

TrojanDam: Detection-Free Backdoor Defense in Federated Learning through Proactive Model Robustification utilizing OOD Data

Yanbo Dai[†], Songze Li[†], Zihan Gan[‡], Xueluan Gong[§]

[†]Southeast University

[‡]HKUST(GZ)

[§]Nanyang Technological University

Abstract—Federated learning (FL) systems allow decentralized data-owning clients to jointly train a global model through uploading their locally trained updates to a centralized server. The property of decentralization enables adversaries to craft carefully designed backdoor updates to make the global model misclassify only when encountering adversary-chosen triggers. Existing defense mechanisms mainly rely on post-training detection after receiving updates. These methods either fail to identify updates which are deliberately fabricated statistically close to benign ones, or show inconsistent performance in different FL training stages. The effect of unfiltered backdoor updates will accumulate in the global model, and eventually become functional. Given the difficulty of ruling out every backdoor update, we propose a backdoor defense paradigm, which focuses on proactive robustification on the global model against potential backdoor attacks. We first reveal that the successful launching of backdoor attacks in FL stems from the lack of conflict between malicious and benign updates on redundant neurons of ML models. We proceed to prove the feasibility of activating redundant neurons utilizing out-of-distribution (OOD) samples in centralized settings, and migrating to FL settings to propose a novel backdoor defense mechanism, TrojanDam. The proposed mechanism has the FL server continuously inject fresh OOD mappings into the global model to activate redundant neurons, canceling the effect of backdoor updates during aggregation. We conduct systematic and extensive experiments to illustrate the superior performance of TrojanDam, over several SOTA backdoor defense methods across a wide range of FL settings.

I. INTRODUCTION

The proliferation of personal and corporate data brought by the booming of computational resources makes proper exploitation on sensitive data a challenging problem. Federated learning (FL) [34] offers a privacy-preserving solution through enabling multiple data owners to jointly train a global model under the coordination of a central server. For every global round in FL, the server first broadcasts the global model to selected local clients. The server proceeds to aggregate received updates, which are trained by participating clients on their local datasets, to generate a global model for the next iteration. During the interaction between the server and clients, only the model updates, instead of their raw data, are exposed.

Despite respecting participants' privacy, FL frameworks are notorious for their vulnerability against various malicious behaviors [1], [53], [11], [16], [15], [38], [2]. Due to the decentralized nature of the FL, adversaries could easily compromise

participants to launch either untargeted attacks [11], [43], [7], [20] or backdoor attacks [2], [1]. Different from untargeted attackers, who aim to compromise the overall performance of the global model, backdoor adversaries are especially destructive because of their stealth. Once successfully injected, the infected model will misclassify into an adversary-chosen label when encountering predefined triggers, while leaving other tasks uninfluenced. The feasibility of injecting backdoors into FL models has been extensively demonstrated in prior work. Bagdasaryan *et al.* [1] proposes to upload a poisoned model, which is scaled up to cancel the effect of other benign updates during model aggregation, to successfully replace a backdoor model with the global model. Zheng *et al.* [53] identifies parameters which are frequently updated by benign updates, and excludes them from training the poisoned model to inject more stealthy and durable backdoors.

The threat posed by backdoor attacks necessitates designs of backdoor elimination mechanism in practical FL frameworks. Previous works on defending against backdoor attacks in FL rely on *post-processing* techniques. After collecting updates from clients, the server either tries to limit the influence of backdoor updates on the global model [47], [5], [35], [37], [52], or identify and further filter out backdoors through comparing model parameters [42], [3], [39], [14], [50], [36], [44], [54], [30], [4], [13], [27], [41]. However, it is widely believed that influence-reduction-based methods can merely slow the rate of backdoor success, rather than eliminating the injected backdoors entirely. For instance, Sun *et al.* [47] proposes to clip all received updates to an agreed bound to limit the influence of scaled backdoor updates. Other approaches apply differential privacy by injecting random noise into the aggregated model, aiming to obscure adversarial features. Detect-then-filter methods generally assume that poisoned updates differ from benign ones in the parameter space, using statistical metrics to flag anomalies. Updates exhibiting significant deviations are treated as poisoned and excluded from aggregation. A recent work, BackdoorIndicator [31], enhances post-training detection by proactively injecting pseudo-backdoors into the global model prior to client distribution. When adversaries later upload malicious updates, the pre-planted pseudo-backdoors are triggered, allowing the server to detect and remove compromised updates. While

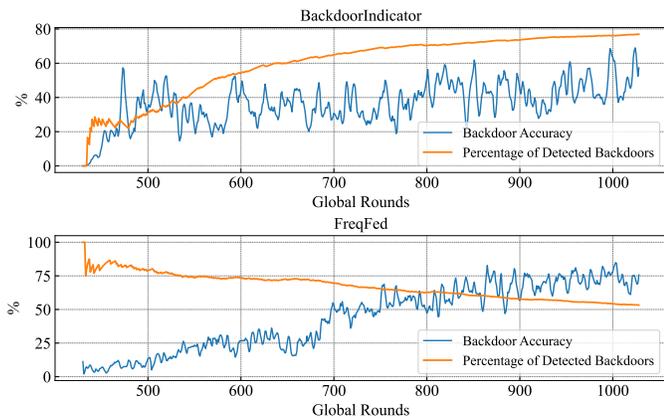


Fig. 1: The backdoor task accuracy and the percentage of detected backdoors of (UPPER) BackdoorIndicator, and (LOWER) FreqFed.

BackdoorIndicator achieves SOTA detection performance, it struggles to identify poisoned updates during the early stages of FL training. Consequently, some malicious updates are aggregated into the global model, allowing the backdoor to gradually take effect as the adversary continues participating in training. We empirically evaluate the performance of BackdoorIndicator and another SOAT backdoor detection scheme, FreqFed [13], against long-term backdoor injection. Assuming the adversary continuously attacks throughout training, our results (Figure 1) show that BackdoorIndicator fails to detect early-stage attacks, allowing backdoor accuracy to rise to 40% before detection stabilizes. Although FreqFed maintains an 80% detection rate in the first 100 rounds, unfiltered malicious updates gradually influence the global model. As the poisoned models become less distinguishable from benign ones, detection effectiveness declines over time. *These observations highlight the limitations of post-hoc detection methods and underscore the need for defense mechanisms that do not rely solely on discriminating between benign and backdoor updates. Instead, robust defenses must be capable of withstanding persistent backdoor injection across long FL training horizons.*

In this work, we propose a novel *pre-processing* backdoor defense mechanism, *TrojanDam*, to defend against backdoor injection by robustifying the FL global model *prior* to broadcasting. **Unlike existing approaches, TrojanDam does not require the server to identify potential malicious updates from clients.** Instead, we propose to have the server *proactively robustify redundant neurons*, which are the most susceptible to backdoor injections. The key observation is that effective FL backdoor attacks tend to exploit redundant neurons which are rarely updated by benign training. Fortunately, these neurons could be activated to mitigate backdoors, by training using a mixture of main task samples and OOD samples, which we refer to as flood data. TrojanDam operates by activating redundant neurons at the beginning of every global round using flood data, thereby reducing their capacity to encode adversarial triggers. Since the server typically lacks

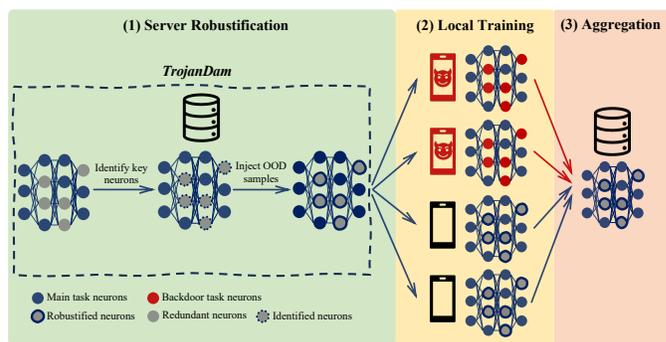


Fig. 2: The overview of FL systems with TrojanDam.

access to main task data, we introduce an additional set of OOD samples-termed shadow data-to approximate the distribution of the main task. The server computes gradients from a mixture of flood and shadow data, and projects them onto key kernels identified as influential to redundant neurons. This targeted gradient projection allows for a more efficient and lightweight update, reducing the required amount of flood data and improving deployment practicality. We demonstrate the overview of FL systems equipped with TrojanDam in Figure 2. At the beginning of every FL global round, the server activates redundant neurons by introducing OOD mappings before broadcasting. While adversaries may still attempt to inject backdoors by exploiting these neurons, their efforts are mitigated by aggregation with benign updates in which redundant neurons remain activated

We further conduct extensive experiments on three image datasets: CIFAR10, CIFAR100 [29] and EMNIST [8], with three model architectures: VGG16 [45], ResNet18 and ResNet34 [22]. We demonstrate the effectiveness of TrojanDam, by comparing its backdoor suppression performance with several SOTA defense mechanisms against a powerful adversary. The adversary could upload different types of backdoor updates trained using different training algorithms. We assume that the adversary could continuously participate in the FL paradigm for a large number of global rounds. We also provide results to reveal the influence of several key hyper-parameters, including the source of the flood dataset, flood dataset size, and the ratio of key kernels, on the performance of TrojanDam.

In summary, our contribution is four folds: 1) we propose a novel detection-free backdoor defense paradigm, which mainly relies on a proactive process on the global model to robustify the FL model. 2) We reveal that redundant neurons in neural networks could be robustified in a centralized setting through injecting OOD samples. 3) Motivated by the above observation, we migrate such an idea to FL settings, and propose a novel backdoor suppression mechanism, TrojanDam. The proposed method has the FL server consistently inject OOD mappings to robustify redundant neurons in the global model, canceling backdoors during aggregation. 4) We provide extensive empirical results to demonstrate the effectiveness of TrojanDam over several SOTA backdoor defense methods

across various adversarial and FL scenarios.

II. PRELIMINARIES AND RELATED WORK

A. Federated Learning

In an FL system, multiple data-owning clients jointly train a global DNN model under the guidance of a central server. Participating clients interact with the central server through downloading the global model, and uploading local models trained on their private data. This training paradigm reduces the infrastructure cost through offloading computing tasks to local devices, and also preserves users' privacy as no raw data is exchanged throughout the whole training process. The baseline algorithm for implementing an FL system is FedAVG [34]. Generally speaking, FedAVG aims to minimize the summation of the local empirical losses $\sum_{i=1}^S \mathcal{L}_i(\theta)$ of S participating clients. Here, we denote \mathcal{L}_i as the cross-entropy over the local dataset D_i of client i , and θ as the global model. For global round t of the FL, the server first broadcasts the current FL global model θ^t to a subset S_t of selected clients. Each local client i in S_t then initializes its local model θ_i^t from θ^t , and then trains its local model through computing updates on D_i . The server then aggregates all received local updates through $\theta^{t+1} = \frac{1}{|S_t|} \sum_{i \in S_t} \theta_i^t$ to generate the global model for global round $t+1$. In the rest of the paper, we adopt FedAVG for FL training.

B. Backdoor Attacks in FL

After corrupting local clients, the adversary could incorporate poisoned samples into the local training dataset. The adversary proceeds to train the poisoned model through computing updates on the constructed training dataset using mini-batch stochastic gradient descent. Besides the baseline backdoor injection methods, previous works have proposed various advanced backdoor training methods, which enjoy stronger stealth against backdoor defenses. To escape from the norm-clipping defense, which limits the influence of individual updates through regularizing the norm of each received update to a predefined bound, the adversary could adopt projected gradient descend (PGD) to inject the backdoor [47]. Specifically, the poisoned model is trained and then projected onto an ℓ_2 ball around the model of the previous iteration.

Another line of work tries to fabricate backdoor updates to make them statistically close to benign updates to escape from backdoor detection. F3BA [12] first identifies a small fraction of parameters with the lowest movement-based importance scores computed from the element-wise product between their weights and gradients. The identified candidate parameters are the least important to the performance of the main task. The adversary then compromises these parameters by flipping their sign to enhance their sensitivity to the trigger. AutoAdapt proposes to leverage the Augmented Lagrangian-based methods for automatically evading backdoor detection [28]. The adversary could first train several benign models to compute legitimate values for the detection metric used by the known defense. The extracted values further form a valid range, which is then transformed into the Lagrangian multipliers.

The backdoor model is then trained under the constraint of these Lagrangian multipliers to achieve a satisfactory solution. Several recent works propose to cast the backdoor attack as a joint optimization problem. While training poisoned local models, they directly optimize the backdoor trigger to make models misclassify into the adversarial target label. CerP [32] formulates the distributed backdoor attack in terms of three learning objectives, which are the fine-tuning of backdoor triggers, the control over poisoned model bias, and the diversity of poisoned local models. PFedBA [33] further improves the trigger optimization procedure through aligning gradients of the backdoor task with those of the benign task.

Adversaries could also choose to inject different types of backdoors into the global model. BadNets [19] proposes to directly modify pixels in the original image, and considers the overlaid pixel-pattern as the backdoor trigger. Targeted contamination attack (TaCT) [48] further strengthens the stealth of the injected pixel-pattern backdoor by only assigning the target label to trigger-carrying samples from the specific class. Blended backdoors [6] sample a random image or a fixed noise mask from uniform distribution as backdoor triggers, and construct backdoor images by mixing up triggers with original images. This renders the constructed poisoned images visually indistinguishable from benign ones, evading potential human inspection. While aforementioned backdoors choose manually created features as backdoor triggers, triggers for semantic backdoors [1] could be selected as any naturally occurring feature of the physical world. The adversary could also construct a special type of semantic backdoors, termed as the edge-case backdoors [49]. This kind of backdoors adopts data that lives in the tail of the input distribution as poisoned images. Such injected backdoors are less likely to conflict with benign updates. This equips injected backdoors with stronger durability against the vanishing backdoor effect [9], [53] and stealth against backdoor detection mechanisms.

C. Backdoor Defenses in FL

Most existing backdoor defense mechanisms require the central server to process received updates after clients finish local training. The server could try to limit the influence of individual updates by either clipping them to a predefined bound [47], or adding random noises to the aggregated model to interfere with potentially injected backdoors [35]. However, it is generally believed that solely applying influence-reduction-based methods can only slow down the backdoor injection rate, but disable from eliminating backdoors. Another line of work tries to have the server identify backdoor updates among benign updates by comparing model parameters. Identified anomaly updates are filtered out from aggregation for the next global round. We then elaborate on introducing several SOTA detect-then-filter-based methods in the following. Also, we select these detection methods as baseline algorithms to demonstrate the effectiveness of the proposed method in defending against long-term backdoor injections.

The server could equip the FL system with the byzantine-robust aggregation protocol to filter out backdoor updates.

The baseline algorithm, Multi-Krum [3], could ensure the convergence of the distributed training with n participants and at most f malicious clients. For every received update, the server identifies $n - f - 1$ updates which are the closest in ℓ_2 norm. The summation of the computed $n - f - 1$ distances is assigned to every update, and the one with the smallest value is selected iteratively until m updates are selected. The server then aggregates all selected updates as the global model for the next round.

A major line of detection-based methods tries to isolate backdoor updates from benign updates by computing statistical metrics on received model parameters. These methods generally build upon the assumption that introducing backdoors makes poisoned models distinguishable from benign updates in the parameter space. Deepsight [42] detects backdoor attacks by measuring differences in model updates using two metrics: Division Differences (DDifs) and Normalized Energy Updates (NEUPs). DDifs compare prediction scores between local and global models on random inputs, with deviations indicating poisoned models. NEUPs analyze parameter updates to infer label distributions. Local updates are classified as benign or poisoned based on NEUP thresholds and then clustered using NEUPs, DDifs, and cosine similarity. Clusters with a high ratio of benign models are accepted for aggregation. Foolsgold [14] focuses on securing FL in the sybil setting, where multiple clients could be controlled by a single adversary. The defense mechanism is built upon that benign updates are more diverse than poisoned ones, as they share the same training objective. Different from directly ruling out potential backdoors, Foolsgold assigns low aggregation weight to updates with large cosine similarity with others to mitigate backdoor injections. FLAME [39] first rules out suspicious updates through clustering based on cosine similarity. All accepted updates are then clipped to the median of the norm of all accepted models m . Finally, the server adds a sufficient amount of Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the aggregated model to eliminate the injected backdoors, where $\sigma = \frac{m}{\epsilon} \cdot \sqrt{2 \ln \frac{1.25}{\delta}}$ for privacy budget ϵ and δ .

However, a recent work, BackdoorIndicator, reveals the inherent weakness of methods that relies on statistically examining received model parameters [31]. That is, these detection methods fail to identify backdoor updates, which are fabricated statistically close to benign ones. This work further combines proactive processing on the global model with post-training detection to enhance the detection success rate. This method has the server first inject a pseudo-backdoor task, termed as the indicator task, into the global model utilizing OOD data before broadcasting to local clients. Once adversaries upload poisoned models, the indicator task will be triggered and help the server rule out backdoor updates from aggregation.

Despite being the SOTA backdoor detection theme, BackdoorIndicator has relatively poor performance when identifying backdoor updates in earlier training stages. The undetected backdoor updates could still be incorporated into the global model, rendering eliminating backdoors impossible. Consid-

ering the difficulty of identifying every backdoor update for a long term, we propose a novel backdoor defense mechanism, TrojanDam, which relies on robustifying the FL global model leveraging OOD data before broadcasting to clients. We then elaborate on the details in the following.

III. TROJANDAM

In this section, we illustrate the rationale and detailed methodology of the proposed method. We start by describing the threat model in terms of the goal and capability of both the adversary and the defender. We proceed to reveal the key intuition behind our method through considering why backdoors could be planted into the FL global model. Through carefully designed experiments, we find the decisive role of redundant neurons in successfully planting backdoors, and further explore ways to robustify these neurons in a centralized machine learning (ML) model leveraging OOD samples. Due to the privacy requirement of the FL paradigm, we describe challenges when migrating the method to decentralized settings and their corresponding solutions. Finally, we present the detailed methodology of TrojanDam.

A. Threat Model

Adversary’s goal and capability. The adversary tries to plant backdoors into the FL global model by corrupting participating clients. Successfully injected backdoors will make the FL model misclassify into an adversary-chosen label, termed the target label, when encountering predefined backdoor triggers, while leaving the main task accuracy uninfluenced. Once a local client is corrupted, the adversary has access to its local dataset, and gains full control of the model training and uploading process. The adversary could continuously participate in the FL training process starting from any global round for a long range of global rounds. We do not make constraints on types of injected backdoors. We assume a benign majority, where in each global round, the adversary could compromise up to 50% of all participating clients.

Defender’s goal and capability. The defender (also the FL server) aims to learn a global model on the main task $\{(x_m, y_m) | y_m \in \mathcal{Y}_m\}$, where \mathcal{Y}_m is the label space of the main task data, through iteratively broadcasting to a selected number of clients, and aggregating received updates. Meanwhile, the defender will try to protect the global model from backdoor injections. We assume that the defender has no access to data that is in the same distribution as the raw data of participating clients, which adheres to the privacy requirement of the FL framework. We further assume that the defender could collect a number of OOD samples $\{(x_o, y_{ro}) | y_{ro} \in \mathcal{Y}_{ro}\}$ from the public dataset. The collected OOD samples have distinct real label space with the main task data, which means $\mathcal{Y}_m \cap \mathcal{Y}_{ro} = \emptyset$.

B. Effective Backdoor Injection via Poisoning Redundant Neurons

We first illustrate the key intuition behind our proposed defense through exploring the reason why backdoors can be

planted into the FL global model, and how it could be utilized to launch more powerful attacks.

In centralized settings, it is generally believed that the over-parameterization of deep neural networks renders adversaries the ability to plant backdoors into the machine learning model [40], [19], [6]. Specifically, previous studies [10], [21], [18], [17] revealed that only a limited number of parameters are closely related to the main task, while pruning the rest parameters (considered to be redundant neurons) has minimal impact on the accuracy of the main task. Adversaries could thus inject backdoors on these redundant neurons by incorporating poisoned samples into the training dataset. The described over-parameterization assumption also accounts for the sparse nature of gradients in stochastic gradient descent (SGD), which further sheds light on why backdoors can be injected into the FL global model, even if adversaries control only a single participant. Relevant studies [46], [26] empirically show that the majority of the ℓ_2 norm of the aggregated benign updates is concentrated in a very small number of coordinates, while leaving most redundant parameters largely unchanged. *Backdoors planted on these redundant parameters are then aggregated into the global model without interference from other benign updates.*

Further exploiting the redundant neurons can enable adversaries to launch more powerful backdoor attacks. Neurotoxin [53] proposes to identify parameters that are most frequently updated by benign updates, and exclude these parameters when training poisoned models. This allows adversaries to focus on planting backdoors on redundant neurons. Specifically, adversaries first identify the top- $k\%$ coordinates of the benign gradients utilizing their benign dataset. Adversaries could then update the malicious model by computing gradients on the poisoned dataset. These gradients are then projected onto the bottom- $(100-k)\%$ coordinates, leaving top- $k\%$ parameters which are frequently updated by benign clients unchanged.

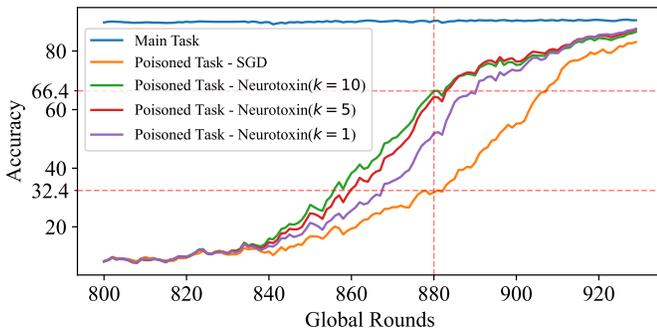


Fig. 3: Accuracies of FL global model on the main task and the poisoned tasks which are trained using SGD, and Neurotoxin with different percentages of excluded parameters (k). The adversary conducts single client attack in a continuous fashion starting from 800th global round.

We empirically compare the backdoor performance between vanilla SGD and Neurotoxin with different percentages of excluded parameters for backdoor injection. We assume that the adversary, who tries to inject pixel-pattern backdoors[19],

successively controls only one out of all selected clients in every global round starting from 800th round. As shown in Figure 3, the increase in the percentage of excluded parameters achieves higher poisoned task accuracy on the FL global model, when poisoning for the same number of global rounds. Specifically, injecting backdoors utilizing Neurotoxin when excluding 10% of coordinates achieves 66.4% poisoned task accuracy for the 880th global round, which is over twice of the poisoned task accuracy when only using SGD. This is because deliberately excluding more frequently updated neurons causes backdoor information to be planted more in redundant neurons. During FL aggregation, backdoor information on redundant neurons hardly conflicts with benign updates, resulting in high backdoor accuracy. The phenomenon further verifies the effectiveness of injecting backdoors in FL by poisoning redundant neurons. *It also indicates that the effectiveness comes from a lack of conflict between benign and backdoor updates on redundant neurons.*

Motivated by the special role of redundant neurons in planting backdoors into the FL global model, *we explore ways to defend against backdoor attacks through robustifying redundant neurons in the FL global model. Specifically, we try to activate redundant neurons by correlating them with additional information. Thus, the effects of backdoor updates on redundant neurons can be canceled during aggregation with benign updates, whose redundant neurons remain activated.* In the following section, we propose such a method that could activate redundant neurons, and correlate them with out-of-distribution (OOD) data in centralized ML models.

C. Activating Redundant Neurons Leveraging OOD Data

The idea of activating redundant neurons utilizing OOD data is motivated by the recently revealed property of backdoor tasks[31], which is **backdoor samples are essentially OOD samples concerning benign samples from the target class**. Thus, the defender could first construct a dataset with samples that are OOD concerning main task samples. The defender could then simulate the behaviors of backdoor attackers, and inject a sufficient number of OOD mappings into the model to activate redundant neurons. We first elaborate on the proposed method in a centralized setting.

In the following, we consider a defender who aims to train a centralized ML model, and tries to activate redundant neurons utilizing OOD data. Specifically, we assume the defender is training ResNet18 on CIFAR10. In addition to main task samples (x_m, y_m) from CIFAR10, the defender has extra access to a limited number of OOD samples (x_o, y_o) , which we assume are sampled from CIFAR100 wlog. To activate redundant neurons, the defender then trains the model through computing updates on the dataset $D_1 = (\{(x_m^i, y_m^i)\}_{i=1}^N, \{(x_o^j, y_o^j)\}_{j=1}^M)$, which is constructed by mixing up N main task samples and M OOD samples. Corresponding labels for OOD samples are randomly assigned, and updated every a fixed number of training iterations. This could help to ensure a continuously sufficient number of OOD mappings for the ML model to learn over iterations. For comparison, we also consider two

additional scenarios where the defender constructs the training dataset using only main task data, or only OOD data, which are $D_2 = \{(x_m^i, y_m^i)\}_{i=1}^{N+M}$ and $D_3 = \{(x_o^i, y_o^i)\}_{i=1}^{N+M}$ respectively.

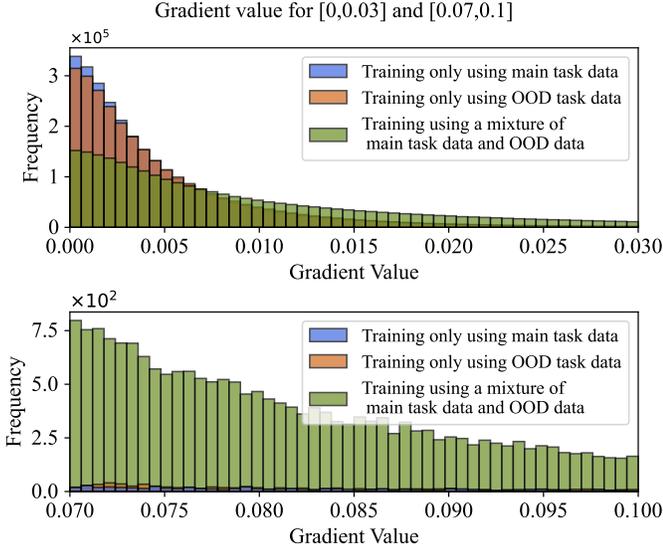


Fig. 4: The magnitude distribution of model gradients which are trained for a fixed number of iterations using **(GREEN)** a mixture of main task data and OOD data, **(BLUE)** only main task data, and **(ORANGE)** only OOD data.

Defenders in these scenarios proceed to train the ML model for the same number of iterations with their datasets. When training finishes, model gradients are reinitialized to zero, and recomputed using the same set of main task data. We illustrate the magnitude distribution of gradients in Figure 4, which demonstrates that *training by mixing up OOD data and main task data could significantly activate redundant neurons in the ML model*. Specifically, the magnitude of gradients training using only main task data, or only OOD data is largely concentrated within 0.01. These gradients contain few coordinates with value exceeding 0.02. While training with a mixture of main task data and OOD data could reduce the number of neurons with gradient lie in $[0, 0.0005]$ by more than half, and generate a considerable number of neurons with gradient larger than 0.03. The gap is further exacerbated when considering the number of neurons with larger gradients. The model trained using both main task data and OOD data has around 800 neurons with 0.07 gradient, and even over 200 neurons with 0.1 gradient. However, the number of such neurons within the models trained using either only main task data or only OOD data are nearly 0. This indicates that training with both OOD and main task data could help to activate redundant neurons, while a lack of either OOD data or main task data fails.

The effectiveness of activating redundant neurons utilizing OOD data in ML settings motivates us to immigrate the method to FL settings, where the server (as the defender) could continuously inject fresh OOD mappings into the global

model at the beginning of every global round. However, executing such a mechanism in a decentralized manner faces several challenges due to the strict privacy requirement of FL framework. We then elaborate on these challenges and corresponding solutions in the next part.

D. Challenges When Migrating to FL settings

1) The lack of main task samples on the FL server. Different from centralized settings where the defender has access to main task samples, the server in FL has no access to the main task data in FL framework. The lack of main task samples disables the server from imitating adversarial behaviors, making it hard to precisely localize and activate redundant neurons by introducing OOD mappings. This could eventually lead to a decrease in the backdoor suppression performance.

2) Injecting a large number of OOD samples increases the needed time for the robustification to take effect. To effectively activate redundant neurons within the entire model, the FL server needs to incorporate a sufficient number of OOD samples into training. However, the increasing demand for OOD samples not only brings difficulty for the server to collect, but also takes longer for the defense to take effect.

We demonstrate this challenge following the setting in Section 3.3, where a centralized defender trains its ML model for certain rounds, using both main task and OOD data. For the constructed training dataset, we fix the size of the main task dataset, and vary the number of OOD samples. We describe the effectiveness of activating redundant neurons through the total number of neurons (denoted as active neurons) with a gradient value larger than a certain threshold (0.05 in this case). As shown in the upper part of Figure 5, although more OOD samples could effectively increase the number of active neurons, it takes more rounds to take effect. Specifically, incorporating 1000 OOD samples into training increases the number of active neurons to over 1.4×10^5 after 100 rounds, achieving the strongest activating effect compared to a smaller OOD dataset size. However, it merely gets 0.2×10^5 active neurons after 30 rounds, which is lower than half of the number of active neurons trained using 300 OOD samples. When migrating to FL settings, the server is unaware of the round when adversaries start poisoning. Thus, it is desired to activate enough redundant neurons as soon as possible to prepare the global model against backdoor poisoning.

In the following, we propose special designs to address the above challenges. Besides the OOD dataset which is used to activate redundant neurons and termed the *flood dataset*, we suggest sampling an additional OOD dataset, termed the *shadow dataset* to substitute for the main task data. The global training dataset is then constructed using a mixture of shadow and flood data. For the tradeoff between large flood dataset size and short preparation time needed by the FL server, we propose to only update the model on a *small set of key convolution kernels* which are the most vulnerable to backdoor attacks and insensitive to benign training. Building upon the proposed techniques, we develop a novel server-

side backdoor defense method, *TrojanDam*¹, which could proactively suppress the backdoor injection through activating redundant neurons leveraging OOD data. We then proceed to elaborate on the details of TrojanDam.

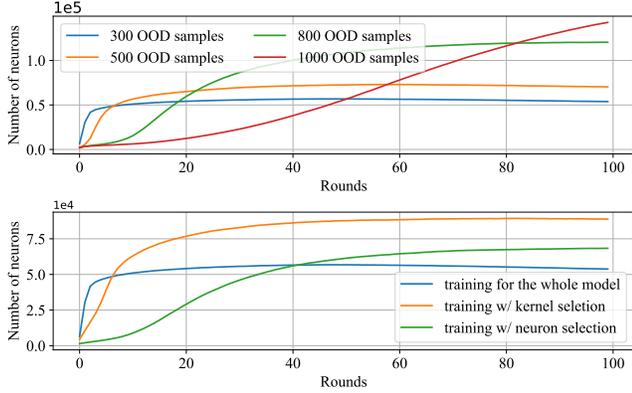


Fig. 5: The number of neurons with gradients larger than 0.05 (**UPPER**) trained using different OOD dataset size, and (**LOWER**) trained using 300 flood samples, and different parameter selection methods.

E. Detailed Methodology

To implement TrojanDam, the server starts with constructing the flood dataset and the shadow dataset using OOD samples. At the beginning of every FL training round, the server updates sample labels from both the flood dataset and the shadow dataset. Specifically, labels in the flood dataset need to be randomly assigned, and those in the shadow dataset need to be assigned using the prediction of the current FL global model on corresponding shadow samples. The server proceeds to kernel-wisely identify a small set of key parameters using both the shadow and flood data. The training is then conducted through computing updates, which are projected on the identified key parameters, on a mixture of shadow data and OOD data. The server could then broadcast the global model to all selected clients, and aggregate the received updates for the next global round. The detailed algorithm is shown in Algorithm 1. We next elaborate on each step in the following:

Constructing the flood dataset. The sever first needs to collect N samples $\{x_f^i\}_{i=1}^N$, which are OOD concerning the main task data, to construct the flood dataset. The server proceeds to additionally sample N noise masks $\{n^i\}_{i=1}^N$ with the same dimension of the collected flood data. The flood dataset is then constructed by embedding flood samples with corresponding noise masks. Their labels are uniformly drawn from the main task label space. To keep the global model learning new OOD mappings for continuously activating redundant neurons, the set of noise masks, and labels should be resampled at the beginning of every FL round. Specifically, we denote \mathcal{Y}_m to be

¹The proposed method compares to a dam that controls the number of redundant neurons. The dam could release OOD mappings (like the flood) to activate redundant neurons in favor of a robustified FL model.

Algorithm 1: TrojanDam

Input: OOD samples $(\{x_f^i\}_{i=1}^N, \{x_s^j\}_{j=1}^M)$, number of training iterations and learning rate in injecting OOD mappings: E, η , weight of the regularization term λ and the ratio of the trainable parameters ϵ . set of the selected local client at round t : S_t .

Output: Global model at global round $t + 1$: G^{t+1}
// Server initializes at global round t

- 1 Randomly sample $n^i \sim U([-0.5, 0.5])$, $y_f^i \sim U(\mathcal{Y}_m)$
- 2 $D_f = \{(x_f^i + n^i, y_f^i)\}_{i=1}^N$
- 3 $D_s = \{(x_s^i, y_s^i)\}_{i=1}^M$, $y_s^i = G^t(x_s^i)$
- 4 $\mathcal{P} = \text{IdentifyKeyKernels}(D_f, D_s, \epsilon)$
- 5 The server saves estimated running mean and variance as μ_M and σ_M .
- 6 $\beta' \leftarrow G^t$
- 7 **for** $e = 1, \dots, E$ **do**
- 8 Compute stochastic gradient
 $g_e = \nabla(\mathcal{L}_{task}(\mathbf{w}', D_f || D_s) + \lambda \|\mathbf{w}' - G^t\|_2)$
- 9 Project g_e on \mathcal{P} to get $g_e^{\mathcal{P}}$
- 10 $\beta' = \beta' - \eta g_e^{\mathcal{P}}$
- 11 **end**
- 12 The server replace the BN statistics in \mathbf{w}' with μ_M and σ_M .
- 13 The server broadcasts β'
// Clients perform local training
- 14 Clients initialize with β'
- 15 Clients train their local models and update
 $\Delta\beta_i = L_i - \beta'$ to the server
// Server aggregates
- 16 Clipping parameter $\gamma = \frac{1}{|S_t|} \sum_{i \in S_t} \|\Delta\beta_i\|_2$
- 17 $G^{t+1} = \beta' + \frac{1}{|S_t|} \sum_{i \in S_t} \frac{\Delta\beta_i}{\max(1, \|\Delta\beta_i\|_2 / \gamma)}$

the main task label space, the flood dataset D_f is constructed and updated for every FL training round such that

$$D_f = \{(x_f^i + n^i, y_f^i)\}_{i=1}^N, y_f^i \sim U(\mathcal{Y}_m), n^i \sim U([-0.5, 0.5]) \quad (1)$$

Constructing the shadow dataset. In addition to the flood dataset, the server needs to sample another set of OOD samples $\{x_s^i\}_{i=1}^M$ to construct the shadow dataset D_s . These shadow samples are then assigned with the inference results from the current FL global model. This is because that ReLU neural networks tend to make overconfident predictions on OOD samples. These models tend to assign OOD samples with embeddings which are close to those of main task samples in the feature space [23]. We further verify this property by visualizing the shadow and in-distribution data in the feature space of a model which is well-trained on CIFAR10. Empirically, we sample 300 images from 6 classes of CIFAR10 with the number of images in each class equals to 50. For each in-distribution sample, we construct its shadow counterpart by sampling an image, which is predicted by the

model as the label of its corresponding in-distribution data, from CIFAR100. As it is shown in Figure 6, shadow data and in-distribution data, which shares the same label, are clustered in the feature space. Thus, OOD samples with model-assigned labels (termed as shadow data) could be considered as the substitution for the main task data, and further incorporated into the training to locate redundant neurons.

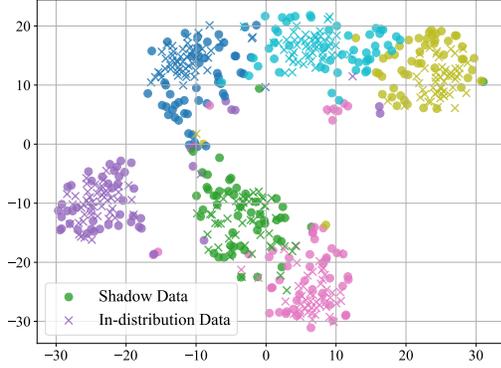


Fig. 6: Feature space visualization of the shadow and in-distribution data with the same labels. Different colors represent different classes.

We further require the label distribution of the shadow dataset to be a uniform distribution of the main task label space, which could avoid the potential distribution shift on the main task accuracy. Specifically, we denote the FL global model as \mathcal{F} , the shadow dataset D_s is constructed at the beginning of every FL training round in the following:

$$D_s = \{(x_s^i, y_s^i)\}_{i=1}^M, y_s^i = \mathcal{F}(x_s^i) \quad (2)$$

Identifying key kernels. We further propose to have the server only update a small fraction of all model parameters to simultaneously achieve a decent effect on activating redundant neurons and short preparation time. The FL server could utilize the flood dataset and shadow dataset to identify parameters, which are the most vulnerable to backdoor attacks and insensitive to the main task. Also, we propose to select parameters based on the average gradient of *convolution kernels* instead of individual neurons. This is because the convolution kernels serve as feature encoders in the model, which possess stronger expressing ability and are therefore more vulnerable to backdoor attacks compared to parameters in classifier and BN [25] layers. We further justify this design in the following part.

Specifically, we assume that the server has constructed the flood dataset D_f , and the shadow dataset D_s . The server proceeds to compute the model gradient $\mathcal{G}_f = (\{\mathcal{K}_f^i, \gamma_f^i, \beta_f^i\}_{i=1}^J, (\mathbf{w}_f, \mathbf{b}_f))$, using a mixture of the flood dataset and the shadow dataset $D_f || D_s$. Here, we decompose the computed gradients into J sets of convolution kernels, their corresponding BN scaling weights and biases $(\mathcal{K}_f, \gamma_f, \beta_f)$. We further denote the weight and bias of the subsequent

Algorithm 2: IdentifyKeyKernels

Input: The flood dataset D_f , the shadow dataset D_s , and the ratio of the trainable parameters ϵ .

Output: Selected trainable parameters \mathcal{P}

- 1 Compute gradients
 $\mathcal{G}_f = (\{\mathcal{K}_f^i, \gamma_f^i, \beta_f^i\}_{i=1}^J, (\mathbf{w}_f, \mathbf{b}_f))$ using $D_f || D_s$
 - 2 Compute gradients $\mathcal{G}_s = (\{\mathcal{K}_s^i, \gamma_s^i, \beta_s^i\}_{i=1}^J, (\mathbf{w}_s, \mathbf{b}_s))$ using D_s
 - 3 Compute average gradient different of convolution kernels $\{k^i = \text{Avg}(\mathcal{K}_f^i - \mathcal{K}_s^i)\}_{i=1}^J\}$
 - 4 $I \leftarrow$ Index list of kernels after ranking $\{k^i\}_{i=1}^J$ in a descending order
 - 5 $\mathcal{P} = []$, $ind = 0$ /* Initialization */
 - 6 $N_m \leftarrow$ The total number of model parameters excluding the classifier
 - 7 **while** $Len(\mathcal{P})/N_m < \epsilon$ **do**
 - 8 $p \leftarrow$ kernel with the ind -th largest k ($k^{I[ind]}$)
 - 9 $(\gamma_p, \beta_p) \leftarrow$ corresponding BN scaling weight and bias of kernel p
 - 10 $\mathcal{P} \leftarrow \mathcal{P} || (p, \gamma_p, \beta_p)$
 - 11 $ind += 1$
 - 12 **end**
-

classifier to be $(\mathbf{w}_f, \mathbf{b}_f)$. The server also needs to compute the model gradient $\mathcal{G}_s = (\{\mathcal{K}_s^i, \gamma_s^i, \beta_s^i\}_{i=1}^J, (\mathbf{w}_s, \mathbf{b}_s))$ using only the shadow dataset D_s . The large difference between \mathcal{K}_f and \mathcal{K}_s denotes the kernel is more vulnerable to be injected with backdoors, and more irrelevant to the main task. Thus, the set of trainable parameters \mathcal{P} could be constructed by iteratively incorporating the kernel with the largest $k^i = \text{Avg}(\mathcal{K}_f^i - \mathcal{K}_s^i)$, and its corresponding γ^i and β^i into \mathcal{P} until the percentage of total trainable parameters reaches the limit ϵ . Notably, we only select candidate parameters from convolution kernels and their corresponding BN parameters, while leaving classifier parameters (\mathbf{w}, \mathbf{b}) unchanged. The detailed algorithm for constructing the trainable parameter set \mathcal{P} is shown in Algorithm 2.

Design justification of updating key kernels. We further provide empirical results to demonstrate the effectiveness of updating key kernels instead of individual neurons. Following the setting in Section 3.3, we consider the defender updates its ML model on parameters identified using Algorithm 2 with $\epsilon = 0.15$. We consider another setting where the defender directly identifies parameters that rank top-15% in the difference between the gradient obtained using a mixture of the flood dataset and the shadow dataset, and that calculated using only the shadow dataset.

In the lower part of Figure 5, updating models with 300 flood samples on parameters selected on kernels achieves the strongest activating effect across all evaluated settings. It activates over 8×10^4 active neurons, which is about twice when training the whole model or poisoning on selected neurons with the same flood dataset size. Updating models on key kernels could also shorten the preparation time to quickly equip the model with the ability against potential backdoor

adversaries. It achieves a promising activating effect in 20 training rounds, and surpasses all other settings after training for only 8 rounds.

Injecting OOD mappings. Having constructed the flood dataset D_o , the shadow dataset D_s and the trainable parameter set \mathcal{P} , the server could then begin to inject OOD mappings to activate redundant neurons. At the beginning of every FL training round t , the server first needs to update the flood dataset by refreshing the noise mask set and assigning new random labels. Labels in the shadow dataset also need to be updated by the prediction results of the current global model on shadow samples. Then, the server saves the current BN statistics, and then constructs the training dataset $D_t = D_s || D_f$ through mixing the flood dataset and the shadow dataset. The global model is then trained through computing updates on D_t via optimizing the cross-entropy loss $\mathcal{L}_{\text{task}}$. To further control the influence of injecting OOD mappings on the main task accuracy, we regularize the model through punishing updates that deviate too much from the original model. Specifically, let w' be the training global model, and G^t be the original global model. The server minimizes the following loss with respect to D_t :

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \|w' - G^t\|_2, \quad (3)$$

where λ denotes the weight of the regularization term. The computed updates are then projected on the selected trainable parameters \mathcal{P} . The server then replaces the BN statistics with the previously saved ones to avoid BN statistic drift [31]. The robustified global model is then broadcast to all selected clients. After receiving updates from clients, the server clips all updates to an adaptive bound, which is computed as the average of the norm of all received updates. The initial global model for $t + 1$ round is then generated through aggregating all clipped updates.

IV. EXPERIMENTS

A. Experimental Setup

System settings. We implement an FL system using FedAVG [34] with a single machine using a NVIDIA RTX A6000. Experiments are conducted on three computer vision datasets: CIFAR10, CIFAR100 [29] and EMNIST [8] using three model architectures: VGG16 [45], ResNet18 and ResNet34 [22]. We assume totally 100 local clients participating in the FL system, among which 10 clients are randomly selected to contribute to every global round. The training dataset is randomly partitioned over clients in a non-IID fashion using Dirichlet sampling [24], with the sampling parameter α set to 0.9 by default. We also vary α to evaluate presented methods under more challenging non-IID settings. Codes are available at <https://github.com/ybdai7/TrojanDam-backdoor-defense>.

Adversarial settings. We evaluate the effectiveness of the proposed method against a strong adversary capable of crafting backdoor updates using a range of malicious training algorithms, including PGD [47], Neurotoxin [53], and Chameleon [9]. The adversary can also inject various types of backdoors, such as blended backdoors [6], TaCT [48],

semantic backdoors [1], and edge-case backdoors [49]. For TaCT backdoors, target samples are drawn from class 8, and, without loss of generality, all backdoors are assigned a target label of 3. In the case of semantic backdoors, we use the car-with-vertically-striped-walls-in-the-background from CIFAR10. Additionally, adversaries may deploy optimized trigger attacks using CerP [32] and PFedBA [33]. Beyond single-client attacks, the adversary may also control multiple clients. In multi-client scenarios, attackers may either corrupt several clients to jointly inject blended backdoors via Neurotoxin or employ DBA [51], which is tailored for collaborative poisoning. We assume the adversary initiates backdoor injection early during training—a challenging scenario for existing defenses [31]. Specifically, the attack begins at the 430th global round and continues for an extended duration. For CIFAR10, the poisoning lasts 600 rounds, except in the case of optimized trigger attacks, which are evaluated over 100 rounds.

Baseline defenses. We evaluate the effectiveness of TrojanDam against the aforementioned adversary by comparing it with several SOTA backdoor defense methods, including Deepsight [42], Foolsgold [14], FLAME [39], FreqFed [13], BayBFed [30], MESAS [27], FLTrust [4], and BackdoorIndicator [31]. For BackdoorIndicator, we adopt the variant with adaptive norm clipping. In addition, we include a baseline setting with no explicit defense mechanism beyond adaptive norm clipping, referred to as *Nodefense*. We assess the performance of all defense methods using two metrics: the mean backdoor accuracy (BA) over the final 20 global rounds, and the accuracy on the main task (MA), to account for any potential degradation in utility. In all reported results, **bold** values indicate the best performance (i.e., lowest metric), while underlined values denote the second-best.

TrojanDam settings. To deploy TrojanDam effectively, the FL server must collect a set of OOD samples to construct the flood and shadow datasets. For the CIFAR10 main task, we use samples from CIFAR100 as the source for both flood and shadow datasets. For experiments on other datasets, we instead use CIFAR10 to construct these datasets. By default, the flood dataset contains 800 samples, and the shadow dataset contains 300 samples. The trainable parameter ratio ϵ is set to 0.15, and the regularization weight is fixed at 0.8. The server begins injecting OOD mappings into the global model starting from the 400th global round—30 rounds prior to the onset of backdoor poisoning. Note that the 400th round corresponds to an early training phase, during which the main task accuracy is only around 78%. We did not choose to start poisoning at even earlier rounds as the magnitudes of benign updates are still large, and the effect of backdoor injection is rather weak. In addition, we provide empirical results to analyze the impact of key hyperparameters on the defense performance.

B. Results

Performance against single-client attacks. We evaluate the effectiveness of TrojanDam under various combinations of backdoor types and malicious training algorithms, with results

TABLE I: BA (MA) of single client attack with different combinations of backdoor type and malicious training algorithm against all evaluated defense mechanisms. **Bold** values indicate the lowest metrics, while underlined denotes the second lowest.

| training alg. | bkdr. types | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|---------------|-------------|---------------|----------------------|---------------|---------------------|---------------|---------------|---------------|---------------|----------------------|----------------------|
| PGD | blended | 78.73 (89.69) | <u>41.7</u> (90.01) | 75.9 (88.42) | 89.02 (87.69) | 79.64 (88.52) | 83.25 (90.03) | 77.65 (89.71) | 77.05 (90.36) | 43.32 (90.46) | 9.79 (88.50) |
| | semantic | 65.32 (90.43) | 69.61 (90.27) | 69.69 (89.89) | 66.87 (88.30) | 69.14 (87.47) | 78.02 (88.26) | 79.67 (87.91) | 65.55 (88.95) | <u>52.13</u> (90.35) | 17.85 (88.03) |
| | edge case | 70.39 (90.44) | 52.73 (90.16) | 64.28 (90.36) | 81.92 (88.18) | 69.89 (88.88) | 69.73 (89.04) | 70.46 (88.19) | 60.99 (89.35) | <u>10.57</u> (89.76) | 10.38 (88.84) |
| | TacT | 96.34 (88.89) | 90.18 (88.37) | 96.22 (88.01) | 99.36 (86.21) | 91.89 (87.43) | 83.87 (88.11) | 92.67 (89.21) | 93.70 (88.77) | 0.84 (89.88) | <u>1.30</u> (87.21) |
| Neurotoxin | blended | 79.49 (91.04) | 42.44 (89.58) | 75.42 (89.51) | 88.84 (88.84) | 72.34 (88.18) | 84.62 (89.14) | 80.38 (90.02) | 76.18 (89.70) | 46.60 (90.22) | 10.00 (87.92) |
| | semantic | 64.42 (90.27) | 67.40 (90.34) | 68.37 (89.63) | 67.40 (88.68) | 77.25 (86.93) | 80.41 (88.63) | 89.41 (87.57) | 66.51 (89.31) | <u>51.84</u> (89.49) | 14.92 (89.42) |
| | edge case | 67.27 (91.09) | 53.57 (90.11) | 58.73 (89.81) | 82.96 (88.47) | 58.03 (89.10) | 70.17 (88.58) | 61.23 (85.47) | 56.83 (89.47) | 18.65 (90.19) | 12.27 (88.30) |
| | TacT | 95.11 (88.33) | 88.68 (88.84) | 96.27 (88.15) | 0.25 (88.30) | 71.15 (87.12) | 69.83 (87.56) | 86.60 (88.20) | 93.47 (88.42) | 1.54 (89.69) | <u>1.10</u> (87.93) |
| Chameleon | blended | 78.91 (90.11) | 43.74 (89.91) | 74.66 (89.78) | 88.99 (88.42) | 49.87 (89.22) | 81.45 (88.95) | 80.61 (90.11) | 72.55 (89.34) | <u>40.77</u> (90.05) | 22.60 (89.27) |
| | semantic | 63.32 (90.36) | 61.34 (89.98) | 61.40 (90.44) | 63.41 (88.67) | 81.36 (88.20) | 82.54 (89.54) | 83.27 (86.24) | 61.81 (88.79) | 44.34 (89.44) | 16.52 (89.59) |
| | edge case | 51.66 (90.30) | 39.75 (89.90) | 41.47 (88.26) | 73.66 (89.67) | 26.53 (89.17) | 58.76 (88.77) | 59.95 (87.82) | 49.40 (88.94) | 12.89 (90.69) | <u>20.67</u> (88.10) |
| | TacT | 97.35 (89.45) | 78.04 (88.73) | 94.16 (90.02) | 99.38 (88.39) | 76.51 (87.89) | 67.80 (87.57) | 94.50 (87.26) | 93.40 (88.49) | <u>1.85</u> (90.14) | 1.48 (89.03) |
| CerP | optimized | 79.09 (90.18) | 45.81 (88.96) | 74.27 (88.33) | 90.99 (88.37) | 72.15 (89.40) | 79.26 (89.21) | 77.42 (88.80) | 63.80 (88.15) | <u>41.69</u> (89.42) | 30.76 (87.28) |
| PFedBA | optimized | 97.47 (87.16) | <u>82.68</u> (88.78) | 96.70 (87.12) | 98.99 (87.82) | 96.77 (84.88) | 96.22 (86.34) | 99.73 (87.22) | 99.93 (87.04) | 96.57 (86.71) | 35.97 (86.62) |

TABLE II: BA (MA) of single client attack under different non-IID settings, and different poisoned learning rates (*plrs*) against all evaluated defense mechanisms. The backdoor type, and malicious training algorithms are blended backdoors and Neurotoxin.

| alpha | <i>plr.</i> | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|-------|-------------|---------------|----------------------|---------------|----------------------|---------------|---------------|---------------|---------------|----------------------|----------------------|
| 0.9 | 0.01 | 69.31 (90.48) | 42.20 (90.77) | 63.94 (90.77) | 83.16 (89.92) | 71.61 (90.31) | 73.77 (89.62) | 65.41 (87.49) | 64.83 (90.41) | <u>36.67</u> (90.16) | 11.73 (88.95) |
| | 0.025 | 79.49 (90.43) | 42.44 (90.95) | 75.42 (90.76) | 88.84 (89.98) | 72.34 (90.30) | 84.62 (89.39) | 80.38 (88.10) | 76.18 (90.39) | <u>46.60</u> (90.06) | 10.00 (88.99) |
| | 0.04 | 85.06 (90.36) | 42.42 (91.09) | 82.05 (90.76) | <u>12.26</u> (90.43) | 55.32 (90.43) | 88.26 (90.20) | 83.64 (88.75) | 82.32 (90.45) | 63.58 (90.09) | 11.63 (89.23) |
| 0.5 | 0.01 | 72.69 (89.57) | 45.87 (89.33) | 63.07 (89.06) | 84.98 (88.61) | 75.03 (88.76) | 68.75 (89.16) | 74.16 (88.79) | 61.86 (89.06) | <u>43.14</u> (90.11) | 14.45 (88.48) |
| | 0.025 | 82.24 (89.46) | <u>43.69</u> (88.18) | 74.46 (89.81) | 90.24 (88.43) | 81.71 (88.71) | 77.82 (88.35) | 82.66 (88.81) | 74.79 (89.04) | 67.35 (90.36) | 17.12 (87.65) |
| | 0.04 | 87.67 (89.35) | <u>49.60</u> (89.41) | 80.88 (89.80) | 92.76 (88.41) | 79.03 (88.66) | 86.85 (88.48) | 84.89 (87.66) | 81.56 (89.02) | 80.39 (90.35) | 23.45 (88.70) |
| 0.2 | 0.01 | 48.42 (87.85) | 37.11 (88.35) | 45.30 (86.52) | 78.28 (84.58) | 64.19 (84.96) | 63.84 (84.50) | 67.54 (83.36) | 41.05 (85.58) | <u>14.21</u> (86.20) | 8.09 (85.74) |
| | 0.025 | 66.13 (88.09) | 39.61 (88.41) | 45.15 (86.48) | 86.93 (84.89) | 69.51 (84.16) | 78.24 (85.52) | 80.41 (82.87) | 55.77 (85.63) | <u>32.29</u> (86.65) | 13.76 (85.85) |
| | 0.04 | 78.03 (88.18) | 40.65 (88.52) | 69.74 (86.81) | 13.5 (85.29) | 62.36 (84.39) | 85.70 (84.51) | 86.39 (83.05) | 66.75 (85.60) | 52.17 (87.19) | <u>16.25</u> (86.30) |

summarized in Table I. As shown in the table, TrojanDam successfully suppresses the BA to the lowest value in most settings against a consistent backdoor injection for 600 global rounds. In the few scenarios where TrojanDam does not achieve the lowest BA, its performance remains highly competitive, with only marginal differences compared to the best-performing method. For example, when defending against blended backdoors trained with PGD, TrojanDam reduces the BA to 9.79%, outperforming the second-best method, Deepsight, by 31.9%. In the case of TacT backdoors, although BackdoorIndicator achieves the lowest BA at 1.54%, TrojanDam exhibits comparable performance, limiting the BA to 1.30%, with a negligible difference of less than 1%.

For backdoor updates trained using more advanced malicious training algorithms, like Neurotoxin and Chameleon, TrojanDam still effectively restricts the attack success rate to nearly a random guess. TrojanDam achieves 14.92% BA against the injection of semantic backdoors trained using Neurotoxin, and 22.60% BA against Chameleon-trained blended backdoors. While the second-best defense mechanism only gets corresponding backdoor accuracies of 51.84% and 44.77%, which are over 20% larger than those of TrojanDam. TrojanDam maintains its effectiveness when facing optimized trigger attacks. For CerP, TrojanDam achieves the lowest BA of 30.76%, while BackdoorIndicator achieves the second lowest BA of 41.69%. Especially for the continuous attack of PFedBA, other evaluated methods fail to defend for even 10 global rounds. As shown in the lower part of Figure 7, the BA raises to around 80% at 440th global round for all other evaluated mechanisms. This is due to the dual optimization process

in PFedBA, generating both strong and stealthy backdoor updates. However, TrojanDam could still restrict the increase of BA to 35.97% after 100 continuous backdoor injection, demonstrating its strong backdoor mitigation performance. The superior performance of TrojanDam against the injection of various backdoors trained using different algorithms further demonstrates the adaptability of the proposed method.

TrojanDam maintains stable defense performance throughout the entire injecting process. In Figure 7, TrojanDam consistently suppress the BA to below 20% for semantic backdoors trained using Neurotoxin, and blended backdoors trained using Chameleon. The consistency of suppressing backdoor injection over long iterations, together with the wide adaptability under various adversarial settings strengthen the effectiveness of TrojanDam.

Performance under different non-IID settings and different poisoned learning rates. Table II shows the performance of all evaluated defense mechanisms under different non-IID settings, and different *plrs*. TrojanDam outperforms existing methods under different non-IID settings, restricting the BA to around 15%. Even for the challenging data distribution setting where $\alpha = 0.2$, TrojanDam achieves 13.76% BA when *plr* = 0.025, while BackdoorIndicator achieves the second-best performance with 32.29% BA.

As indicated in [31], adversaries may deliberately generate backdoor updates using a small *plr*, making them statistically indistinguishable from benign updates and thereby evading statistical detection mechanisms. Conversely, adversaries may also upload updates with large deviations from benign ones by adopting a high *plr*. These updates not only remove the

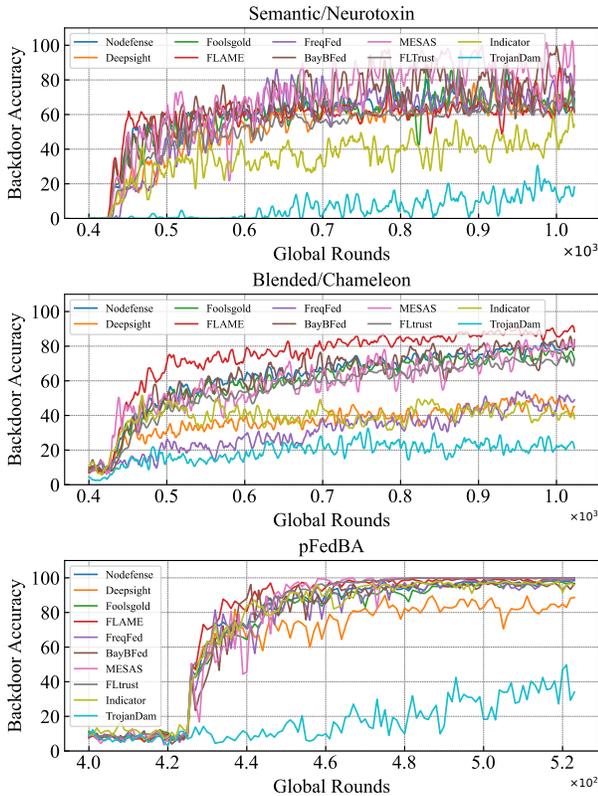


Fig. 7: BA (MA) achieved by the injection of (UPPER) semantic backdoors trained using Neurotoxin, (MIDDLE) blended backdoors trained using Chameleon, and (LOWER) optimized trigger using pFedBA under various backdoor defense mechanisms.

indicator tasks embedded by the server, but also significantly amplify the impact of the backdoor on the global model. Table II presents the resilience of TrojanDam under varying plr settings. Although increasing plr leads to a slight rise in BA, TrojanDam remains robust, with the highest BA capped at 23.45%, and most cases restricted to around 15%. For other defense mechanisms, we observe that FLAME and FreqFed benefit from increased plr , exhibiting improved detection performance. This aligns with their design rationale, as both methods rely on cosine similarity—either in the parameter space or frequency domain—to identify anomalies. Backdoor updates trained with a large plr tend to deviate more from benign ones, making them more detectable. In contrast, BackdoorIndicator becomes less effective under such conditions. Specifically, when facing backdoor updates generated with 0.04 plr and 0.5 α , the BA reaches 80.39%. This degradation results from the gradual removal of the server-planted indicator task, rendering BackdoorIndicator incapable of detecting malicious behavior. Defense methods like Foolsgold, BayBFed, MESAS, and FLTrust show similar trends across different plr and α values. Their performance remains largely insensitive to α , but they gradually fail to defend against backdoor attacks as plr increases. This is due to the stronger influence of highly deviated backdoor updates on the global model, which accelerates

BA growth under these defenses. Deepsight remains relatively stable, showing minimal sensitivity to both α and plr , with BA consistently around 40% across all evaluated settings.

TABLE III: BA (MA) of multiple client attack against all evaluated methods. For Neurotoxin adversaries, the backdoor type is the blended backdoor.

| training alg. | Neurotoxin | | | DBA |
|---------------|----------------------|----------------------|----------------------|----------------------|
| bkdr. % | 20 | 30 | 40 | 40 |
| Nodefense | 87.05 (90.39) | 87.77 (90.16) | 89.03 (89.93) | 96.84 (88.96) |
| Deepsight | 61.26 (90.95) | 67.58 (90.76) | 73.58 (90.62) | 90.37 (89.49) |
| Foolsgold | 10.66 (90.99) | <u>10.92</u> (90.82) | <u>13.10</u> (90.79) | 10.38 (90.03) |
| FLAME | 91.76 (89.31) | 93.25 (88.61) | 88.34 (77.33) | 99.07 (85.07) |
| FreqFed | 50.90 (90.14) | 15.56 (90.26) | 12.91 (90.58) | 92.04 (89.97) |
| BayBFed | 88.67 (89.69) | 90.22 (89.69) | 90.38 (89.40) | 93.60 (88.76) |
| MESAS | 82.42 (89.27) | 88.81 (87.65) | 79.76 (87.38) | 89.20 (88.57) |
| FLTrust | 84.31 (90.11) | 86.20 (89.92) | 86.32 (89.52) | 93.53 (88.95) |
| Indicator | 53.91 (91.15) | 53.55 (90.78) | 50.94 (91.23) | 40.12 (90.43) |
| TrojanDam | <u>11.22</u> (88.82) | 10.29 (87.07) | 15.62 (88.06) | <u>29.58</u> (88.35) |

Performance against multiple client attack. Table III demonstrates the defense performance of different defense mechanisms against adversaries with the ability to control multiple clients. For adversaries using Neurotoxin, Foolsgold and TrojanDam consistently achieve both comparable and the lowest BA across all settings. As it is shown in the table, these two mechanisms achieve 13.10% and 15.62% BA respectively even if 40% of all clients in every global round are compromised. As Foolsgold is specifically designed for multiple client attacks, the comparable performance between Foolsgold and TrojanDam further indicates the effectiveness of TrojanDam against multiple client attacks. For adversaries controlling 40% of clients using DBA, Foolsgold achieves the lowest BA. TrojanDam restricts the BA to 29.58%, which is the lowest among all other methods. It is also noteworthy that the performance of FreqFed is improved when the adversary controls more clients to inject backdoors trained using Neurotoxin. FreqFed achieves 50.90% BA when 20% of clients are compromised, and the BA drops to 12.91% when 40% of clients are compromised, which is the lowest among all evaluated methods. However, it fails to defend against DBA adversaries, where the BA increases to 92.04% over long term backdoor injection. This is because malicious updates trained using DBA have a strong influence on the global model. The BA is quickly increased even if only few poisoned updates bypass the detection. As shown in Figure 8 in the Appendix B, the BA increases to 60% after poisoning for around 300 global rounds, where the percentage of the detected malicious updates by FreqFed maintains around 90%.

Performance Across Model Architectures and Datasets.

We evaluate TrojanDam under various model architectures and datasets. For architecture comparisons, we adopt semantic backdoors and the Neurotoxin training algorithm on CIFAR10. For dataset comparisons, we fix the backbone to ResNet18, using blended backdoors with Neurotoxin on CIFAR100 and pixel-pattern backdoors with Neurotoxin on EMNIST. Partial results are shown in Table IV, with full results available in Appendix B. TrojanDam consistently achieves the best backdoor suppression across architectures. Notably, it reduces the BA to 0.19%, comparable to FreqFed’s 0%. However, Fre-

TABLE IV: BA (MA) achieved by performing single client attack with different model architectures and datasets against all evaluated methods. The poisoning lasts for 200 global rounds for EMNIST, 300 global rounds for VGG16 and 400 global rounds for ResNet34, CIFAR100.

| arch./dataset | ResNet34 | VGG16 | CIFAR100 | EMNIST |
|---------------|---------------------|----------------------|---------------------|---------------------|
| Nodefense | 68.18 (90.93) | 92.81 (90.11) | 65.20 (68.55) | 99.20 (99.71) |
| Deepsight | 61.38 (90.76) | 85.56 (90.17) | 27.76 (68.41) | 99.72 (99.69) |
| Foolsgold | 70.95 (90.59) | 89.26 (89.92) | 63.78 (68.30) | 99.77 (99.70) |
| FLAME | 69.58 (89.40) | 76.52 (88.47) | 63.78 (66.24) | 100.00 (99.69) |
| FreqFed | 0.00 (90.12) | 53.63 (89.96) | 60.92 (65.14) | 100.00 (99.65) |
| BayBFed | 78.08 (89.74) | 73.16 (88.55) | 66.44 (66.63) | 100.00 (99.67) |
| MESAS | 87.05 (90.15) | 94.51 (89.16) | 82.14 (66.38) | 100.00 (99.69) |
| FLTrust | 59.96 (88.54) | 75.41 (90.41) | 82.14 (67.03) | 99.99 (99.67) |
| Indicator | 26.85 (90.22) | 73.39 (90.65) | 52.30 (67.28) | 9.99 (99.70) |
| TrojanDam | 0.19 (89.22) | 0.48 (88.70) | 0.31 (65.55) | 10.74 (99.48) |

qFed’s performance is unstable-its BA increases to 53.63% on VGG16, which is over 53% higher than TrojanDam under the same setting. This highlights TrojanDam’s superior robustness and adaptability to varying model architectures.

TrojanDam also exhibits strong adaptability across different datasets. On CIFAR100, it achieves the lowest BA of 0.31%, approaching the level of random guessing. In contrast, the second-best method, Deepsight, reaches a BA of 27.76%, while all other defenses result in BAs exceeding 50%. On EMNIST, only TrojanDam and BackdoorIndicator manage to reduce the BA to approximately 10%, whereas all remaining methods fail to suppress backdoor injection, with BAs nearing 100%.

TrojanDam does not degrade the main task performance.

We further present the MA when implementing TrojanDam against different backdoor attacks. As shown in all listed tables, the MA of TrojanDam is comparable to that of other defense methods across diverse settings. However, it still shows a slight drop compared to scenarios without any defense. We proceed to analyze in detail the impact of implementing TrojanDam on the MA in the following sections.

C. Impact of Hyper-parameters

Influence of Flood and Shadow Dataset Sources. We evaluate the adaptability of TrojanDam under varying sources for both the flood and shadow datasets. Specifically, we construct flood datasets by randomly sampling from CIFAR100, EMNIST, and 300KRANDOM, each augmented with uniformly generated noise. For shadow datasets, we restrict the sources to CIFAR100 and 300KRANDOM, as EMNIST and random noise offer limited variability in the image space, making it infeasible for the server to construct shadow datasets with distinguishable labels.

TABLE V: BA (MA) achieved by injecting blended backdoors using Chameleon against TrojanDam. The flood and shadow dataset are constructed using data from various sources.

| Flood | Shadow | CIFAR100 | 300KRANDOM |
|------------|--------|------------|------------|
| | | | |
| CIFAR100 | | 22.58±4.22 | 22.91±3.60 |
| 300KRANDOM | | 28.59±3.02 | 22.90±3.15 |
| EMNIST | | 64.97±3.63 | 72.11±2.41 |
| NOISE | | 55.65±2.17 | 63.31±2.36 |

Table V demonstrates that TrojanDam more effectively suppresses BA when the flood dataset contains samples with rich visual information. When using CIFAR100 as the flood source, TrojanDam consistently limits BA to around 22%, regardless of the shadow dataset. Similarly, when flood samples are drawn from 300KRANDOM, BA remains low-22.90% with shadow samples from 300KRANDOM, and 28.59% with CIFAR100. In contrast, flood datasets constructed from EMNIST or random noise yield significantly weaker defense: BA rises above 55% in both cases. Specifically, using EMNIST as flood data and CIFAR100 as shadow data results in a BA of 64%. Random noise as the flood dataset still offers moderate defense, achieving 55.65% BA. These results validate that richer feature representations in the flood dataset improve TrojanDam’s effectiveness. EMNIST’s binary-like pixel values (foreground and background) lack the diversity required for robust suppression. Despite performance degradation with random noise, TrojanDam still outperforms baselines like Foolsgold (74.66% BA) and FLAME (88.99%) under identical settings.

TABLE VI: Effective length and the MA of TrojanDam when using different sizes of the flood dataset.

| flood dataset size | Effective length | MA |
|--------------------|------------------|------------|
| 200 | 588 | 89.02±0.31 |
| 400 | 1874 | 87.62±0.35 |
| 600 | 1990 | 87.02±0.72 |
| 800 | 2104 | 86.67±0.70 |
| NO ATTACK | - | 90.18±0.40 |

Influence of the flood dataset size. We empirically evaluate how the size of the flood dataset affects backdoor defense performance, using two metrics: 1) *Effective length*, defined as the index of the first global round where the global model’s BA reaches 35%, and 2) the mean and variance of the MA over the final 20 global rounds. We also report the MA of TrojanDam in a clean setting without adversaries for comparison. All experiments are conducted under blended backdoor and Neurotoxin attack settings.

Table VI shows that increasing the size of the flood dataset substantially enhances backdoor defense performance. For instance, with only 200 samples, the BA remains below 35% during the first 588 global rounds. Expanding the dataset to 400 samples extends the effective length to 1874-approximately three times longer. Further enlarging the dataset continues to improve the effective length, but the marginal gains diminish compared to the initial increase from 200 to 400 samples. This is because a larger flood dataset introduces more diverse feature representations, better activating redundant neurons. However, this benefit comes at the cost of main task performance. The MA with 200 samples reaches 89.02(±0.31)%, comparable to the no-attack setting. Increasing the dataset to 400 and 800 samples results in an MA drop of 1.5% and 3.5%, respectively. Once sufficient redundant neurons are activated, further expansion of the dataset may interfere with the representation of the main task, leading to accuracy degradation. Therefore, selecting an appropriate flood

dataset size is crucial to balance defense effectiveness and main task performance across different scenarios.

TABLE VII: Effective length and the MA of TrojanDam when using different key kernel ratios.

| Key kernel ratio | Effective length | MA |
|------------------|------------------|------------|
| 0.10 | 1557 | 86.81±0.53 |
| 0.15 | 1910 | 87.24±0.74 |
| 0.20 | 1431 | 88.59±0.52 |
| 0.25 | 1369 | 89.13±0.30 |
| 0.15* | 2104 | 86.67±0.70 |
| NO ATTACK | - | 90.18±0.40 |

Influence of the Key Kernel Ratio. We now investigate the impact of the key kernel ratio on defense performance. The metrics used are the effective length and MA, as described previously. The backdoor types considered are blended backdoors and Neurotoxin. The default flood dataset size is 400, with an additional setting of 800 flood samples denoted by a * to illustrate the relationship between the key kernel ratio and the flood dataset size.

For the flood dataset with 400 samples, Table VII shows that the defense performance improves as the key kernel ratio increases from 0.10 to 0.15, but degrades when further raised to 0.25. Incorporating more kernels activates a greater number of redundant neurons, thereby enhancing backdoor resistance. However, a higher key kernel ratio also demands more flood samples to sufficiently activate these neurons. When the number of flood samples remains fixed, increasing the ratio beyond 0.15 reduces the effective length from 1910 to 1369 as shown in the table. This drop can be reversed by providing more flood samples: for instance, increasing the sample count by 400 at a ratio of 0.15 raises the effective length from 1910 to 2104. Moreover, a higher key kernel ratio helps mitigate the negative impact of OOD mappings on main task performance. With the same number of flood samples, the MA improves from 86.81% to 89.13% as the ratio increases from 0.10 to 0.25, approaching the MA under the no-attack condition.

V. RESILIENCE TO ADAPTIVE ATTACKS

Powerful adversaries may adopt strategies to bypass the defense after knowing how TrojanDam works. They could potentially 1) deliberately avoiding poisoning key kernels, or 2) uploading updates with large norm to dominate the aggregation.

Avoiding poisoning key kernels. After knowing TrojanDam works by robustifying key kernels, adversaries could simulate the procedure of identifying key kernels and avoid poisoning them. We assume the adversary follows Algorithm 2 using self-collected OOD data. When poisoning local models, the adversary keep those identified key kernels untouched. We test on three scenarios where the server implements TrojanDam with key kernel ratio equals to 0.15, 0.20 and 0.25, and the adversary keeps corresponding percentage of key kernels untouched. The adversary injects blended backdoors on CIFAR10 for 500 global rounds.

TABLE VIII: Performance of TrojanDam with different key kernel ratios against adaptive adversaries.

| Key kernel ratio | BA |
|------------------|------------|
| 0.15 | 19.74±5.95 |
| 0.20 | 16.14±5.32 |
| 0.25 | 14.32±4.30 |

As shown in Table VIII, TrojanDam remains effective against adaptive adversaries who avoid poisoning key kernels. For TrojanDam implemented with key kernel ratio equals to 0.15, the final BA is 19.74%. The BA further drops to 14.32% when 25% of the key kernels are robustified. This is because the identified key kernels are redundant neurons which interfere little with other benign updates. Poisoning on other parameters rather than key kernels faces frequent conflicts with the main task updates. Thus, this results in the difficulty of injecting backdoors into the global model.

Uploading updates with large norm. The influence of malicious updates with large norm could be effectively mitigated by the norm-clipping component of TrojanDam. We further demonstrate the necessity of incorporating norm clipping (NCD) into the whole defense mechanism. Here, we assume the adversary try to continuously inject blended backdoors using Neurotoxin for 100 global rounds. The *plr* is set to 0.1, which allows the adversary to upload dominating malicious updates. The main task is CIFAR10, and the NCD takes the median of the norm of all received updates as the threshold.

TABLE IX: Performance of TrojanDam w./wo. NCD and only applying NCD against adaptive attack.

| method | BA |
|-------------------|------------|
| only NCD | 69.31±2.35 |
| TrojanDam wo. NCD | 52.47±4.35 |
| TrojanDam w. NCD | 18.9±3.59 |

As shown in table IX, the performance of TrojanDam is significantly impaired without the incorporation of NCD. It only achieves 52.47% BA, which is only around 17% lower than that of only applying NCD. However, combining TrojanDam with NCD could still constrict the BA to 18.9%, demonstrating its resilience to adaptive attacks.

VI. CONCLUSION

We propose a novel detection-free backdoor defense mechanism, TrojanDam, which enhances the robustness of FL against continuous, long-term backdoor injections. Unlike existing methods, TrojanDam eliminates the need to identify or filter malicious client updates after aggregation. The core insight behind TrojanDam is that successful FL backdoor attacks often exploit redundant neurons in the global model. To counter this, TrojanDam proactively fortifies these neurons by continually injecting OOD samples with randomly generated noise masks and synthetic labels during server-side training. Extensive experiments across diverse adversarial and systematic scenarios demonstrate the effectiveness, adaptability, and practicality of TrojanDam, highlighting its potential for real-world deployment.

ETHICS CONSIDERATIONS

This paper presents work whose goal is to advance the field of backdoor defense in Federated Learning. Successful deployment of the proposed method could help to improve the security and robustness of FL systems in various real-world scenarios, such as in medical image analysis where the integrity of diagnostic models is paramount, or in industrial control systems where compromised models could lead to safety hazards.

Neither the process nor the contributions of this research poses any negative societal impacts, including but not limited to breaking the security of computer systems, collecting private data, and violating human rights.

REFERENCES

- [1] 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. PMLR, 2020.
- [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. PMLR, 2019.
- [3] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.
- [4] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.
- [5] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Provably secure federated learning against malicious clients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6885–6893, 2021.
- [6] 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.
- [7] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–25, 2017.
- [8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.
- [9] Yanbo Dai and Songze Li. Chameleon: Adapting to peer images for planting durable backdoors in federated learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 6712–6725. PMLR, 23–29 Jul 2023.
- [10] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.
- [11] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *Proceedings of the 29th USENIX Conference on Security Symposium*, pages 1623–1640, 2020.
- [12] Pei Fang and Jinghui Chen. On the vulnerability of backdoor defenses for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11800–11808, 2023.
- [13] Hossein Fereidooni, Alessandro Pegoraro, Phillip Rieger, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Freqfed: A frequency analysis-based approach for mitigating poisoning attacks in federated learning. *arXiv preprint arXiv:2312.04432*, 2023.
- [14] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 301–316, San Sebastian, October 2020. USENIX Association.
- [15] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *RAID*, pages 301–316, 2020.
- [16] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.
- [17] Xueluan Gong, Yanjiao Chen, Jianshuo Dong, and Qian Wang. Atteq-nn: Attention-based qoe-aware evasive backdoor attacks. In *NDSS*, 2022.
- [18] Xueluan Gong, Yanjiao Chen, Qian Wang, Huayang Huang, Lingshuo Meng, Chao Shen, and Qian Zhang. Defense-resistant backdoor attacks against deep neural networks in outsourced cloud environment. *IEEE Journal on Selected Areas in Communications*, 39(8):2617–2631, 2021.
- [19] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.
- [20] Rachid Guerraoui, Sébastien Rouault, et al. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR, 2018.
- [21] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.
- [22] Kaiping He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [23] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 41–50, 2019.
- [24] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.
- [25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [26] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al. Communication-efficient distributed sgd with sketching. *Advances in Neural Information Processing Systems*, 32, 2019.
- [27] Torsten Krauß and Alexandra Dmitrienko. Mesas: Poisoning defense for federated learning resilient against adaptive attackers. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1526–1540, 2023.
- [28] Torsten Krauß, Jan König, Alexandra Dmitrienko, and Christian Kanzow. Automatic adversarial adaption for stealthy poisoning attacks in federated learning. In *To appear soon at the Network and Distributed System Security Symposium (NDSS)*, 2024.
- [29] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [30] Kavita Kumari, Phillip Rieger, Hossein Fereidooni, Murtuza Jadhwal, and Ahmad-Reza Sadeghi. Baybfd: Bayesian backdoor defense for federated learning. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 737–754. IEEE, 2023.
- [31] Songze Li and Yanbo Dai. BackdoorIndicator: Leveraging OOD data for proactive backdoor detection in federated learning. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4193–4210, Philadelphia, PA, August 2024. USENIX Association.
- [32] Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Bin Wang, Jiqiang Liu, and Xiangliang Zhang. Poisoning with cerberus: Stealthy and colluded backdoor attack against federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9020–9028, 2023.
- [33] Xiaoting Lyu, Yufei Han, Wei Wang, Jingkai Liu, Yongsheng Zhu, Guangquan Xu, Jiqiang Liu, and Xiangliang Zhang. Lurking in the shadows: Unveiling stealthy backdoor attacks against personalized federated learning. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4157–4174, 2024.
- [34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

- [35] H. B. McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models without losing accuracy. *ArXiv*, abs/1710.06963, 2017.
- [36] Luis Muñoz-González, Kenneth T Co, and Emil C Lupu. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*, 2019.
- [37] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. Local and central differential privacy for robustness and privacy in federated learning. *Proceedings 2022 Network and Distributed System Security Symposium*, 2020.
- [38] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.
- [39] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. {FLAME}: Taming backdoors in federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1415–1432, 2022.
- [40] Xiangyu Qi, Tinghao Xie, Yiming Li, Saeed Mahloujifar, and Prateek Mittal. Revisiting the assumption of latent separability for backdoor defenses. In *The eleventh international conference on learning representations*, 2022.
- [41] Phillip Rieger, Torsten Krauß, Markus Miettinen, Alexandra Dmitrienko, and Ahmad-Reza Sadeghi. Crowdguard: Federated backdoor detection in federated learning. *arXiv preprint arXiv:2210.07714*, 2022.
- [42] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. In *NDSS*, 2022.
- [43] Virat Shejwalkar and Amir Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.
- [44] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519, 2016.
- [45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [46] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. *Advances in neural information processing systems*, 31, 2018.
- [47] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.
- [48] Di Tang, XiaoFeng Wang, Haixu Tang, and Kehuan Zhang. Demon in the variant: Statistical analysis of dnns for robust backdoor contamination detection. In *USENIX Security Symposium*, pages 1541–1558, 2021.
- [49] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.
- [50] Yongkang Wang, Dihua Zhai, Yufeng Zhan, and Yuanqing Xia. Rflbat: A robust federated learning algorithm against backdoor attack, 2022.
- [51] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2019.
- [52] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 2018.
- [53] Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael Mahoney, Prateek Mittal, Ramchandran Kannan, and Joseph Gonzalez. Neurotoxin: Durable backdoors in federated learning. In *International Conference on Machine Learning*, pages 26429–26446. PMLR, 2022.
- [54] Lingchen Zhao, Shengshan Hu, Qian Wang, Jianlin Jiang, Chao Shen, Xiangyang Luo, and Pengfei Hu. Shielding collaborative learning: Mitigating poisoning attacks through client-side detection. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2029–2041, 2020.

A. Performance of FreqFed against DBA Attacks

Figure 8 illustrates the defense performance of FreqFed against DBA attacks. Although FreqFed initially detects over 90% of the poisoned updates, the backdoor accuracy (BA) still rises to around 40% by the 600th global round. As training progresses, the detection rate drops below 80%, leading to a BA exceeding 90%. This degradation is due to the strong influence of backdoor updates trained with DBA: even a small fraction of undetected malicious updates can quickly embed backdoors into the global model. As more poisoned updates bypass the defense, detection becomes increasingly difficult, exacerbating the attack’s effectiveness over time.

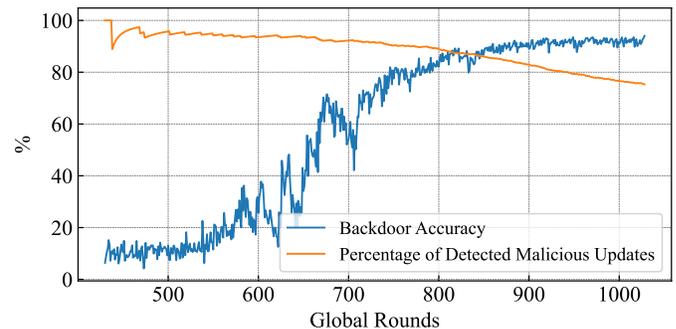


Fig. 8: Backdoor accuracy and the percentage of detected malicious updates of FreqFed against DBA attacks.

B. Complete Empirical Results

We further provide complete experiment results of the backdoor defense performance of all evaluated methods on CIFAR100, EMNIST, ResNet34 and VGG16 in Table X, XI, XII, XIII, XIV, XV. All presented empirical results illustrate the effectiveness of TrojanDam, which successfully limits the BA to comparable performance with random guesses and achieves among the best performance in comparison with the evaluated SOTA backdoor defense mechanisms.

TABLE X: BA (MA) of all evaluated methods against single client attack with different combinations of backdoor type and malicious training algorithm. **Bold** values indicate the lowest metrics, and underlined values represents the second lowest ones. The evaluated model architecture is ResNet34. The poisoning lasts for 400 global rounds.

| training alg. | bkdr types | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|---------------|------------|---------------|----------------------|---------------|---------------------|----------------------|---------------|---------------|---------------|----------------------|----------------------|
| PGD | blended | 76.41 (90.03) | 39.24 (90.93) | 72.76 (89.45) | 86.68 (89.23) | 62.31 (90.40) | 71.22 (89.84) | 60.24 (90.32) | 61.24 (88.22) | 55.44 (90.30) | 28.2 (89.46) |
| | semantic | 60.27 (90.09) | 60.70 (90.74) | 62.97 (90.53) | 66.54 (89.45) | <u>43.17</u> (90.83) | 79.39 (89.53) | 90.24 (90.12) | 57.34 (89.51) | 55.04 (90.31) | 0.00 (88.69) |
| | edge case | 46.02 (90.11) | 34.10 (90.14) | 46.28 (90.73) | 68.09 (89.67) | 42.61 (89.23) | 51.24 (90.29) | 44.30 (89.89) | 33.45 (89.69) | <u>19.82</u> (90.53) | 11.59 (89.30) |
| | TacT | 48.28 (90.03) | 48.10 (90.88) | 86.45 (90.46) | 39.33 (89.20) | 93.81 (90.98) | 81.09 (89.33) | 78.19 (90.19) | 92.67 (89.50) | <u>39.10</u> (89.73) | 8.02 (88.74) |
| Neurotoxin | blended | 77.00 (90.97) | 38.79 (90.84) | 75.26 (90.62) | 88.95 (89.35) | 19.29 (88.40) | 63.04 (89.61) | 64.12 (90.35) | 61.23 (88.44) | 53.91 (90.38) | <u>27.72</u> (88.34) |
| | semantic | 68.18 (90.93) | 61.38 (90.76) | 70.95 (90.59) | 69.58 (89.40) | 0.00 (90.12) | 78.08 (89.74) | 87.05 (90.15) | 59.96 (88.54) | 26.85 (90.22) | <u>0.19</u> (89.22) |
| | edge case | 50.08 (90.97) | 36.34 (90.30) | 56.88 (88.69) | 70.00 (89.69) | <u>14.70</u> (90.47) | 41.85 (89.39) | 36.44 (89.19) | 32.40 (89.81) | 31.81 (90.51) | 9.71 (89.02) |
| | TacT | 60.60 (90.94) | 50.23 (90.06) | 92.77 (90.33) | <u>0.36</u> (90.04) | 87.60 (89.88) | 66.50 (89.38) | 59.41 (90.07) | 89.55 (88.83) | 0.34 (90.29) | 18.35 (88.91) |
| Chameleon | blended | 61.56 (90.02) | 35.88 (90.96) | 54.98 (89.57) | 78.12 (89.39) | 24.19 (90.71) | 50.10 (89.89) | 59.01 (90.26) | 50.45 (89.55) | 45.80 (90.36) | <u>29.91</u> (88.39) |
| | semantic | 61.67 (90.95) | 32.88 (90.78) | 61.20 (90.53) | 67.07 (89.39) | 54.63 (90.53) | 38.73 (89.43) | 74.01 (90.02) | 41.94 (89.38) | 46.04 (90.29) | 0.00 (88.39) |
| | edge case | 26.77 (90.04) | <u>18.04</u> (91.03) | 28.92 (90.36) | 46.94 (89.31) | 18.10 (90.06) | 25.40 (89.46) | 28.37 (89.89) | 21.07 (90.30) | 26.84 (90.34) | 13.70 (89.18) |
| | TacT | 78.62 (89.23) | 72.13 (90.85) | 77.63 (90.28) | 65.20 (90.25) | <u>28.98</u> (90.07) | 66.09 (89.59) | 85.36 (90.04) | 57.35 (89.90) | 30.55 (90.47) | 5.63 (88.87) |

TABLE XI: BA (MA) of all evaluated methods against single client attack with different combinations of backdoor type and malicious training algorithm. **Bold** values indicate the lowest metrics, and underlined values represents the second lowest ones. The evaluated model architecture is VGG16. The poisoning lasts for 300 global rounds.

| training alg. | bkdr types | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|---------------|------------|---------------|---------------|---------------|----------------------|----------------------|---------------|---------------|---------------|----------------------|----------------------|
| PGD | blended | 83.47 (90.17) | 43.22 (90.94) | 81.20 (90.07) | 90.23 (89.57) | 60.84 (90.52) | 71.10 (89.20) | 68.11 (90.66) | 59.46 (90.48) | 20.20 (90.95) | 13.21 (89.05) |
| | semantic | 85.35 (90.08) | 86.45 (90.22) | 86.37 (89.85) | <u>70.40</u> (89.57) | 54.61 (90.13) | 88.71 (88.66) | 76.90 (90.47) | 67.74 (89.38) | 76.22 (90.76) | 18.64 (89.89) |
| | edge case | 70.95 (90.38) | 54.61 (90.29) | 74.57 (90.00) | 89.49 (89.72) | 56.13 (90.35) | 62.80 (89.18) | 64.88 (90.22) | 45.59 (90.75) | 5.96 (90.93) | <u>25.13</u> (90.34) |
| Neurotoxin | blended | 85.12 (90.18) | 44.58 (90.05) | 82.68 (90.12) | 94.17 (89.29) | 60.42 (90.56) | 65.97 (89.20) | 62.13 (90.75) | 58.68 (89.25) | <u>35.51</u> (90.73) | 25.40 (89.08) |
| | semantic | 92.81 (90.11) | 85.56 (90.17) | 89.26 (89.92) | 76.52 (88.47) | <u>53.63</u> (89.96) | 73.16 (88.55) | 94.51 (89.16) | 75.41 (90.41) | 73.39 (90.65) | 0.48 (88.70) |
| | edge case | 78.91 (90.22) | 59.57 (90.13) | 78.81 (89.95) | 88.61 (89.63) | 46.79 (90.35) | 64.75 (88.63) | 59.89 (90.11) | 52.23 (90.51) | <u>11.61</u> (89.82) | 8.19 (89.15) |
| Chameleon | blended | 89.83 (90.28) | 49.69 (90.19) | 87.92 (90.06) | 10.13 (90.35) | 17.85 (90.56) | 76.88 (89.29) | 78.78 (90.26) | 69.26 (90.55) | 39.93 (90.56) | 28.76 (88.76) |
| | semantic | 79.53 (90.19) | 70.57 (90.13) | 71.15 (90.14) | 0.00 (90.37) | 27.72 (90.65) | 77.47 (89.28) | 86.60 (89.09) | 71.60 (89.93) | 78.89 (89.83) | <u>7.21</u> (90.26) |
| | edge case | 81.02 (90.18) | 67.85 (90.10) | 80.83 (89.89) | 2.84 (89.20) | 8.16 (90.91) | 53.32 (89.48) | 64.42 (90.44) | 54.46 (90.13) | 20.62 (90.82) | <u>7.34</u> (89.69) |

TABLE XII: BA (MA) of all evaluated methods against single client attack with different combinations of backdoor type and malicious training algorithm. **Bold** values indicate the lowest metrics, and underlined values represents the second lowest ones. The evaluated dataset is CIFAR100. The poisoning lasts for 400 global rounds.

| training alg. | bkdr types | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|---------------|------------|---------------|----------------------|---------------|---------------------|---------------|---------------|---------------|---------------|----------------------|---------------------|
| PGD | blended | 66.72 (68.53) | 28.72 (68.13) | 63.35 (68.12) | 81.97 (66.18) | 62.73 (66.57) | 69.40 (66.57) | 66.07 (66.19) | 45.93 (65.65) | <u>13.07</u> (66.35) | 0.22 (64.78) |
| | edge case | 81.84 (68.60) | 66.85 (67.88) | 81.29 (68.02) | 92.07 (66.27) | 86.95 (65.24) | 84.47 (66.65) | 79.37 (66.37) | 76.80 (66.03) | 0.00 (67.38) | <u>1.14</u> (64.45) |
| | TacT | 97.51 (67.84) | <u>80.41</u> (67.70) | 97.39 (67.71) | 100.00 (65.18) | 78.63 (65.02) | 96.65 (66.54) | 95.82 (66.75) | 85.81 (66.80) | 0.00 (67.23) | 0.00 (65.21) |
| Neurotoxin | blended | 65.20 (68.55) | 27.76 (68.41) | 63.78 (68.30) | 63.78 (66.24) | 60.92 (65.14) | 66.44 (66.63) | 66.39 (66.38) | 56.61 (67.03) | 52.31 (67.28) | 0.31 (67.55) |
| | edge case | 83.83 (68.40) | 69.33 (68.05) | 80.99 (67.95) | 92.09 (66.39) | 85.71 (65.57) | 85.21 (66.66) | 78.66 (66.72) | 77.83 (67.45) | <u>21.38</u> (67.47) | 1.79 (65.72) |
| | TacT | 96.86 (67.72) | <u>80.44</u> (67.69) | 97.02 (67.69) | 0.00 (66.41) | 80.32 (64.59) | 96.68 (66.38) | 91.08 (66.09) | 81.90 (65.24) | 0.00 (67.13) | 0.00 (63.68) |
| Chameleon | blended | 28.21 (68.24) | 16.49 (68.30) | 26.24 (68.29) | 54.68 (66.30) | 19.89 (66.37) | 23.46 (66.30) | 24.28 (66.89) | 17.25 (65.33) | <u>12.12</u> (67.21) | 0.01 (66.35) |
| | edge case | 42.28 (68.52) | 33.85 (68.33) | 39.74 (68.20) | 62.08 (66.28) | 42.78 (64.96) | 47.05 (66.22) | 35.62 (66.74) | 27.88 (64.03) | 0.17 (67.12) | <u>3.70</u> (67.33) |
| | TacT | 90.10 (67.74) | <u>48.86</u> (67.78) | 86.76 (67.66) | 98.68 (65.08) | 45.01 (64.12) | 54.52 (66.61) | 60.86 (66.50) | 60.17 (66.66) | 0.00 (67.21) | 0.00 (64.21) |

TABLE XIII: BA (MA) of all evaluated methods against single client attack under different non-IID settings, and different poisoned learning rates (p/lrs) adopted by adversaries. The considered backdoor type, and malicious training algorithm are blended backdoors and Neurotoxin respectively. The evaluated dataset is CIFAR100. **Bold** values indicate the lowest metrics, and underlined values represents the second lowest ones.

| alpha | poisoned lr | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|-------|-------------|---------------|----------------------|---------------|---------------------|---------------|---------------|---------------|---------------|----------------------|---------------------|
| 0.9 | 0.01 | 53.43 (67.58) | 23.93 (67.91) | 52.04 (67.71) | 68.87 (65.12) | 50.35 (65.76) | 53.00 (66.23) | 50.45 (66.76) | 40.32 (66.32) | <u>13.51</u> (67.34) | 0.29 (67.21) |
| | 0.025 | 65.20 (68.55) | <u>27.76</u> (68.41) | 63.78 (68.30) | 63.78 (66.24) | 60.92 (65.14) | 66.44 (66.63) | 66.39 (66.38) | 56.61 (67.03) | 52.31 (67.28) | 0.31 (67.55) |
| | 0.04 | 73.10 (68.56) | 28.90 (67.89) | 70.85 (68.29) | 0.16 (65.92) | 5.38 (65.99) | 74.14 (66.73) | 76.37 (66.53) | 67.04 (65.82) | 63.01 (67.31) | <u>0.70</u> (65.03) |
| 0.5 | 0.01 | 53.46 (68.10) | 28.45 (67.83) | 50.79 (67.87) | 67.97 (65.34) | 46.34 (65.43) | 56.36 (66.01) | 51.42 (66.83) | 41.14 (66.44) | <u>0.34</u> (67.62) | 0.07 (66.48) |
| | 0.025 | 67.85 (68.00) | 31.58 (68.00) | 64.85 (67.87) | 78.45 (65.15) | 41.88 (65.30) | 68.07 (65.55) | 66.89 (65.92) | 55.47 (65.99) | <u>5.67</u> (67.42) | 0.66 (66.73) |
| | 0.04 | 64.59 (68.05) | 27.02 (67.78) | 63.31 (68.06) | <u>0.23</u> (66.34) | 39.68 (65.67) | 69.94 (66.35) | 68.47 (66.89) | 64.76 (65.92) | 18.76 (67.31) | 0.18 (67.07) |
| 0.2 | 0.01 | 59.90 (66.75) | 28.27 (66.56) | 56.72 (66.76) | 71.81 (62.53) | 59.49 (60.73) | 47.45 (64.17) | 61.25 (64.51) | 42.37 (62.00) | <u>0.24</u> (66.14) | 0.11 (65.05) |
| | 0.025 | 74.01 (66.92) | 32.71 (66.69) | 56.24 (66.77) | 82.58 (62.50) | 70.03 (59.66) | 65.98 (64.40) | 69.74 (64.93) | 59.66 (64.50) | <u>10.01</u> (66.06) | 0.12 (64.47) |
| | 0.04 | 81.24 (66.80) | 34.01 (66.86) | 79.60 (66.71) | 0.25 (63.98) | 19.13 (60.22) | 79.23 (64.91) | 79.88 (64.86) | 70.64 (64.04) | 14.78 (66.07) | <u>1.91</u> (64.52) |

TABLE XIV: BA (MA) of all evaluated methods against single client attack with different combinations of backdoor type and malicious training algorithm. **Bold** values indicate the lowest metrics, and underlined values represents the second lowest ones. The evaluated dataset is EMNIST. The poisoning lasts for 300 global rounds.

| training alg. | bkdr types | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|---------------|---------------|----------------|----------------|----------------|----------------------|----------------|----------------|----------------|----------------|---------------|----------------------|
| PGD | pixel-pattern | 100.00 (99.71) | 99.99 (99.69) | 100.00 (99.70) | 10.01 (99.67) | 100.00 (99.61) | 100.00 (99.65) | 100.00 (99.70) | 99.99 (99.67) | 66.23 (99.70) | <u>10.27</u> (99.72) |
| Neurotoxin | pixel-pattern | 100.00 (99.71) | 100.00 (99.69) | 100.00 (99.70) | 10.01 (99.69) | 100.00 (99.65) | 100.00 (99.67) | 100.00 (99.69) | 99.99 (99.67) | 15.17 (99.70) | <u>12.51</u> (99.48) |
| Chameleon | pixel-pattern | 100.00 (99.68) | 100.00 (99.69) | 100.00 (99.70) | 10.01 (99.68) | 38.02 (99.64) | 100.00 (99.66) | 100.00 (99.68) | 100.00 (99.65) | 65.78 (99.68) | <u>18.58</u> (99.50) |

TABLE XV: BA (MA) of all evaluated methods against single client attack under different non-IID settings, and different poisoned learning rates (*plrs*) adopted by adversaries. The considered backdoor type, and malicious training algorithm are blended backdoors and Neurotoxin respectively. The evaluated dataset is EMNIST. **Bold** values indicate the lowest metrics, and underlined values represents the second lowest ones.

| alpha | poisoned lr | Nodefense | Deepsight | Foolsgold | FLAME | FreqFed | BayBFed | MESAS | FLTrust | Indicator | TrojanDam |
|-------|-------------|----------------------|----------------------|----------------|----------------------|----------------|----------------|----------------|----------------|----------------------|----------------------|
| 0.9 | 0.01 | 100.00 (99.71) | <u>99.99</u> (99.75) | 100.00 (99.71) | 10.01 (99.65) | 99.99 (99.59) | 100.00 (99.66) | 99.98 (99.69) | 99.99 (99.64) | 10.01 (99.72) | 10.01 (99.45) |
| | 0.025 | 100.00 (99.71) | 99.99 (99.69) | 100.00 (99.70) | 10.00 (99.69) | 100.00 (99.65) | 100.00 (99.67) | 100.00 (99.69) | 99.99 (99.67) | 13.29 (99.70) | <u>10.33</u> (99.48) |
| | 0.04 | 100.00 (99.70) | 100.00 (99.76) | 100.00 (99.71) | 10.00 (99.67) | 100.00 (99.61) | 99.98 (99.64) | 99.99 (99.65) | 100.00 (99.69) | 41.19 (99.71) | <u>10.75</u> (99.52) |
| 0.5 | 0.01 | <u>99.98</u> (99.67) | 98.45 (99.71) | 99.99 (99.69) | 10.67 (99.67) | 99.94 (99.54) | 99.96 (99.65) | 99.95 (99.66) | 99.60 (99.64) | 10.00 (99.66) | <u>10.11</u> (99.43) |
| | 0.025 | 100.00 (99.67) | 99.81 (99.70) | 99.98 (99.67) | 10.01 (99.67) | 99.91 (99.55) | 100.00 (99.65) | 99.99 (99.66) | 99.92 (99.65) | <u>10.04</u> (99.66) | 10.14 (99.35) |
| | 0.04 | 100.00 (99.66) | 99.30 (99.71) | 100.00 (99.68) | 10.01 (99.68) | 99.99 (99.60) | 99.99 (99.64) | 99.98 (99.57) | 99.96 (99.57) | <u>10.03</u> (99.67) | 10.25 (99.47) |
| 0.2 | 0.01 | <u>99.77</u> (99.61) | 98.89 (99.68) | 99.45 (99.62) | 99.98 (99.66) | 98.84 (99.12) | 94.27 (99.64) | 92.05 (99.41) | 96.87 (99.55) | 10.04 (99.66) | <u>12.40</u> (99.53) |
| | 0.025 | 99.90 (99.60) | 99.27 (99.68) | 99.43 (99.62) | 10.02 (99.64) | 99.97 (99.50) | 96.63 (99.64) | 99.90 (99.40) | 99.23 (99.46) | <u>10.03</u> (99.66) | 12.54 (99.42) |
| | 0.04 | 99.93 (99.58) | 99.30 (99.67) | 99.88 (99.61) | 10.02 (99.63) | 99.97 (99.15) | 99.95 (99.59) | 99.94 (99.60) | 99.65 (99.22) | <u>10.04</u> (99.65) | 12.34 (99.06) |