8	98.1
Square	0.5	1e-4	Yes	0.1	94.9	98.1
Copyright	1	0	No	0.1	96.5	98.3
Apple	0	5e-3	Yes	0.1	96.2	98.4

Trojan Pattern	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
Plus	0.25	1e-3	Yes	54.2	94.6	98.3
Square	0	0	Yes	0	95.4	98.3
Copyright	0	1e-4	Yes	0	94.6	98.6
Apple	0	0	Yes	0	95.4	98.5

Table 8: Results for FedAvg in i.i.d. setting under different trojans. Top-left/right: plus/square, bottom-left/right: copyright/Apple logo.

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	1	93.5	98.5
FedAvg	0	0	No	100	93.4	98.5
FedAvg	0	0	Yes	0	92.9	98.3
FedAvg	0	1e-4	No	100	93.3	98.8
FedAvg	0	1e-4	Yes	0	92.5	98.0
FedAvg	0	1e-3	No	100	93.2	98.4
FedAvg	0	1e-3	Yes	0	92.4	98.0
FedAvg	0	5e-3	No	100	93.2	98.6
FedAvg	0	5e-3	Yes	0	92.5	98.4
FedAvg	2	0	No	100	93.3	98.8
FedAvg	2	0	Yes	0	92.7	98.0
FedAvg	2	1e-4	No	100	93.5	99.2
FedAvg	2	1e-4	Yes	0	92.9	98.8
FedAvg	2	1e-3	No	100	93.4	98.8
FedAvg	2	1e-3	Yes	0	92.6	98.3
FedAvg	2	5e-3	No	99.8	91.2	97.6
FedAvg	2	5e-3	Yes	4	89.6	95.5
FedAvg	4	0	No	100	93.4	99.0
FedAvg	4	0	Yes	0	93.3	98.3
FedAvg	4	1e-4	No	100	93.6	99.0
FedAvg	4	1e-4	Yes	0	92.9	98.2
FedAvg	4	1e-3	No	100	93.2	99.1
FedAvg	4	1e-3	Yes	0.5	92.2	97.4
FedAvg	4	5e-3	No	99.0	89.0	96.3
FedAvg	4	5e-3	Yes	3.1	87.2	94.0
FedAvg	6	0	No	100	93.4	98.5
FedAvg	6	0	Yes	0	93.0	97.9
FedAvg	6	1e-4	No	100	93.5	99.1
FedAvg	6	1e-4	Yes	0	92.9	98.1
FedAvg	6	1e-3	No	100	93.1	98.6
FedAvg	6	1e-3	Yes	0.5	91.8	97.2
FedAvg	6	5e-3	No	92.3	85.7	93.9
FedAvg	6	5e-3	Yes	6.1	83.9	93.1

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0.7	93.2	99.0
FedAvg	0	0	No	95.0	93.3	98.5
FedAvg	0	0	Yes	0	92.6	98.5
FedAvg	0	1e-4	No	94.9	93.4	98.3
FedAvg	0	1e-4	Yes	0	92.4	98.3
FedAvg	0	1e-3	No	95.9	93.4	98.8
FedAvg	0	1e-3	Yes	0	92.5	97.9
FedAvg	0	5e-3	No	94.0	93.6	99.0
FedAvg	0	5e-3	Yes	0	92.6	98.3
FedAvg	2	0	No	96.5	93.4	98.9
FedAvg	2	0	Yes	0	92.7	98.4
FedAvg	2	1e-4	No	94.2	93.4	98.7
FedAvg	2	1e-4	Yes	0	92.8	98.1
FedAvg	2	1e-3	No	93.8	93.6	99.3
FedAvg	2	1e-3	Yes	0	92.6	98.0
FedAvg	2	5e-3	No	42.6	91.2	97.5
FedAvg	2	5e-3	Yes	1.4	89.8	96.3
FedAvg	4	0	No	95.0	93.5	98.8
FedAvg	4	0	Yes	0.1	93.2	98.3
FedAvg	4	1e-4	No	93.7	93.5	99.0
FedAvg	4	1e-4	Yes	0	93.4	97.7
FedAvg	4	1e-3	No	94	93.4	99.0
FedAvg	4	1e-3	Yes	0.5	92.3	97.5
FedAvg	4	5e-3	No	33.6	88.8	97.1
FedAvg	4	5e-3	Yes	3.1	87.2	94.0
FedAvg	6	0	No	93.2	93.4	98.5
FedAvg	6	0	Yes	0	93.1	97.7
FedAvg	6	1e-4	No	94	93.4	99.2
FedAvg	6	1e-4	Yes	0	93.1	98.5
FedAvg	6	1e-3	No	94.3	92.9	98.5
FedAvg	6	1e-3	Yes	0.5	91.4	98.1
FedAvg	6	5e-3	No	25.4	85.3	94.5
FedAvg	6	5e-3	Yes	6.1	83.3	88.2

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0.4	93.6	98.8
FedAvg	0	0	No	98	93.3	98.8
FedAvg	0	0	Yes	98.7	92.7	98.3
FedAvg	0	1e-4	No	98.0	93.3	98.8
FedAvg	0	1e-4	Yes	0	92.9	98.2
FedAvg	0	1e-3	No	97.6	93.6	98.9
FedAvg	0	1e-3	Yes	0	92.6	97.9
FedAvg	0	5e-3	No	98.9	93.0	99.0
FedAvg	0	5e-3	Yes	0	93.0	97.6
FedAvg	2	0	No	98.1	93.4	98.9
FedAvg	2	0	Yes	0	92.7	97.8
FedAvg	2	1e-4	No	98.2	93.7	98.7
FedAvg	2	1e-4	Yes	0	92.7	98.4
FedAvg	2	1e-3	No	97.4	93.4	98.8
FedAvg	2	1e-3	Yes	0	92.6	97.5
FedAvg	2	5e-3	No	93.2	91.2	97.1
FedAvg	2	5e-3	Yes	1.5	90.0	96.6
FedAvg	4	0	No	98.7	93.6	99.2
FedAvg	4	0	Yes	0.1	93.1	98.1
FedAvg	4	1e-4	No	97.5	93.4	99.0
FedAvg	4	1e-4	Yes	0	93.1	97.9
FedAvg	4	1e-3	No	98.0	93.0	98.8
FedAvg	4	1e-3	Yes	0	92.1	97.9
FedAvg	4	5e-3	No	94.7	88.7	95.3
FedAvg	4	5e-3	Yes	7.6	86.8	90.5
FedAvg	6	0	No	98.7	93.4	98.5
FedAvg	6	0	Yes	0	93.1	98.1
FedAvg	6	1e-4	No	97.6	93.3	98.8
FedAvg	6	1e-4	Yes	0	93.0	98.2
FedAvg	6	1e-3	No	97.1	93.0	98.5
FedAvg	6	1e-3	Yes	0.5	91.8	97.6
FedAvg	6	5e-3	No	93.9	85.4	91.2
FedAvg	6	5e-3	Yes	9.2	82.8	86.8

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0.4	93.3	98.7
FedAvg	0	0	No	90.1	93.4	98.7
FedAvg	0	0	Yes	0	92.9	97.6
FedAvg	0	1e-4	No	90.7	93.3	99.0
FedAvg	0	1e-4	Yes	0	92.7	98.2
FedAvg	0	1e-3	No	90.8	93.3	98.6
FedAvg	0	1e-3	Yes	0	92.5	97.8
FedAvg	0	5e-3	No	91.3	93.4	98.9
FedAvg	0	5e-3	Yes	0.1	92.6	97.7
FedAvg	2	0	No	86.7	93.2	99.1
FedAvg	2	0	Yes	0	92.7	98.2
FedAvg	2	1e-4	No	85.1	93.4	98.8
FedAvg	2	1e-4	Yes	0	92.8	98.1
FedAvg	2	1e-3	No	87.4	93.5	99.0
FedAvg	2	1e-3	Yes	0	92.5	97.9
FedAvg	2	5e-3	No	54.8	91.2	97.7
FedAvg	2	5e-3	Yes	1.5	89.8	97.0
FedAvg	4	0	No	89.5	93.5	99.0
FedAvg	4	0	Yes	0.1	93.0	97.7
FedAvg	4	1e-4	No	88.0	93.4	98.5
FedAvg	4	1e-4	Yes	0	93.2	97.9
FedAvg	4	1e-3	No	84.2	93.3	98.8
FedAvg	4	1e-3	Yes	0.3	92.1	97.4
FedAvg	4	5e-3	No	50.4	88.5	94.2
FedAvg	4	5e-3	Yes	2.6	87.5	96.3
FedAvg	6	0	No	90.6	93.4	99.0
FedAvg	6	0	Yes	0.1	93.2	98.3
FedAvg	6	1e-4	No	89.4	93.3	98.4
FedAvg	6	1e-4	Yes	0	92.5	97.3
FedAvg	6	1e-3	No	91.8	93.0	97.8
FedAvg	6	1e-3	Yes	0.1	92.1	97.7
FedAvg	6	5e-3	No	20.3	85.2	91.7
FedAvg	6	5e-3	Yes	6.1	84.4	91.2

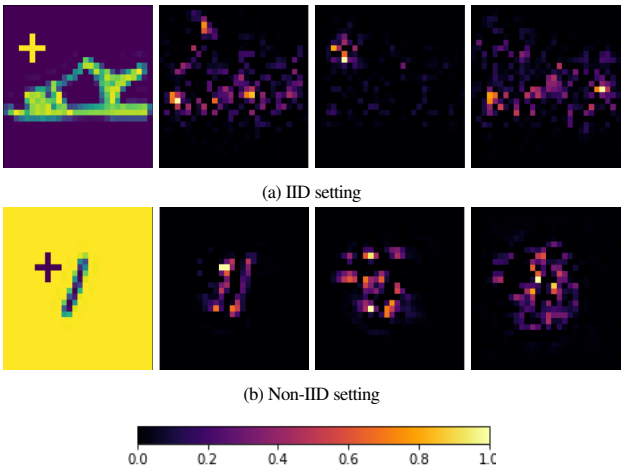
Table 9: Results for FedAvg in non-i.i.d. setting under different trojans. Top-left/right: plus/square, bottom-left/right: copyright/Apple logo.

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0	98.5	99.0
FedAvg	0	0	No	99.5	98.5	98.9
FedAvg	0	0	Yes	0	95.2	97.7
FedAvg	0	1e-4	No	99.6	98.5	99.0
FedAvg	0	1e-4	Yes	0	95.3	98.1
FedAvg	0	1e-3	No	99.6	98.5	99.0
FedAvg	0	1e-3	Yes	0	95.1	98.3
FedAvg	0	5e-3	No	99.6	98.5	99.1
FedAvg	0	5e-3	Yes	0	95.1	98.0
FedAvg	0.25	0	No	99.5	96.7	98.4
FedAvg	0.25	0	Yes	0	90.0	97.0
FedAvg	0.25	1e-4	No	99.6	96.7	98.6
FedAvg	0.25	1e-4	Yes	0	89.8	97.7
FedAvg	0.25	1e-3	No	99.6	96.6	98.3
FedAvg	0.25	1e-3	Yes	0	89.9	96.6
FedAvg	0.25	5e-3	No	99.5	96.6	98.5
FedAvg	0.25	5e-3	Yes	0	89.8	96.1
FedAvg	0.5	0	No	99.5	98.0	98.8
FedAvg	0.5	0	Yes	0	93.8	97.9
FedAvg	0.5	1e-4	No	99.5	98.1	98.8
FedAvg	0.5	1e-4	Yes	0	93.7	97.18
FedAvg	0.5	1e-3	No	99.5	98.1	98.7
FedAvg	0.5	1e-3	Yes	0	93.8	97.5
FedAvg	0.5	5e-3	No	99.4	97.7	98.4
FedAvg	0.5	5e-3	Yes	0	94.4	97.4
FedAvg	1.0	0	No	99.6	98.4	99.0
FedAvg	1.0	0	Yes	0	94.9	97.8
FedAvg	1.0	1e-4	No	99.4	98.5	98.8
FedAvg	1.0	1e-4	Yes	0	94.8	97.8
FedAvg	1.0	1e-3	No	99.6	98.3	98.9
FedAvg	1.0	1e-3	Yes	0	94.6	97.4
FedAvg	1.0	5e-3	No	99.2	98.5	99.1
FedAvg	1.0	5e-3	Yes	28.4	94.9	97.5

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0.7	93.2	99.0
FedAvg	0	0	No	95.0	93.3	98.5
FedAvg	0	0	Yes	0	92.6	98.5
FedAvg	0	1e-4	No	94.9	93.4	98.3
FedAvg	0	1e-4	Yes	0	92.4	98.3
FedAvg	0	1e-3	No	95.9	93.4	98.8
FedAvg	0	1e-3	Yes	0	92.5	97.9
FedAvg	0	5e-3	No	94.0	93.6	99.0
FedAvg	0	5e-3	Yes	0	92.6	98.3
FedAvg	0.25	0	No	96.5	93.4	98.9
FedAvg	0.25	0	Yes	0	92.7	98.4
FedAvg	0.5	1e-4	No	94.2	93.4	98.7
FedAvg	0.5	1e-4	Yes	0	92.8	98.1
FedAvg	0.25	1e-3	No	93.8	93.6	99.3
FedAvg	0.25	1e-3	Yes	0	92.6	98.0
FedAvg	0.25	5e-3	No	42.6	91.2	97.5
FedAvg	0.25	5e-3	Yes	1.4	89.8	96.3
FedAvg	0.5	0	No	95.0	93.5	98.8
FedAvg	0.5	0	Yes	0.1	93.2	98.3
FedAvg	0.5	1e-4	No	93.7	93.5	99.0
FedAvg	0.5	1e-4	Yes	0	93.4	97.7
FedAvg	0.5	1e-3	No	94	93.4	99.0
FedAvg	0.5	1e-3	Yes	0.5	92.3	97.5
FedAvg	0.5	5e-3	No	33.6	88.8	97.1
FedAvg	0.5	5e-3	Yes	3.1	87.2	94.0
FedAvg	1	0	No	93.2	93.4	98.5
FedAvg	1	0	Yes	0	93.1	97.7
FedAvg	1	1e-4	No	94	93.4	99.2
FedAvg	1	1e-4	Yes	0	93.1	98.5
FedAvg	1	1e-3	No	94.3	92.9	98.5
FedAvg	1	1e-3	Yes	0.5	91.4	98.1
FedAvg	1	5e-3	No	25.4	85.3	94.5
FedAvg	1	5e-3	Yes	6.1	83.3	88.2

Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0.1	98.5	99.1
FedAvg	0	0	No	99.5	98.5	99.0
FedAvg	0	0	Yes	0	94.9	97.6
FedAvg	0	1e-4	No	99.5	98.5	99.0
FedAvg	0	1e-4	Yes	0	95.1	97.8
FedAvg	0	1e-3	No	99.5	98.5	99.1
FedAvg	0	1e-3	Yes	0	95.2	97.7
FedAvg	0	5e-3	No	99.6	98.5	99.0
FedAvg	0	5e-3	Yes	0	95.2	98.1
FedAvg	0.25	0	No	99.4	96.6	98.5
FedAvg	0.25	0	Yes	0	89.3	97.1
FedAvg	0.25	1e-4	No	99.4	96.9	98.6
FedAvg	0.25	1e-4	Yes	0	89.5	97.0
FedAvg	2	1e-3	No	99.3	96.9	98.5
FedAvg	2	1e-3	Yes	0	89.2	96.3
FedAvg	0.25	5e-3	No	99.2	96.7	98.4
FedAvg	0.25	5e-3	Yes	0	90.2	96.4
FedAvg	0.5	0	No	99.4	98.0	98.9
FedAvg	0.5	0	Yes	0.0	93.2	96.9
FedAvg	0.5	1e-4	No	99.5	98.0	98.8
FedAvg	0.5	1e-4	Yes	0	93.5	97.7
FedAvg	0.5	1e-3	No	99.4	98.1	98.9
FedAvg	0.5	1e-3	Yes	0	93.4	96.9
FedAvg	0.5	5e-3	No	99.5	97.8	98.5
FedAvg	0.5	5e-3	Yes	0	94.3	97.6
FedAvg	1	0	No	99.5	98.5	99.0
FedAvg	1	0	Yes	0	95.1	98.0
FedAvg	1	1e-4	No	99.6	98.4	99.1
FedAvg	1	1e-4	Yes	0	95.0	98.1
FedAvg	1	1e-3	No	99.5	98.4	98.9
FedAvg	1	1e-3	Yes	0	95.0	98.0
FedAvg	1	5e-3	No	99.4	98.0	98.7
FedAvg	1	5e-3	Yes	0	94.9	97.7

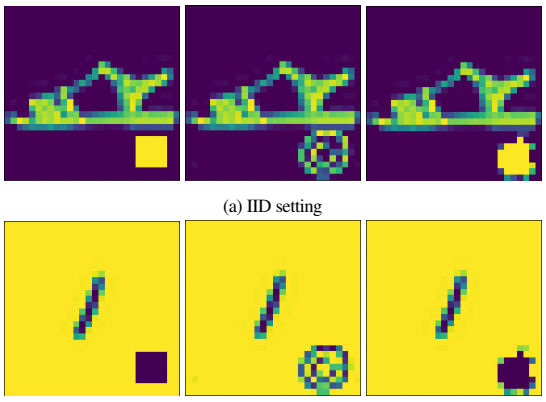
Aggregation	M	σ	RLR used?	Backdoor (%)	Validation (%)	Base (%)
FedAvg*	0	0	No	0	98.5	99.1
FedAvg	0	0	No	99.4	98.5	99.1
FedAvg	0	0	Yes	0	95.0	97.6
FedAvg	0	1e-4	No	99.6	98.5	99.0
FedAvg	0	1e-4	Yes	0	95.4	98.1
FedAvg	0	1e-3	No	99.5	98.4	99.0
FedAvg	0	1e-3	Yes	0	95.0	97.7
FedAvg	0	5e-3	No	99.3	98.5	99.0
FedAvg	0	5e-3	Yes	0	95.2	97.6
FedAvg	0.25	0	No	99.5	96.8	98.6
FedAvg	0.25	0	Yes	0	89.2	97.4
FedAvg	0.25	1e-4	No	99.5	96.5	98.6
FedAvg	0.25	1e-4	Yes	0	89.3	97.0
FedAvg	2	1e-3	No	99.4	97.0	98.5
FedAvg	2	1e-3	Yes	0	89.8	96.4
FedAvg	0.25	5e-3	No	99.2	96.4	98.6
FedAvg	0.25	5e-3	Yes	0	90.7	96.7
FedAvg	0.5	0	No	99.5	98.0	98.6
FedAvg	0.5	0	Yes	0.0	93.4	98.2
FedAvg	0.5	1e-4	No	99.5	98.0	98.8
FedAvg	0.5	1e-4	Yes	0	93.5	97.7
FedAvg	0.5	1e-3	No	99.4	98.0	98.9
FedAvg	0.5	1e-3	Yes	0	93.5	96.9
FedAvg	0.5	5e-3	No	99.3	97.8	98.6
FedAvg	0.5	5e-3	Yes	0	94.2	97.2
FedAvg	1	0	No	99.5	98.5	99.0
FedAvg	1	0	Yes	0	95.1	98.0
FedAvg	1	1e-4	No	99.6	98.4	99.0
FedAvg	1	1e-4	Yes	0	94.9	97.8
FedAvg	1	1e-3	No	99.5	98.4	98.7
FedAvg	1	1e-3	Yes	0	94.9	97.9
FedAvg	1	5e-3	No	99.5	97.9	98.6
FedAvg	1	5e-3	Yes	0	94.8	98.0



(a) IID setting

(b) Non-IID setting

Figure 6: Feature maps (FM) for i.i.d. and non-i.i.d. settings on a trojaned sample given by Gradient SHAP [22]. 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.



(a) IID setting

(b) Non-IID setting

Figure 7: Extra trojans patterns from [6] as applied to datasets we use.