To appear in the IEEE Symposium on Security & Privacy Workshops, 2023

On the Pitfalls of Security Evaluation of Robust Federated Learning

Momin Ahmad Khan*⋆, Virat Shejwalkar*†, Amir Houmansadr†, Fatima M. Anwar⋆
University of Massachusetts Amherst
†{vshejwalkar, amir}@cs.umass.edu, ⋆{makhan, fanwar}@umass.edu

***Abstract*—Prior literature has demonstrated that Federated learning (FL) is vulnerable to poisoning attacks that aim to jeopardize FL performance, and consequently, has introduced numerous defenses and demonstrated their robustness in various FL settings. In this work, we closely investigate a largely overlooked aspect in the robust FL literature, i.e., the *experimental setup* used to evaluate the robustness of FL poisoning defenses. We thoroughly review 50 defense works and highlight several questionable trends in the experimental setup of FL poisoning defense papers; we discuss the potential repercussions of such experimental setups on the key conclusions made by these works about the robustness of the proposed defenses. As a representative case study, we also evaluate a recent poisoning recovery paper from IEEE S&P'23, called FedRecover. Our case study demonstrates the importance of the experimental setup decisions (e.g., selecting representative and challenging datasets) in the validity of the robustness claims; For instance, while FedRecover performs well for MNIST and FashionMNIST (used in the original paper), in our experiments it performed poorly for FEMNIST and CIFAR10.**

I. Introduction

Federated learning (FL) is an emerging learning paradigm in which data owners (called *clients*) collaboratively train a common machine learning model (called the *global model*) without sharing their private training data. In FL, a central *server* (e.g., a service provider) repeatedly collects model updates from clients that they compute using their local data, aggregates the updates using an *aggregation rule* (AGR), and finally uses the aggregated update to tune the global model, which is broadcast to a subset of the clients at the end of each FL training round. FL is increasingly adopted by various distributed platforms, in particular by Google's Gboard [1] for next word prediction, by Apple's Siri [54] for automatic speech recognition, and by WeBank [68] for credit risk predictions.

The threat of poisoning FL: The key feature of FL, i.e., collaboration between *mutually untrusted* clients, e.g., Android users or competing banks, is also the root of its susceptibility to a threat known as *poisoning*: a small fraction of FL clients, called *compromised clients*, who are either owned or controlled by a *poisoning adversary*, may act maliciously during the FL training process in order to *poison* the global model. There are three major approaches to poisoning FL: *targeted*, *backdoor*, and *untargeted* poisoning. Backdoor [8], [66] and targeted attacks [65], [10] aim to misclassify only a small set of inputs *with* or *without* specific properties, respectively. While, untargeted attacks [63], [21] aim to reduce model accuracy on all the test inputs. For further discussions, please refer to [62].

*The first two authors contributed equally to this work.

Depending on the FL setting, poisoning can have significant impact on the overall utility of FL. On the one hand, as [62] argues, ultra large-scale FL applications, e.g., Gboard and Siri, are less susceptible due to the high cost of effective poisoning. On the other hand, FL can be instrumental in numerous small- to moderate-scale applications built using popular FL libraries, e.g., FedML [26], Tensorflow Federated [3], Pytorch [53]. Adversaries can effectively poison such FL settings, e.g., network level adversaries [61] can hack communication channels to mount effective poisoning attacks.

Defenses against poisoning in FL: To mitigate the threat of poisoning in FL, community has investigated numerous defenses of various types. For example, robust aggregation rules (AGRs) aim to detect and remove malicious updates (Multi-krum [11] and Trimmed-mean [79]), certified defenses aim to provide robustness certificate (CRFL [73]), and others aim to recover from poisoning (FedRecover [14]). We refer the readers to [62] for further discussions of defense classes.

Our work: Each of the prior FL defense works claims robustness to FL poisoning under specific experimental settings (e.g., specific benchmark datasets, target attacks, etc.). However, upon closer inspection, we find multiple common, but questionable, trends in their setups and evaluations that may lead to serious misrepresentation of the claim robustness. For instance, many defenses use suboptimal and extremely slow-converging FL algorithms, e.g., FedSGD, to motivate the need for their new defenses, and many defenses do not consider adaptive and/or strong state-of-the-art (SOTA) poisoning attacks. Our work makes the following two concrete contributions.

(Contribution-1) We thoroughly review the experimental setup of 50 FL defense works, from the past 5 years, from the lens of four critical components of robustness evaluation of defenses against FL poisoning: (1) FL baselines, (2) datasets used for evaluation, (3) distribution of FL clients' data, (4) attacks evaluated against. Our evaluations **shed light on several misleading trends in the experimental setup of the evaluated FL defense works**. For example, we highlight that, while the community is aware of strong FL poisoning techniques [63], [21], [75], [9], *almost 40% of the defense papers evaluate only against a subset of naive attacks (ran-*

dom Gaussian [11], label flipping [37], sign flip [34], bit flipping [74]), which are known to perform poorly even under strong adversarial settings [62], [21], [34], [56]. Similarly, we find that *close to 50% of the defense papers consider client data to be i.i.d.*, although FL clients' data has been known to be highly heterogeneous since the inception of FL [43].

Along with the review of these works, we use Trimmed-mean AGR to empirically demonstrate how the sub-optimal or unrealistic choices of the four aforementioned components can lead to faulty conclusions about the robustness of FL defenses. For instance, we show that this defense mechanism can easily defend against a simple task like MNIST, but fails to defend properly for more difficult tasks like FEMNIST or CIFAR10: when training on MNIST using FedAVG + Trimmed-mean, SOTA Trim attack [21] reduces the accuracy of global model from 98.7% (without attack) to 96.7% with 20% malicious clients, but for FEMNIST (CIFAR10) the accuracy reduces from 82% (82%) to 65% (50%).

(Contribution-2) As a case study, we thoroughly re-evaluate a recent FL defense work from IEEE S&P'23 called FedRecover [14]. FedRecover aims to recover a global model from strong FL poisoning attacks. Our re-evaluations of FedRecover from the lens of the four aforementioned components reveal that: **(1)** When there are no poisoning attacks during recovery, FedRecover decreases the accuracy of FL on complex tasks compared to the case of normal FL training (we refer to this as *train-from-scratch* (TFS))—this is not evaluated in the original paper. For example, for FEMNIST (CIFAR10), it achieves 79% (61%) accuracy while the accuracy of regular FL training is 82% (66%). **(2)** We show that in the presence of attacks during recovery, FedRecover's performance further reduces compared to TFS, e.g., for FEMNIST (CIFAR10), FedRecover and TFS accuracies are 75% (46%) and 82% (66%), respectively. **(3)** FedRecover's performance is highly sensitive to the choice of its hyperparameters, e.g., increasing the correction period from 3 to 5 reduces FedRecover's accuracy from 81% to 74%, and decreasing the warmup period from 20 to 10 reduces accuracy from 75% to 72%.

The **key observation** of our work is that the FL robustness literature has been building up on several simplistic (and to some extent, faulty) practices in the setup of its experimental evaluations. We conclude with several concrete recommendations on the experimental setup of future works on FL robustness in Section V.

II. Background

A. Federated learning (FL)

In FL [28], [43], [31], a service provider, called *server*, trains a *global model*, θ^g, on the private data of multiple collaborating clients without directly collecting their data. In the t^{th} FL round, the server selects n out of total N clients and shares the most recent global model, i.e., θ^t_g, with them. Then, a client k uses their local data D_k and computes an update ∇^t_k and shares it with the server. Depending on how does a client compute their update, FL algorithms can be broadly divided into *FedSGD* and *FedAvg*. In FedSGD, client computes the update by sampling a subset b of their local data and computing a gradient of loss $\ell(b;\theta^t_g)$ of the global model on the subset, i.e., $\nabla^t_k = \partial\ell(b;\theta^t_g)/\partial\theta^t_g$. On the other hand, in FedAvg, a client k *fine-tunes* θ^t_g on their local data using stochastic gradient descent (SGD) for a fixed number of local epochs E and obtains updated local model θ^t_k. Then, they compute their as the difference $\nabla^t_k = \theta^t_k - \theta^t_g$ and shares ∇^t_k with the server. Next, the server computes an aggregate of all of the client updates using some aggregation rule, f_{agg}, i.e., using $\nabla^t_{\text{agg}} = f_{\text{agr}}(\nabla^t_{\{k\in[n]\}})$. Finally, the server updates the global model of the $(t+1)^{th}$ round using SGD as $\theta^{t+1}_g \leftarrow \theta^t_g + \eta\nabla^t_{\text{agg}}$; η is the server's learning rate.

Practical deployments of FL can be either **cross-device** or **cross-silo** [28]. In *cross-device FL*, N is very large (from few thousands to billions) and only a small fraction of them is chosen in each FL training round, i.e., $n \ll N$. Clients are resource constrained devices such as mobile phones, smartwatches, IoT devices, etc. While, in *cross-silo FL*, N is moderate (up to 100) and all clients are selected in each round, i.e., $n = N$. Clients are large corporations, e.g., banks, schools, hospitals, etc.

B. Brief overview of FL poisoning attacks

There are three major types of FL poisoning [62]: untargeted, targeted, and backdoor. The majority of defenses against FL poisoning aim to defend against untargeted attacks, so our work only focuses on untargeted poisoning defenses.

FL poisoning threat models. Next, we detail the untargeted poisoning threat model, i.e., goal, knowledge and capabilities of the poisoning adversary, we use in this work.

Goal: We consider an *untargeted poisoning adversary* who controls m out of N FL clients and aims to manipulate the global model such that the model will misclassify all (or most) of the inputs at test time. Unless stated otherwise, we assume that our adversary controls 20% of total FL clients.[1]

Knowledge: Following most of the defense works, we assume that the adversary knows the robust AGR that the server uses. The adversary has *partial knowledge* of federated data, i.e., the adversary knows local data only of the malicious clients they control and not of the benign clients.

Capabilities: In terms of capabilities, we consider strong *model poisoning* adversary who can directly manipulate the model updates that the malicious clients share with the server. In particular, we use two state-of-the-art model poisoning attacks: NDSS [62] and Trim [21] detailed in Section VII-C.

III. Overview of The Experimental Setup of FL Poisoning Defense Works

In this section, we review 50 works that propose defenses, also called *robust aggregation rules (AGR)*, against FL poisoning. In particular, we review them from the lens of four critical components of robustness evaluation setup. We highlight some

[1]Most defense works assume very high percentages of malicious clients to demonstrate that their defenses work even in highly adversarial settings. Hence, although unreasonable in practical FL settings [62], we follow prior defense works and use 20% malicious clients.

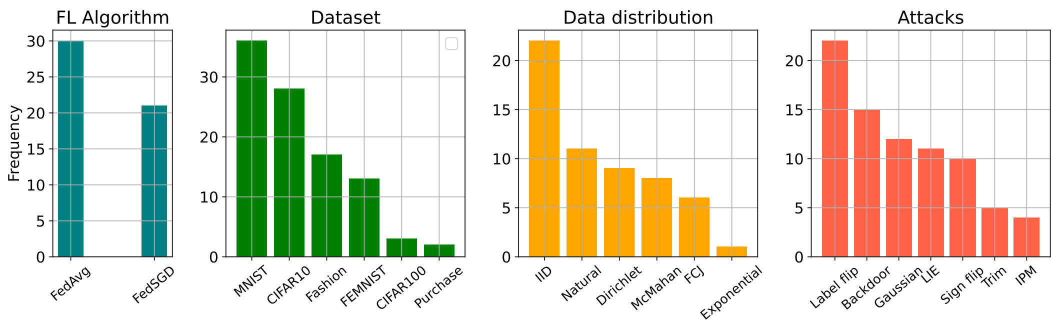


Figure 1: Frequency of choices of the four key components (FL baseline, dataset, distribution of clients' data, attacks) of robustness evaluation setup. In Section III, we discuss impacts of these choices on robustness of FL poisoning defenses.

of the most overlooked aspects of FL robustness evaluations and show how it can lead to misleading conclusions and/or false sense of security. These works are listed in Table I. The four evaluation components are: (1) FL baseline, (2) dataset used for evaluation, (3) distribution of FL clients' data, (4) attacks used for robustness evaluation. Note that, FL is a complex system with many more components impacting its robustness [62], e.g., assumptions about the server's capabilities, number of clients, type of FL (cross-silo or cross-device), and even the robustness metric. However, we only focus on the four aforementioned fundamental components. Figure 1 provides the frequency of choices of the four components made in the 50 works we review; Table I in Appendix VII-A gives the detailed classification of each work.

Below, we elaborate on the choices made in the 50 prior works. Then we showcase how some of the popular choices can lead to a false sense of security for Trimmed-mean (*TrMean*) AGR [79], [74], a classic defense mechanism that is used as a building block of many advanced AGRs [14], [82], [63]. TrMean aggregates each dimension of input updates separately. For j^{th}-dimension, it sorts the values of all updates, removes m (i.e., the number of compromised clients) of the largest and smallest values, and computes the average of the remaining values as the aggregate. Later in Section IV, we will also thoroughly re-evaluate the robustness of the latest SOTA defense, FedRecover [14].

Below, we discuss the four components in detail.

Experimental setup: We use four datasets in this work: FEMNIST, CIFAR10, MNIST and Fashion-MNIST. Due to space constraints, we defer complete details to Appendix VII-B.

(1) Choice of baseline FL algorithm: As discussed in Section II-A, all the FL algorithms can be categorized into FedSGD and FedAvg types. In FedAvg, clients *fine-tune* the global model using their local data for multiple steps as opposed a single step in FedSGD. Consequently, FedAvg achieves higher performance, faster convergence and lower communication than FedSGD, and numerous works have demonstrated this [43], [30]. Nonetheless, we observe that ***about 40% of prior works use FedSGD based slow FL algorithms for robustness evaluations***.

Consequence of the choices: In Figure 2, we plot the accuracy of FedSGD and FedAvg when combined with TrMean AGR under benign and adversarial FL settings. For FedSGD, we match accuracy in recent state-of-the-art defense works [14], [21], while for FedAvg, we tune hyperparameters to achieve much higher accuracy and only 5% to 20% of communication compared to FedSGD. We use Trim attack here.

In the benign setting, FedAvg significantly outperforms FedSGD in terms of performance, convergence and communication cost. This also reflects in adversarial setting: *FedAvg is less susceptible to untargeted poisoning*, because its faster convergence leaves significantly less time for adversary to perform poisoning. These observations apply to all the four datasets shown in Figure 2.

(2) Choice of dataset: Real-world FL tasks are very challenging and a long line of research [13], [19] devotes substantial time to design open-source datasets that resemble real-world FL datasets. Nonetheless, we observe that ***most prior works use MNIST, an extremely simple, and hence, intrinsically robust task, for robustness evaluation***; moreover, 30% of these works base majority of their conclusions on evaluations using *only* the MNIST dataset. For instance, [29], [79], [16], [69], [34], [40] use MNIST for all evaluations, while [15], [14], [38] use multiple datasets, but they also draw significant conclusions based on evaluations that uses only MNIST.

The next two most common datasets, CIFAR10 and Fashion-MNIST, are relatively more difficult. But they are far from FL datasets in that they are very well-curated and class-balanced. Unfortunately, the common ways of distributing these datasets among FL clients, e.g., in independent and identical (IID) fashion, makes the resulting FL setting less representative of real-world FL; we discuss this in more detail in next section. Finally, FEMNIST [13], a real-world FL dataset, is at the fourth place, and just 20% of the works use it for evaluations.

Consequence of the choices: The intrinsic robustness of MNIST is evident from Figure 2: Trim attack with a very high (20%) percentage of malicious clients reduces the accuracy of the MNIST-trained FL model by less than 1%. *TrMean AGR is highly robust when evaluated using MNIST, but this conclusion does not hold when we evaluate TrMean robustness using other three datasets.*

(3) Choice of data distribution: Prior works use various strategies to distribute a non-federated dataset, e.g., MNIST or CIFAR10, among FL clients. These strategies have significant impacts on robustness of AGRs. For instance, with more

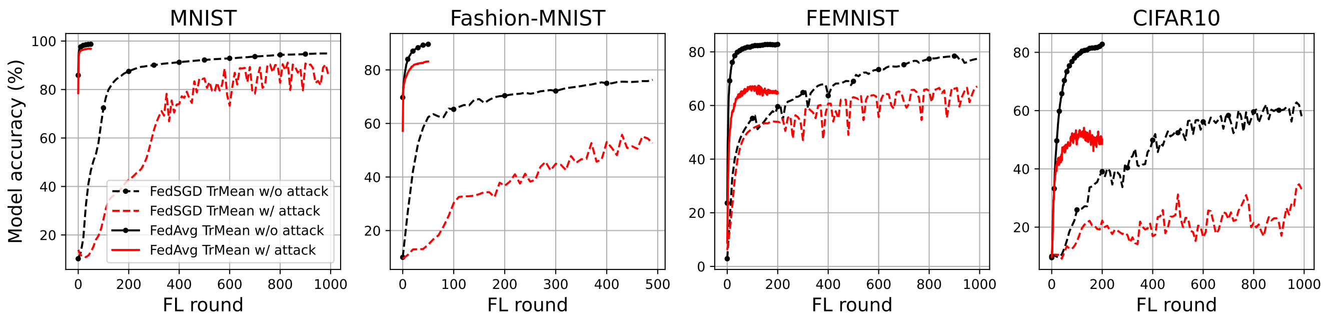


Figure 2: Comparison of the robustness of TrMean AGR when used with FedSGD and FedAvg (Section II-A). FedSGD based algorithms make TrMean more susceptible to poisoning due to their slow convergence which allows the adversary more time/rounds to poison the global model. We use Trim attack here. Please check Section II-B for details of threat model.

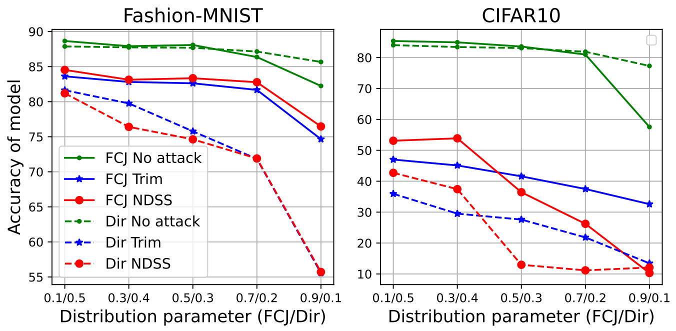


Figure 3: Impact of choice of data distribution strategy and attack on FedAvg + TrMean: **(1)** FCJ distribution [21] generates locally IID datasets which makes TrMean more robust, while Dirichlet distribution generates more heterogeneous and hence real-world datasets, hence to stress-test robust AGRs we suggest using Dirichlet distribution; note that, in both plots non-iid degree increases from left to right. **(2)** Ideally defenses should be evaluated against multiple strong attacks, but many works neglect state-of-the-art attacks, e.g., [63].

independent and identically distributed (IID) data, detecting and mitigating the impact of malicious updates becomes easier, and hence, defending against FL poisoning also becomes easier. In spite of numerous works [21], [63], [9], [80] already pointing this out, we observe in Figure 1-(c) that ***close to 50% of the works use IID data to evaluate their defenses***.

The second most common choice is *real* distribution, i.e., using real-world federated datasets, e.g., FEMNIST where each sample is already associated with a client. However, we observe this only in 22% of works that use FEMNIST, StackOverflow [2] or Shakespeare [13] data. The rest of the works artificially partition data to create FL clients. We denote the three most common artificial distributions by *FCJ* [21], *Dirichlet* [8], [58], [45] and *McMahan* [43].

Consequence of the choices: Figure 3 shows the impact of different strategies on the robustness of TrMean for Fashion-MNIST and CIFAR10 datasets. We use the two most popular synthetic data distribution strategies; FCJ and Dirichlet. By varying their distribution parameters, we can produce varying levels of non-IID datasets; for consistency, we use the parameters used in prior works [21], [14], [62].

We note that with FCJ distributed dataset, TrMean is seemingly more robust. This is because FCJ distributes data in a *partially* non-IID fashion (Figures 6), i.e., if dataset has C classes and total number of FL clients in N, then FCJ distributes data in C clusters in non-IID fashion. But within each of the C clusters, the data is IID among $\frac{N}{C}$ FL clients.

On the other hand, Dirichlet distributes data such that each client's data distribution is different. Hence, *we argue that Dirichlet produces more real-world FL datasets than FCJ*. To justify this argument, we perform statistical analyses of the FCJ and Dirichlet distributed client datasets and show that FCJ produces more IID datasets than Dirichlet; due to space limits, we defer details to Appendix VII-E. Hence, we argue to use Dirichlet distribution instead of FCJ for robustness evaluations.

(4) Choice of attacks: This is probably the most important component of the evaluation of the robustness of FL defenses; Figure 1-(d) shows the frequency of various attacks that prior works use for robustness evaluation. We note that in-spite of multiple works introducing strong FL poisoning attacks [63], [21], [75], [9], ***almost 40% of the defense papers evaluate only against a subset of naive attacks (random Gaussian [11], label flipping [37], sign flip [34], bit flipping [74])***, which are known to perform poorly even under strong adversarial settings [62], [21], [34], [56]. Ideally defenses should consider multiple strong poisoning attacks for evaluation, but unfortunately, over 95% of works *do not* evaluate against SOTA attacks, e.g., untargeted [63] and backdoor [66].

Consequence of the choices: Figure 3 shows the impact of two strong model poisoning attacks, Trim [21] and NDSS [63], on FashionMNIST and CIFAR10 trained using FedAvg. NDSS attack outperforms Trim attack, especially when the datasets are more non-IID and resemble real-world FL.

IV. Re-evaluating FedRecover

In this section, we showcase our critical evaluation on FedRecover [14] by re-evaluating it from the lens of the four components of evaluation setup discussed in Section III. We have chosen this work only as a representative of the recent literature on FL poisoning defenses.

A. Introducing FedRecover

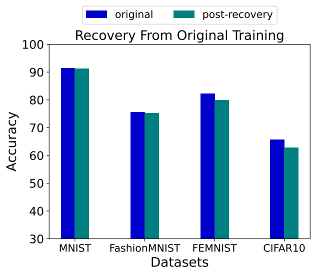
FedRecover aims to *recover* an FL global model that has been compromised by a poisoning attack. During the *original training phase*, at every round t, FedRecover server stores the model updates ∇_t^k of client k and global models θ_g^t, and uses these as *historical information* during *recovery phase*. The recovery process has broadly three phases.

During the first *warmup phase*, the server asks the clients to send their *exact updates* for the first T_w rounds. During the second *estimation phase*, the server *estimates* the client updates at each FL round; $\hat{\nabla}_k^t$ is the estimated update for client k at round t. It estimates the update by applying L-BFGS algorithm [49] on the original global model at round t ($\overline{\theta}_g^t$), original model update by client k at round t ($\overline{\nabla}_k^t$), and the recovered global model at round t ($\hat{\theta}_g^t$). The estimated model update is thus computed as $\hat{\nabla}_t^k = \overline{\nabla}_k^t + \tilde{H}_k^t(\hat{\theta}_g^t - \overline{\theta}_g^t)$, where $\tilde{H}_k^t(\hat{\theta}_g^t - \overline{\theta}_g^t)$ is the Hessian vector product. To ensure that the estimated global model $\hat{\theta}_g^t$ remains close to the accurate global model, the server, after every T_c rounds, asks the clients to send their *exact updates*; this is called *periodic correction*. Additionally, if any component of a specific client's estimated update exceeds a pre-specified *abnormality threshold*, τ, the server asks that client to send their exact update. Finally, in the last *fine-tuning phase*, for T_f rounds, the server again asks the clients to send their exact updates, ∇_k^t, to improve global model performance by preventing any estimation errors.

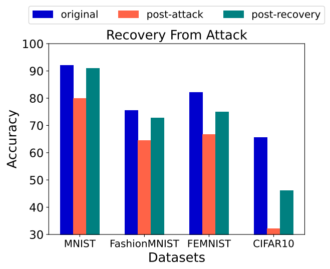
B. Critical re-evaluation of FedRecover's robustness

Re-evaluation methodology: We test FedRecover from the lens of the four aforementioned components (Section III); as we could not obtain FedRecover code from its authors, we implemented it based on [14]. Specifically, we re-evaluate FedRecover using choices for baseline FL algorithms, datasets and data distributions, that are more challenging (FEMNIST instead of MNIST) and/or more relevant (FedAvg instead of FedSGD) in practice, compared to choices in [14].

1) Choice of baselines: The original FedRecover work uses FedSGD to train a CNN on MNIST for 2000 rounds with mini-batch of 32 and learning rate of 3×10^{-4}. Consequently, FedRecover reports that Trim attack reduces the accuracy of Trimmed-mean from 96% to 81%. However, appropriate baseline, e.g., FedAVG with sufficiently tuned hyperparameters, can achieve much faster convergence: in just 50 rounds we achieve 98.7% accuracy and under Trim attack, accuracy reduces to 96.7%. Furthermore, note from Figure 2 that, for all the datasets, the accuracy of the global model in adversarial setting is higher (or same) with FedAvg than with FedSGD.



(a) No attack in original training.



(b) Attack in original training.

Figure 4: Performance of FedRecover without any attack and with Trim attack during original training.

2) Choice of datasets: FedRecover reports all of its results (except in Figure 1 of [14] where it uses four datasets) only on the MNIST data. However, due to its simplicity, MNIST is intrinsically robust to poisoning (Section III) and may give a false sense of security. Therefore, below we evaluate FedRecover under three adversarial settings (A1-3) using FEMNIST and CIFAR10 which are more challenging tasks than MNIST and FashionMNIST.

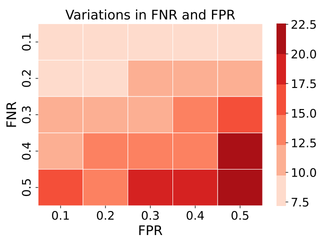
(A1) FedRecover + No attack during original training or recovery: Figure 4a shows the results. Note that this is a hypothetical scenario designed to understand how FedRecover performs in a complete absence of poisoning adversary. We note that, *even in the complete absence of attacks, FedRecover cannot completely recover for FEMNIST and CIFAR10.*

(A2) FedRecover + Trim attack only during original training: Figure 4b shows the results. Here we assume that none of the malicious clients participate during recovery phase.[2] We note that FedRecover performs well with MNIST (FashionMNIST) and achieves 91% (72.7%) accuracy compared to 80% (64%) accurate poisoned global model, while the best achievable accuracy with only the benign FL clients is 92% (75.2%). However, even without any attack during recovery, FedRecover cannot recover well for FEMNIST and CIFAR10, the differences in best achievable and FedRecover accuracies are 11% and 19%, respectively. We observe that, our results match that of [14] and we almost completely recover for MNIST and FashionMNIST, which also validates correctness of our implementation. However, as expected, due to their challenging nature, we are unable to perform perfect recovery for FEMNIST and CIFAR10 with reasonable amount of communication during recovery phase.

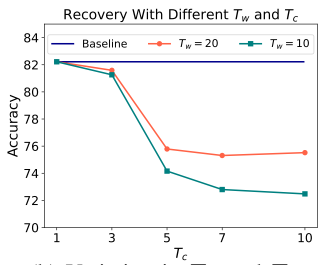
(A3) FedRecover + Trim attack during original training and recovery: This is a more real-world scenario, e.g., when a detection mechanism that FedRecover uses during original training is not perfect. Hence, FedRecover's recovery phase now includes undetected malicious clients (i.e., false negatives) and excludes incorrectly detected benign clients (i.e., false positives). Note that, *this evaluation is related to our fourth component, i.e., choice of attacks*, and that FedRecover considers such an adaptive attack, but evaluates it only for the simple and intrinsically robust MNIST task. Hence, to better understand efficacy of FedRecover, we evaluate it using FEMNIST, a more challenging and real-world FL task.

Figure 5a shows results for FEMNIST when we vary the FNR and FPR for FEMNIST with $T_w = 20$ warmup rounds, periodic correction at every $T_c = 10$ round, and we fine-tune the global model for the last $T_f = 5$ rounds. We use 300 of total FEMNIST clients and 20% (60) of them are malicious. We train FEMNIST using FedAvg for 200 rounds; please check Appendix VII-B1 for complete setup. The results represent the difference in accuracy of FedRecover and the best achievable accuracy using only the benign clients. Note that, during recovery, malicious clients (due to non-zero

[2]Majority of evaluations in FedRecover work make this assumption; only Figure 8 of [14] assumes presence of malicious clients during recovery.



(a) Variation in FNR and FPR



(b) Variation in T_w and T_c

Figure 5: Ablation study of FedRecover for FEMNIST: **(a)** FedRecover performance with an imperfect detection during original training, e.g., with non-zero FNR and FPR, **(b)** FedRecover performance with different periodic correction time T_c, different warmup rounds T_w, and *no attack during recovery*. Please refer to Section IV-B3 for more details.

FNR) send malicious Trim attack updates whenever the server requests exact updates from them. Such malicious exact updates effectively lower the FedRecover accuracy. FedRecover performs the worst with FNR = FPR = 0.5, i.e., when detector fails to detect 30 malicious clients while incorrectly detects 120 benign clients as malicious. Compared to the best accuracy (82%) achievable with only benign clients, FedRecover accuracy is 23% less (i.e., 59%), while the accuracy of the poisoned model is 25% less (i.e., 57%). In short, FedRecover gains just 2% in accuracy. Not only at high FPR/FNR, but even with low FPR and FNR of 0.1 each, FedRecover recovers poorly with 8% lower (i.e., 74%) than the best possible accuracy.

Ablation study: We show in Figure 5b that FedRecover's performance is very sensitive to the choice of its recovery parameters such as number of warmup rounds, T_w, and periodic correction period, T_c. Note that, here we do not mount any attack during recovery. Higher T_w lowers the estimation errors in the later rounds, but incurs high computation/communication cost. Hence, decreasing T_w increases lowers the FedRecover accuracy. Note that due to fast FedAvg algorithm, we achieve 73% accuracy in $T_w = 20$, hence in the rest of 180 rounds of recovery, FedRecover gains only 3% in accuracy with $T_c \in \{5, 7, 10\}$.

Similar to warmup rounds, periodic correction reduces estimation errors by periodically collecting exact updates from all clients every T_c rounds. Smaller T_c reduces estimation errors and increases FedRecover accuracy, but increases communication. We note that FedRecover performs poorly for $T_c > 3$. For $T_c = 3$ FedRecover performs very well, however note that at $T_c \leq 3$, total number of rounds in which FedRecover collects exact updates is $T_w + \frac{T_{total} - T_w - T_f}{T_c} + T_f$, which is 85 in our FEMNIST case. Now note in Figure 2 that fast FedAvg algorithm with 80 benign clients achieves well over 81% accuracy in 85 rounds. Hence, we don't really need to use FedRecover if T_c is too low in this particular setting.

3) Choice of data distribution: Dirichlet (Dir) distribution produces more non-IID datasets compared to FCJ (Section III-**(3)**). Below we compare FedRecover's performance with the two distributions for FedSGD setting with 500 FL rounds, batch size 32 and learning rate 0.1; we present three accuracies (as in original work): train-from-scratch (TFSA), post-attack (PAA), and FedRecover's post-recovery (PRA) accuracies. Note that FedRecover claims to recover to TFSA from PAA.

We do not observe significant differences between the two distributions, but contrary to the conclusion of the original work [14], in general FedRecover works well for low degree of non-IID but performs poorly at higher non-IID degrees. For example, for low non-IID FCJ-0.3 (for FCJ (Dir) higher (lower) param values give more non-IID datasets), (TFSA, PAA, PRA) are (85.23%, 78.68%, 78.25%), while for high non-IID FCJ-0.7 they are (85.17%, 61.05%, 72.56%). For the extreme non-IID case of FCJ-1.0, they are (84.6%, 35.83%, 46.9%). That is for FCJ with 0.3, 0.7, 1.0 parameter values, FedRecover's PRA is respectively 7%, 12.61% and 37.7% lower than TFSA. Similarly for lower non-IID case Dir-0.7, the accuracies are (84.66%, 77.3%, 76.46%) while for more non-IID case Dir-0.3, they are (85%, 73.71%, 71.69%), i.e., for Dir with 0.7 and 0.3 parameter values, FedRecover' recovered accuracy is respectively 8.2% and 13.31% lower than TFSA. This is because FedRecover uses past global models to estimate clients updates, and with highly non-IID FL, the global model is not aligned with client updates very well. Hence FedRecover computes poor estimates of client updates which lower its performance.

V. Key Recommendations

Below we provide four recommendations to future work on FL poisoning, based on our qualitative (Section III) and quantitative (Section IV) analysis of prior works.

Recommendation-1: Literature on defenses against FL poisoning should use state-of-the-art FL algorithms to motivate and to evaluate the proposed defenses.

Recommendation-2: Defense evaluations should use FL tasks with varying difficulties for their robustness evaluations, as using simple and intrinsically robust tasks can lead to false claims on security.

Recommendation-3: Robustness evaluations should preferably use real-world FL datasets for evaluations, or at least synthetic datasets that represent the characteristics of real-world settings, e.g., high heterogeneity.

Recommendation-4: Robustness evaluations should consider strong state-of-the-art attacks under various (practical) adversarial settings, including adaptive attacks.

VI. Conclusion

In this work, we looked at an often neglected aspect of the literature on defenses against FL poisoning—*experimental setup they use* to measure defenses' performance. We review 50 defense works and highlight the questionable trends in setting up their experiments. Furthermore, using Trimmed-mean, a popular defense, we empirically demonstrated how these trends can misrepresent robustness. Finally, we performed a thorough re-evaluation of a representative recent FL poisoning defense, FedRecover, showing how the choice of experimental setup decisions can influence their robustness claims.

Acknowledgements

This work was supported by the NSF grant 2131910.

References

[1] "Federated learning: Collaborative machine learning without centralized training data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[2] "The stack overflow data," https://www.kaggle.com/datasets/stackoverflow/stackoverflow, 2019.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[4] Z. Allen-Zhu, F. Ebrahimian, J. Li, and D. Alistarh, "Byzantine-resilient non-convex stochastic gradient descent," *arXiv preprint arXiv:2012.14368*, 2020.

[5] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "Baffle: Backdoor detection via feedback-based federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 852–863.

[6] M. Andreux, J. O. du Terrail, C. Beguier, and E. W. Tramel, "Siloed federated learning for multi-centric histopathology datasets," in *Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning: Second MICCAI Workshop, DART 2020, and First MICCAI Workshop, DCL 2020, Held in Conjunction with MICCAI 2020, Lima, Peru, October 4–8, 2020, Proceedings 2*. Springer, 2020, pp. 129–139.

[7] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*. Springer, 2021, pp. 455–475.

[8] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[9] M. Baruch, B. Gilad, and Y. Goldberg, "A Little Is Enough: Circumventing Defenses For Distributed Learning," in *NeurIPS*, 2019.

[10] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *ICML*, 2019.

[11] P. Blanchard, R. Guerraoui, J. Stainer *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *NeurIPS*, 2017.

[12] L. Burkhalter, H. Lycklama, A. Viand, N. Küchler, and A. Hithnawi, "Rofl: Attestable robustness for secure federated learning," *arXiv preprint arXiv:2107.03311*, 2021.

[13] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *arXiv:1812.01097*, 2018.

[14] X. Cao, J. Jia, Z. Zhang, and N. Gong, "Fedrecover: Recovering from poisoning attacks in federated learning using historical information," in *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2023, pp. 326–343. [Online]. Available: https://arxiv.org/pdf/2210.10936.pdf

[15] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping," in *NDSS*, 2021.

[16] X. Cao, J. Jia, and N. Z. Gong, "Provably Secure Federated Learning against Malicious Clients," in *AAAI*, 2021.

[17] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, "Cronus: Robust and Heterogeneous Collaborative Learning with Black-Box Knowledge Transfer," *arXiv:1912.11279*, 2019.

[18] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *IJCNN*, 2017.

[19] J. O. Du Terrail, S.-S. Ayed, E. Cyffers, F. Grimberg, C. He, R. Loeb, P. Mangold, T. Marchand, O. Marfoq, E. Mushtaq *et al.*, "Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings," in *NeurIPS, Datasets and Benchmarks Track*, 2022.

[20] E. M. El Mhamdi, R. Guerraoui, and S. L. A. Rouault, "Distributed momentum for byzantine-resilient stochastic gradient descent," in *9th International Conference on Learning Representations (ICLR)*, no. CONF, 2021.

[21] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning," in *USENIX*, 2020.

[22] S. Fu, C. Xie, B. Li, and Q. Chen, "Attack-resistant federated learning with residual-based reweighting," *arXiv:1912.11464*, 2019.

[23] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *RAID*, 2020.

[24] E. Gorbunov, S. Horváth, P. Richtárik, and G. Gidel, "Variance reduction is an antidote to byzantines: Better rates, weaker assumptions and communication compression as a cherry on the top," *arXiv preprint arXiv:2206.00529*, 2022.

[25] H. Guo, H. Wang, T. Song, Y. Hua, Z. Lv, X. Jin, Z. Xue, R. Ma, and H. Guan, "Siren: Byzantine-robust federated learning via proactive alarming," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 47–60.

[26] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," *Advances in Neural Information Processing Systems, Best Paper Award at Federate Learning Workshop*, 2020.

[27] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," *arXiv preprint arXiv:2207.01982*, 2022.

[28] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *arXiv:1912.04977*, 2019.

[29] S. P. Karimireddy, L. He, and M. Jaggi, "Byzantine-robust learning on heterogeneous datasets via bucketing," *arXiv preprint arXiv:2006.09365*, 2020.

[30] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[31] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *NIPS Workshop on Private Multi-Party ML*, 2016.

[32] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[33] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[34] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *AAAI*, 2019.

[35] S. Li, Y. Cheng, Y. Liu, W. Wang, and T. Chen, "Abnormal client behavior detection in federated learning," *arXiv preprint arXiv:1910.09933*, 2019.

[36] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[37] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *ICML*, 2021.

[38] Z. Li, L. Liu, J. Zhang, and J. Liu, "Byzantine-robust federated learning through spatial-temporal analysis of local model updates," in *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2021, pp. 372–379.

[39] F. Lin, W. Li, and Q. Ling, "Stochastic alternating direction method of multipliers for byzantine-robust distributed learning," *arXiv preprint arXiv:2106.06891*, 2021.

[40] Y. Liu, R. Zhao, J. Kang, A. Yassine, D. Niyato, and J. Peng, "Towards communication-efficient and attack-resistant federated edge learning for industrial internet of things," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 3, pp. 1–22, 2021.

[41] X. Ma, Q. Jiang, M. Shojafar, M. Alazab, S. Kumar, and S. Kumari, "Disbezant: secure and robust federated learning against byzantine attack in iot-enabled mts," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[42] R. A. Mallah, D. Lopez, G. B. Marfo, and B. Farooq, "Untargeted poisoning attack detection in federated learning via behavior attestation," *arXiv preprint arXiv:2101.10904*, 2021.

[43] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[44] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault, "The Hidden Vulnerability of Distributed Learning in Byzantium," in *ICML*, 2018.

[45] H. Mozaffari, V. Shejwalkar, and A. Houmansadr, “Frl: Federated rank learning,” *arXiv preprint arXiv:2110.04350*, 2021.

[46] ——, “Every vote counts: Ranking-based training of federated learning to resist poisoning attacks,” in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[47] L. Nagalapatti and R. Narayanam, “Game of gradients: Mitigating irrelevant clients in federated learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9046–9054.

[48] M. Naseri, J. Hayes, and E. De Cristofaro, “Local and central differential privacy for robustness and privacy in federated learning,” *arXiv preprint arXiv:2009.03561*, 2020.

[49] J. Nocedal, “Updating quasi-newton matrices with limited storage,” *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[50] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, “Defending against backdoors in federated learning with robust learning rate,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9268–9276.

[51] X. Pan, M. Zhang, D. Wu, Q. Xiao, S. Ji, and M. Yang, “Justinian’s gaavernor: Robust distributed learning with gradient aggregation agent,” in *Proceedings of the 29th USENIX Conference on Security Symposium*, 2020, pp. 1641–1658.

[52] J. Park, D.-J. Han, M. Choi, and J. Moon, “Sageflow: Robust federated learning against both stragglers and adversaries,” *Advances in neural information processing systems*, vol. 34, pp. 840–851, 2021.

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

[54] M. Paulik, M. Seigel, H. Mason *et al.*, “Federated Evaluation and Tuning for On-Device Personalization: System Design & Applications,” *arXiv:2102.08503*, 2021.

[55] K. Pillutla, S. M. Kakade, and Z. Harchaoui, “Robust aggregation for federated learning,” *arXiv:1912.13445*, 2019.

[56] S. Praneeth Karimireddy, L. He, and M. Jaggi, “Learning from History for Byzantine Robust Optimization,” *arXiv e-prints*, pp. arXiv–2012, 2020.

[57] P. Ranjan, A. Gupta, F. Coro, and S. K. Das, “Securing federated learning against overwhelming collusive attackers,” in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 1448–1453.

[58] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan, “Adaptive Federated Optimization,” in *ICLR*, 2020.

[59] A. Roy Chowdhury, C. Guo, S. Jha, and L. van der Maaten, “Eiffel: Ensuring integrity for federated learning,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2535–2549.

[60] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, “On the byzantine robustness of clustered federated learning,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8861–8865.

[61] G. Severi, M. Jagielski, G. Yar, Y. Wang, A. Oprea, and C. Nita-Rotaru, “Network-level adversaries in federated learning,” *arXiv preprint arXiv:2208.12911*, 2022.

[62] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, “Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning,” in *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2022, pp. 1117–1134. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00065

[63] V. Shejwalkar and A. Houmansadr, “Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning,” in *NDSS*, 2021.

[64] S. Shen, S. Tople, and P. Saxena, “AUROR: Defending againsts poisoning attacks in collaborative deep learning systems,” *2016 Annual Computer Security Applications Conference*, 2016.

[65] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, “Can you really backdoor federated learning?” *NeurIPS FL Workshop*, 2019.

[66] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, “Attack of the tails: Yes, you really can backdoor federated learning,” in *NeurIPS*, 2020.

[67] N. Wang, Y. Xiao, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, “Flare: defending federated learning against model poisoning attacks via latent space representations,” in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 946–958.

[68] “Utilization of FATE in Risk Management of Credit in Small and Micro Enterprises,” https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/, 2019.

[69] C. Wu, X. Yang, S. Zhu, and P. Mitra, “Mitigating backdoor attacks in federated learning,” *arXiv:2011.01767*, 2020.

[70] S. Wu, T. Li, Z. Charles, Y. Xiao, Z. Liu, Z. Xu, and V. Smith, “Motley: Benchmarking heterogeneity and personalization in federated learning,” 2022.

[71] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, “Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

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

[73] C. Xie, M. Chen, P.-Y. Chen, and B. Li, “CRFL: Certifiably Robust Federated Learning against Backdoor Attacks,” in *ICML*, 2021.

[74] C. Xie, O. Koyejo, and I. Gupta, “Generalized byzantine-tolerant sgd,” *arXiv:1802.10116*, 2018.

[75] C. Xie, S. Koyejo, and I. Gupta, “Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation,” *arXiv:1903.03936*, 2019.

[76] Y. Xie, W. Zhang, R. Pi, F. Wu, Q. Chen, X. Xie, and S. Kim, “Optimizing server-side aggregation for robust federated learning via subspace training,” *arXiv preprint arXiv:2211.05554*, 2022.

[77] C. Xu, Y. Jia, L. Zhu, C. Zhang, G. Jin, and K. Sharif, “Tdfl: Truth discovery based byzantine robust federated learning,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4835–4848, 2022.

[78] J. Xu, S.-L. Huang, L. Song, and T. Lan, “Signguard: Byzantine-robust federated learning through collaborative malicious gradient filtering,” *arXiv preprint arXiv:2109.05872*, 2021.

[79] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *ICML*, 2018.

[80] S. Zawad, A. Ali, P.-Y. Chen, A. Anwar, Y. Zhou, N. Baracaldo, Y. Tian, and F. Yan, “Curse or redemption? how data heterogeneity affects the robustness of federated learning,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 10 807–10 814.

[81] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma *et al.*, “Flip: A provable defense framework for backdoor mitigation in federated learning,” *arXiv preprint arXiv:2210.12873*, 2022.

[82] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, “Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients,” in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2545–2555.

VII. Appendix

A. Methodology we use to classify 50 defense works

We classify 50 works across the following four dimensions:

- *FL algorithm*: We classify the papers based on one of the two types of popular FL Algorithms; FedAvg or FedSGD. Some papers explicitly mention the algorithm/s they use in setup details. However, for some we need to carefully check other parts of the paper to decide the algorithm, e.g., TDFL [77] work does not mention which FL algorithm they use, but their Algorithms 1 and 2 describing FL used FedAvg, hence we belive they use FedAvg in their evaluations.
- *Datasets*: For each paper, we list the datasets it uses for evaluating the performance of its defense proposed;

these are generally clearly mentioned in setup. The most common datasets we found are MNIST, CIFAR10, Fashion MNIST, FEMNIST, CIFAR100, and Purchase. But in the union of datasets used in the 50 works, there are many more and they are listed in Table I. For presentation clarity, Table I uses acronyms for the datasets and attacks, which are listed in Tables II & III.

- *Data distribution*: We classify a defense based on the data distribution strategies it uses. The most common strategies we found are IID, Natural, FCJ, Dirichlet, McMahan, and Exponential.
- *Attacks*: For each paper, we list the attack/s it uses to evaluate its defense, and then we report frequency of attacks using histogram; most papers clearly mentioned the attacks they use or the plots in the paper clarified the attacks used. The popular attacks we found are Label flip, adding random Gaussian noise, Little-is-enough, Backdoor, Sign Flip, Trim, and Inner Product Manipulation (IMP).

Note that it is possible for a paper to belong to multiple categories in each dimension. For instance, This means that the frequencies in Figure 1 can add up to a number greater than 50. For example, if we add up the frequencies in the attacks graph we get the number 79. We also note that, some papers [39], [42], [77] could not be classified for some categories such as choice of FedAvg or FedSGD. Table I lists all the papers we classified using above classification method.

B. Experimental setup

Due to space constraints, we provide detailed experimental setup here.

1) FEMNIST [13], [18]: FEMNIST is a character recognition classification task with 3,400 clients, 62 classes (52 for upper and lower case letters and 10 for digits), and 671,585 grayscale images. Each client has data of their own handwritten digits or letters. We use 300 randomly selected clients with their original data in a cross-silo fashion as FedRecover uses the cross-silo setting in its implementation. We use the CNN used by [14] and use the Xavier weight initialization.

Hyperparameters for re-eval: For FEMNIST, we run over 200 epochs with 300 clients. In the attack setting, 60 clients are malicious. The results in Figure 4 use $T_w = 10$, $T_c = 10$, and $T_f = 5$. The FL algorithm used here is FedAVG with a local learning rate of 0.05 and a global learning rate of 1. We keep the batch size to be 32. The number of local epochs is kept 1. For Figure 5a we vary consider the possibility of benign clients being misclassified as malicious, or malicious clients being misclassified as benign so we vary the false negative and false positive rates between 0.1 and 0.5. For Figure 5b we vary T_c between 1 and 10, where $T_c = 1$ means all updates are exact updates and use 10 and 20 for T_w.

2) CIFAR10 [32]: CIFAR10 is a 10-class classification task with 60,000 total RGB images, each of size 32 × 32. We divide all the data among 100 clients using either Dirichlet [58] or *FCJ* [21] distributions, which are the two most popular synthetic strategies to generate FL dataset. We use a Resnet20 model with the CIFAR dataset.

Hyperparameters for re-eval: We run over 100 epochs with 10 0 clients. In the attack setting, 20 clients are malicious. The FL algorithm used here is FedAVG with a local learning rate of 0.01 and a global learning rate of 1. We keep the batch size to be 16. The number of local epochs is kept 2. The results in Figure 4 use $T_w = 10$, $T_c = 5$, and $T_f = 5$, and the fang distribution. Contrary to the rest of the datasets used, we use $T_c = 5$, because CIFAR10 was a much more challenging learning task.

3) MNIST [33]: MNIST is a 10-class digit image classification dataset, which contains 70,000 grayscale images of size 28 × 28. We consider 100 FL clients and divide all data using Dirichlet or FCJ distributions. We use the same CNN as FEMNIST dataset.

Hyperparameters for re-eval: For MNIST, we run over 2000 epochs with 100 clients, a learning rate of 0.03, and a batch size of 32. In the attack setting, 20 clients are malicious. We set $T_w = 20$, $T_c = 10$, and $T_f = 10$. The FL algorithm used here is FedSGD. The results reported in Figure 4 use the fang distribution.

4) Fashion-MNIST [72]: Fashion-MNIST is a 10-class image classification dataset with grayscale images of clothing of size 28 × 28. It contains 70,000 total images. We consider 100 FL clients and divide all 70,000 images using Dirichlet or FCJ distributions. For CIFAR10, MNIST and FashionMNIST, we divide each client's data in train/test/validation splits in the ratio of $10 : 1 : 1$. We combine clients' validation data and use it for validation and hyperparameter tuning, and report accuracy on test data. We use the same CNN as FEMNIST dataset.

Hyperparameters for re-eval: We run over 2000 epochs with 100 clients, a learning rate of 3×10^{-3} [3], and a batch size of 32. In the attack setting, 20 clients are malicious. We set $T_w = 20$, $T_c = 10$, and $T_f = 10$. The FL algorithm used here is FedSGD. The results reported in Figure 4 use the fang distribution.

C. Untargeted poisoning attacks used for evaluation

Trim Attack: We describe the Trim attack and adopt the description given in [14]: "Fang et al. [21] formulated untargeted poisoning attacks to FL as a general framework. Roughly speaking, the framework aims to craft malicious model updates that maximize the difference between the aggregated model updates before and after attack. The framework can be applied to different aggregation rules. The Trim attack is constructed based on the Trimmed-mean aggregation rule under the framework, and is also effective for other aggregation rules such as FedAvg and Median."

NDSS Attack: We describe the NDSS attack and adopt the description given in [63]: "The adversary computes a benign reference aggregate using some benign data samples she knows; then she computes a malicious perturbation vector,

[3]We could not achieve the same accuracy reported in [14] using their reported 3×10^{-4} learning rate, hence we use 3×10^{-3}.

Table I: Classification of 50 defense works across 4 dimensions of evaluation setup; please check Section VII-A for details.

Work	Datasets	Attacks	Data Distribution	FL algorith
FLDetector [82]	FA,FE,C10	Trim	Fang, Natural	FedSGD
FedRecover [14]	M,FA,P,H	Trim	Fang	FedSGD
Machine Learning with Adversaries [11]	M, spambase	RGA	IID	FedSGD
FLTrust [15]	M,CHM,C10,H	Krum, Trim, LF	Fang	FedAvg
Byzantine-Robust Distributed Learning [79]	M	RGA	IID	FedSGD
Provably Secure Federated Learning against Malicious Clients [16]	M	Not applicable	Fang	FedAvg
Learning to Detect Malicious Clients for Robust FL [36]	M, FE, S140	SF, AN, BD	Natural, McMahan	FedAvg
Robust Federated Learning [76]	M,FE,C10/100,N20	LF, LIE, Fang	Dirichlet, Natural	FedAvg
The Hidden Vulnerability of Distributed Learning in Byzantium [44]	M, C10	Specific attack	IID	FedSGD
Sageflow [52]	M, FA, C10	SF, LF	McMahan	FedAvg
Mitigating Irrelevant Clients in FL [47]	M	LF	McMahan	FedAvg
Cronus [17]	M, C10, P, Svhn	LF, LIE	IID	FedAvg
Can You Really Backdoor Federated Learning? [65]	FE	BD	Natural	FedAvg
Generalized Byzantine-tolerant SGD [74]	M, C10	BF, LF, LIE	IID	FedSGD
The Limitations of Federated Learning in Sybil Settings [23]	M, VGG, KDD, A	LF, BD	Each class to a client	Both
Ditto [37]	FA, FE, CelebA	LF, RGA, BD	Natural, McMahan	FedAvg
Auror [64]	M	Targeted-LF	IID	FedSGD
Robust Aggregation for Federated Learning [55]	FE,S140,S	Specific attacks, RGA	Natural	FedAvg
CRFL [73]	M, FE	BD	IID	FedAvg
FLIP [81]	M, FA, C10	BD	Dirichlet	FedAvg
RoFL [12]	FE, C10	BD	Natural, Dirichlet	FedAvg
Securing FL against Overwhelming Collusive Attackers [57]	M, FA	LF, BD	Dirichlet	FedAvg
Defending against the Label-flipping Attack in FL [27]	M, C10	LF,	IID, Dirichlet	FedAvg
FRL [46]	M, FE, C10	Fang, NDSS21	Dirichlet, Natural	FedAvg
CONTRA [7]	M, C10, Loan	LF, BD	Dirichlet	FedAvg
EIFFeL [59]	M, FA, FE,C10	LIE, RGA, SF, NDSS21	IID, Natural	FedAvg
Local and central DP for robustness and privacy in FL [48]	E, C10, s140, Reddit	BD	McMahan	FedAvg
Signguard [78]	M, FA, C10, AGnews	LIE, RGA, SF, NDSS21	IID	FedSGD
DisBezant [41]	M, FA, C10	RGA	Fang	FedAvg
Learning from History for Byzantine Robust Optimization [56]	M, C10	BF, LF, LIE	Exponential	FedSGD
Byzantine-robust learning on heterogeneous datasets... [29]	M	BF, LF, LIE, IPM, Mimic	McMahan	FedSGD
Byzantine-Resilient Non-Convex Stochastic Gradient Descent [4]	C10, C100	SF, LF, LIE, Delayed-grad,	IID	FedSGD
Byzantine-robust Federated Learning... [38]	M, FA, C10, Spambase	LF, IPM, LIE, Uniform, arbitrary	McMahan	FedAvg
Stochastic alternating direction method of multipliers for... [39]	M, Covertype	RGA, SF, LF	IID	FedSGD
Variance reduction is an antidote to byzantines [24]	LIBSVM	LF, BF, LIE, IPM	IID	FedSGD
On the byzantine robustness of clustered FL [60]	M, FA, C10	RGA, LF, Uniform noise	IID	FedSGD
RSA [34]	M	SF	IID	FedSGD
Federated variance-reduced stochastic gradient descent [71]	ijcnn1, covtype	RGA, SF, Zero-grad	IID	FedSGD
Abnormal client behavior detection in federated learning [35]	FE	SF, RGA, Grad ascent	Natural	FedAvg
Distributed Momentum for Byzantine-resilient SGD [20]	M, FA, C10/100	LIE, IPM	IID	FedSGD
Attack-resistant FL with residual-based reweighting [22]	M, C10, Amazon, Loan	LF, BD	Dirichlet, Natural	FedAvg
Towards communication-efficient and attack-resistant... [40]	M	LF	IID	FedSGD
Justinian's GAAvernor [51]	M, C10, Yelp, Health	RGA	IID	FedSGD
Untargeted poisoning attack detection in FL via... [42]	M, C10, MTL Trajet	BD	IID	FedSGD
TDFL [77]	M, FA, C10	LF, RGA, Krum, Trim, BD	McMahan	FedAvg
Siren [25]	FA, C10	SF, LF, bhagoji	Fang	FedAvg
FLARE [67]	FA, C10, Kather	Krum, Trim	IID	FedAvg
Analyzing Federated Learning Through an Adversarial Lens [10]	FA, UCI Census	Specific attack	IID	FedAvg
BaFFLe [5]	C10, FE	BD	Dirichlet	FedAvg
Defending against backdoors in FL with robust learning rate [50]	FA, FE	BD	Both	FedAvg

e.g., a unit vector in the opposite direction of the benign aggregate. Finally, the adversary computes her malicious model update by maximally perturbing the benign reference aggregate in the malicious direction with the goal of evading detection by robust aggregation algorithms."

D. Explanation of the choice of datasets used FedRecover re-evaluation

Below, we justify the choice of four datasets (Section VII-B) we use to re-evaluate FedRecover (FedRecover). There are two primary reasons for the choice we make.

(Reason-1) We believe, FedRecover is not compatible with cross-device FL setting, as it requires historical information of client's past model updates to estimate next round's update. Note that, in cross-device FL, a client participates in very few rounds. Consider a cross-device FL with 1000 total clients and 10 clients participating in each round, and that it runs for 1000 rounds. In this case, for a client, on average, the server will have just 10 updates, 1 every 100^{th} round. Since FedRecover uses LBFGS to estimate the next update using the client's past updates and global models, in the cross-device setting FedRecover would not be able to estimate a good update if a client gets selected for training once every 100^{th} round.

Hence, we do not consider cross-device datasets, e.g., from LEAF repository [13].

Next, note that cross-silo datasets [19], [70], [6] generally have a very small number of clients, e.g., for all the 9 cross-silo medical datasets from [19], [6] have less than 5 clients. Assuming 20% of them to be malicious (as most works assume) is not practical assumption [62], and sometimes makes evaluation impossible, e.g., when number of clients is less than 5. Hence, we omit using such cross-silo datasets from our evaluation.

(Reason-2) We use MNIST and Fashion-MNIST as they are used in FedRecover work. This serves two-fold purpose: (i) it helps us check our implementation by matching our results with the results reported in original FedRecover work; note that FedRecover work has not open-sourced their code, (ii) as *majority* of FedRecover evaluation uses MNIST data, using Fashion-MNIST allows us to understand the performance of FedRecover for slightly more difficult tasks[4]. Finally, to further stress-test FedRecover, we use even more difficult datasets, i.e., FEMNIST[5] and CIFAR10. As our results show, for more difficult datasets, either FedRecover does not recover or requires very high communication cost, and this is true even when there is no attack during original training or recovery.

E. Comparing different data distribution strategies

It is difficult to know which distribution represents real-world FL settings. However, we argue that some simple statistical analyses of distribution strategies can help us understand which strategies can better represent real-world FL. To this end we analyze Dirichlet and FCJ distributions.

In particular, consider a classification task with C classes; we generate client datasets using Dir and FCJ for 100 clients and varying levels of non-iid-ness. For clarity of presentation we use CIFAR10 data here, but for any datasets we expect similar plots. We then plot the following three statistics of the client datasets.

(Stat-1) We plot the number of samples per user, which is motivated from the client data visualizations provided in the Leaf repository for various real-world datasets.[6] Figure 6 shows the results for three levels of non-iid-ness (For Dir we use $\alpha \in \{0.1, 0.3, 0.5\}$ and for FCJ we use bias $b \in \{0.9, 0.5, 0.1\}$.[7]). We note that Dir produces client datasets with sufficient heterogeneity in terms of size of the client's local datasets. However, we observe that FCJ produces client datasets of almost equal size; note that the the black histogram is concentrated around 500 (i.e., total number of samples in dataset / total number of clients).

[4]Finally, we note that FedRecover uses HAR dataset in their evaluation which has 30 real users' data. However we omit HAR, as similar to MNIST, it is also a very simple task; this is evident from the Figure-1 of FedRecover where they report 98% accuracy with simple fully-connected network.

[5]We use 300 clients from total 3400 clients and train in cross-silo setting, because FedRecover is not compatible with cross-silo settings. This is a common practice to evaluate cross-silo defenses [82], [37]

[6]https://leaf.cmu.edu

[7]Recall that with *higher alpha, Dir* produces *more iid* datasets. While with *higher bias, FCJ* produces *more non-iid* datasets.

To summarize, FCJ produces client datasets that are locally IID and all clients have almost same sizes of datasets. This IID-ness is fundamental to FCJ distribution. As we already showed, with more IID client datasets FL becomes intrinsically robust. Hence, performing robustness analysis of robust AGRs, in combination with FCJ, may give us results that show that the AGR is highly robust.

F. Communication-accuracy trade-off of FedRecover

In Figure 7, we report the tradeoff between the recovery accuracy and communication of FedRecover; the setting is that of Figure 5b for FEMNIST when there is *no attack during recovery* phase. Note that the *minimum number of exact updates* that the FedRecover uses is $T_w + \frac{T_{total} - T_w - T_f}{T_c} + T_f$; this formula does not account for additional exact updates that FedRecover uses in its abnormality fixing phase.

We note that recovery accuracy increases as FedRecover relies on exact updates from the clients. However, note also that using the same amount of exact updates (i.e., same communication) would give us more accuracy using much simpler train-from-scratch setting. For instance, with about 20% exact updates (i.e., 40 rounds of training for train-from-scratch), FedRecover achieves about 76% accuracy while train-from-scratch achieves close to 80% accuracy. This implies, in our setting (and in any more useful fast converging FL algorithms) FedRecover cannot catch up to train-from-scratch.

G. Validating our FedRecover implementation

We were able to match our results for FedRecover for MNIST and FashionMNIST with those reported in [14]. We implement FedRecover with FedSGD, where a client shares its gradient on a single mini-batch of its data. As shown in Figure 4b, we achieve for MNIST(FMNIST) a recovery accuracy of 91%(73%) where the post-attack accuracy was 76%(65%) and the baseline no attack accuracy with all benign clients was 92%(75%).

H. Adaptive Attack FMNIST

We perform FedRecover on FMNIST under imperfect detection. From Figure 8, we can see that there is a slight drop in the performance of FedRecover when some malicious clients slip past the detector stage, i.e., non-zero FNR. These malicious clients then perform attacks during the recovery process and this leads to a lower accuracy in recovery. The variation in FPR, i.e., when some benign clients are flagged as malicious and they are unable to take part in recovery, does not seem to have a significant effect on the recovery accuracy.

Table III: Abbreviations and full-forms of attacks in Table I.

LF	Label Flip
IPM	Inner Product Manipulation
SF	Sign Flip
BF	Bit Flip
LIE	Little is Enough
AN	Additive Noise
BD	Backdoor [8]
RGA	Random gaussian attack

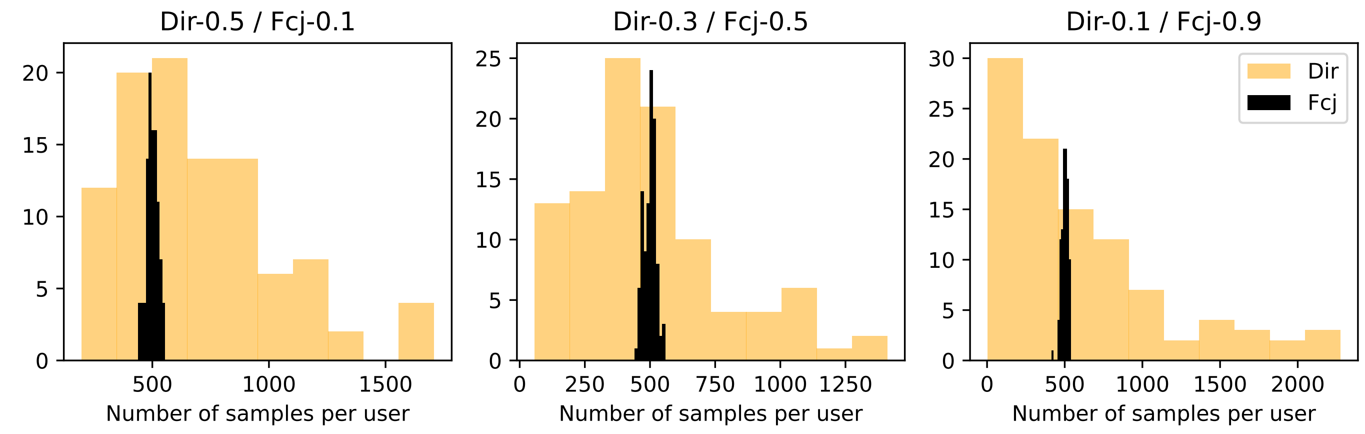


Figure 6: Histograms of the **number of samples per client** in FL client datasets generated using FCJ and Dirichlet (Dir) distributions. From left to right, the non-IID degree of generated datasets increases. We note that all of FCJ client datasets have almost same size, while Dirichlet client datasets have widely varying sizes. Recall that higher value of parameter of FCJ (Dir) distribution generates more (less) non-IID datasets.

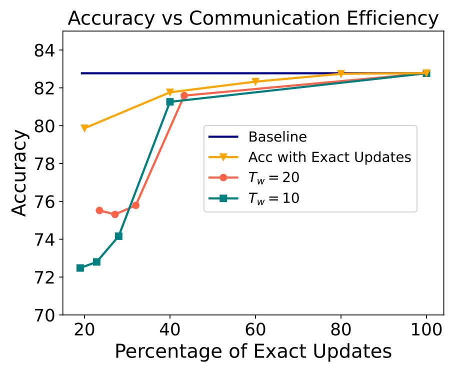


Figure 7: FedRecover accuracy increases as we rely more on exact updates, i.e., with more communication. However, with the same amount of communication, simple train-from-scratch training can achieve higher accuracy than FedRecover. Here, we do not attack during recovery phase.

Table II: Abbreviations and full-forms of datasets in Table I.

M	MNIST
FA	FashionMNIST
FE	FEMNIST
C10	CIFAR10
C100	CIFAR100
P	Purchase
H	HAR
S140	Sentiment140
SVHN	Street-view House Numbers
VGG	VGGFace
KDD	KDDCup
N20	News 20
A	Amazon
S	Shakespeare

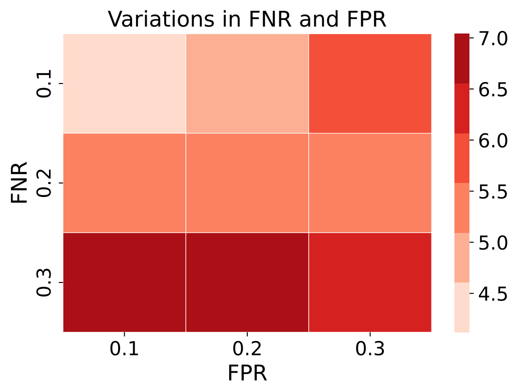


Figure 8: We evaluate FMNIST under imperfect detection i.e., some malicious clients detected as benign(non-zero FNR) and some benign clients detected as malicious(non-zero FPR).