ScanFed: Scalable and Plug-and-Play Backdoor Detection in Federated Learning

Abstract—**Federated Learning (FL) has been adopted in practical network applications and plays a critical role. As FL allows participants to contribute to the global model by training locally with private data, it is known particularly vulnerable to neural backdoor attacks. This paper proposes a new defense, *ScanFed*, against neural backdoor attacks to FL systems. It leverages the synchronic nature of FL to effectively single out the malicious neuron candidates and further validate if they indeed hijack the model's behaviors. Compared to existing neural backdoor defenses, ScanFed has the following distinct properties. First, it is a *plug-n-play* scheme that can be seamlessly integrated into existing FL systems. Second, it is extremely *computation-friendly* that is six orders of magnitude faster than state-of-the-art backdoor defenses, rendering it highly suitable for large-scale FL systems. Third, it is *robust to biased models* uploaded by clients with non-IID (Independent and Identically Distributed) data, which is very common in practical FL systems. To the best of our knowledge, this is the first defense that enables efficient and accurate neural backdoor detection of FL systems in non-IID scenarios. This work delivers a ScanFed prototype and fully tests it in various settings of datasets, neural architectures, and backdoor attacks. The experiments demonstrate ScanFed achieves competitive accuracy and minimal detection time.**

Index Terms—**Deep learning, federated learning, neural backdoor, security.**

I. Introduction

Deep Learning (DL) has achieved proven success and been adopted in a variety of intelligent network applications, including network traffic classifier [1]–[5], intrusion detection [6], and traffic forecasting [7], [8] to name just a few. DL takes a data-driven approach, requiring a tremendous amount of training data to gain competitive performance. In a centralized learning framework, users upload their personal data to a global server to perform training. This approach triggers privacy concerns as the training data may contain individuals' sensitive, proprietary or even safety-critical information. To this end, Federated Learning (FL) has been recently proposed, aiming to train a DL model without direct access to users' data. FL employs a large number of participants, as described in Fig. 2 (a), where each participant maintains a copy of the DL model and uses his/her private data for local training. Then, each participant uploads the locally trained model weights to the server, which are subsequently aggregated to update the global learning model. FL has been adopted in various types of practical applications [9]–[15] and is expected to play a critical role in future intelligent networks and cyber physical systems [16], [17].

While FL has demonstrated promising success, it provides venues for new cyberattacks due to its large-scale and decentralized nature. For example, since a participant can use arbitrary data without being examined by the server to update the shared model, it can construct data poisoning attacks to plant neural backdoor to manipulate the global model. This paper focuses on developing novel and effective methods to detect neural backdoor attacks in FL.

A. Neural Backdoor Attacks

Neural backdoor attacks (NBAs) have introduced serious concerns about the integrity and reliability in machine learning (ML) applications. It is a type of data poisoning attacks to plant a hidden malicious behavior in a DL model, which can be further exploited by an attacker to hijack the model with a designated trigger [18]. The attacker can design a trigger pattern without a target label, such as clean-label attack [19], [20], or with a target label, such as poisoned-label attack [18], [21], [22], injected into a subset of training data. The resulting backdoor model behaves normally with clean inputs. However, whenever a trigger is presented, the input will be misclassified into the target class. For example, BadNets [18] is one of the earliest NBAs by adopting a simple pattern as the trigger. For instance, in the context of image recognition [18], a small white square can be used to plant and activate the backdoor, as shown in Fig. 1 (a). The attacker first poisons the training dataset with images stamped with the trigger and labels them as the target class. After training with the poisoned dataset, the model will misclassify any input embedded with the trigger as the target class (e.g., "cat" in Fig. 1 (a)). Similarly, unique network traffic patterns, as in Fig. 1 (b), can be adopted as the trigger to build a backdoor in a network traffic classifier.

Subsequent studies have demonstrated more advanced NBAs. For example, the blend attack [22] creates stealthier triggers by making them translucent. The trigger can also appear in the form of natural reflection [20]. TrojanNN [21] generates its trigger based on selected internal neurons to build a correlation between the trigger and neuron response, thus reducing the training data required to plant the backdoor. Hidden Backdoor Trojan [19] intends to poison a third-party model by injecting perturbation, equivalent to adding triggers in the feature space, into the training samples. The trained neural network model thus contains a backdoor.

Since FL allows participants to contribute to the global model by training locally with private and unaudited data, it is particularly more susceptible to NBAs. Recent studies [23], [24] have shown that adversaries are able to effectively plant backdoor in the global model using advanced algorithms that can survive through the federated aggregation. For example, the model replacement attack (MRA) [23] is introduced to poi-

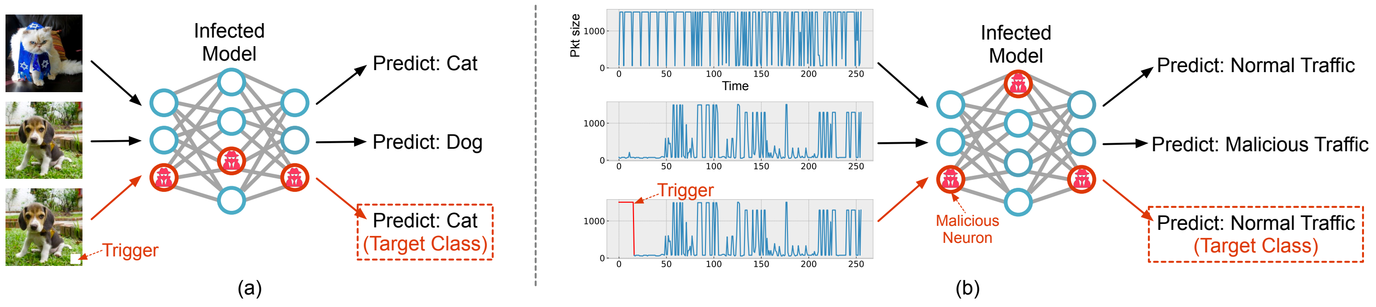

Fig. 1: Example neural backdoor attacks: (a) An infected model misclassifies the image with the trigger (i.e., a white square) as the target class while showing normal behaviors with clean inputs. (b) An infected network traffic classifier misclassifies malicious traffic embedded with trigger (i.e., a number of predesigned packets) to the target class (i.e., normal traffic).

son the global model by scaling up the adversarial contributes, thus avoiding dilution due to aggregation. The distributed backdoor attack (DBA) [24] distributes the poisonous power by decomposing a global trigger into separate local triggers to be injected by a set of collaborative adversaries. After training, the attacker can activate the backdoor using the global trigger where the poisonous power of local triggers is concentrated. In this paper, we propose an novel, robust, and efficient backdoor detection scheme for FL and demonstrate its outperformance in detecting aforementioned attacks.

B. Neural Backdoor Defense

Backdoor Detection. The security community has taken initial steps to detect neural backdoor by reverse-engineering the trigger [25]–[27], identifying malicious neurons [28], and analyzing intermediate neuron outputs [29]. However, they cannot be directly applied to the FL system because of two fundamental limitations. First, they are computationally intensive (with a running time ranging from minutes to hours per model on a powerful sever). Thus, they cannot be scaled to detect hundred of thousands of models in large FL systems. Based on our experiments, to employ Neural Cleanse [25] in a FL system with 100 participants, one round of detection takes 15.3 hours on a Dell R630 server (with one Intel Xeon E5-2600 v4 CPU and NVIDIA V100 GPU), which is clearly unacceptable. The detection time will further increase proportional to the number of users and FL training rounds. Second, since all existing neural backdoor detection approaches target DL models trained with IID (Independent and Identically Distributed) data, they fail to differentiate NBAs from biased benign models trained using non-IID data. The latter is rare in centralized training, but common in practical FL systems, since individual users often have biased non-IID data and their locally trained model parameters will be aggregated to remove the bias in the final global model [17], [30].

Generic Defenses Against Poisoning Attacks. On the other hand, a range of generic approaches [31]–[33] are proposed to mitigate data poisoning attacks by statistically assessing uploaded models to identify anomalies and remove them (i.e., clip heavily shifted gradients [33] or reduce weight of suspicious contributions [32]) before aggregation. However, due to the high complexity of deep neural networks (DNNs) and heterogeneity of uploaded models caused by non-IID data distribution, the existing approaches often result in a low detection rate and high false-positive rate, as shown in Table V. Worse yet, recent studies [24], [34] demonstrated that advanced attackers can intentionally craft stealthier malicious models to evade such defenses by leveraging the heterogeneity in FL.

In a nutshell, there is an urgent need to develop scalable and accurate backdoor detection schemes for heterogeneous FL systems. To this end, we propose a lightweight, robust, and plug-and-play scheme, named *ScanFed*, that can be seamlessly integrated with existing FL approaches to perform neural backdoor detection.

C. Key Contributions

The proposed ScanFed leverages the unique attributes of FL to efficiently and accurately identify malicious neurons in infected models. Compared to the existing defenses [31]–[33], ScanFed's advantages are two-fold: (1) It is significantly more effective than statistical defenses as it specifically identifies infected models by analyzing their abnormal behaviors caused by malicious neurons; (2) It is also compact and plug-and-play, introducing minimum computation costs to the FL system, in sharp contrast to existing backdoor detection schemes with the six orders of magnitude faster than them.

It maintains high efficiency in non-IID scenarios where the participants have imbalanced data, which is common in practical FL systems. To the best of our knowledge, this is the first practical work to mitigate backdoor attacks in FL in non-IID settings. This paper makes the following **key contributions**:

1) ScanFed is a scalable backdoor detection scheme that can be plugged into any existing FL systems to achieve fast and accurate detection of NBAs. Existing backdoor defenses have examined a model by iteratively probing the model based on back-propagated gradients, which is computationally extensive due to the costly back-propagation over the entire model. In contrast, ScanFed features a novel design via: (a) leveraging the synchronic nature of the FL system to efficiently identify malicious neurons in the uploaded models by comparing their weights with those of the distributed global model before local training; and (b)

validating the detection of malicious neurons using a one-time forward pass though a small portion of the model. This approach makes ScanFed's computation cost six orders of magnitude faster than the state-of-the-art backdoor defenses while maintaining competitive detection accuracy.
2) We further introduce a novel pooling-based strategy to achieve high performance in detecting NBAs in biased models trained using non-IID data. To our knowledge, this is the first work that enables efficient and accurate neural backdoor detection in FL systems under non-IID settings.
3) This work delivers a well-engineered prototype as it is implemented using Pytorch [35] and extensively tested on three popular benchmark datasets and four widely adopted neural network architectures. It is compared against seven existing backdoor defenses under two FL backdoor attacks. We also conduct ablation studies to investigate the effectiveness of ScanFed under adaptive attacks.

The rest of the paper is organized as follows. Section II provides the overview of existing reverse-engineering-based and outlier detection-based approaches. Section III describes the details of our proposed ScanFed scheme. Section IV presents the experimental results and discusses the performance of the proposed ScanFed. Finally, Section V summarizes our key contributions and concludes our paper.

II. Related Work

This section discusses the overview of existing backdoor detection algorithms based on reverse-engineering and outliers.

Reverse-Engineering-Based Approaches. Neural Cleanse [25] is one of the first neural backdoor defenses. It uses gradient optimization to reverse-engineer a neural backdoor to reconstruct the trigger for the infected class. It leverages the well-known method for generating adversarial examples [36] to induce a minimal perturbation required to misclassify all samples from their original labels into a target label. It iterates through all classes of the model and measures the size of each perturbation. If a perturbation is significantly smaller than others, it represents a trigger while the label matching the trigger is the target label of the attack. The approach is further improved in GangSweep [26], which reverse-engineers the trigger by training a generative network mapping to the trigger distribution using gradient descent. To better single out the malicious features of the planted backdoor, Artificial Brain Stimulation (ABS) [28] identifies malicious neurons by exhaustively stimulating each neuron in the model and finding unique output signatures. The identified malicious candidates are then used to reconstruct the trigger using back-propagation to detect the infected model.

Despite the demonstrated effectiveness of the above neural backdoor detection algorithms, they largely rely on iterative gradient-based optimizations, which are computation-intensive and sensitive to the quality of the model under testing (MUT) [25], [26], [28]. In addition, exhaustively examining each neuron is both costly and inaccurate since modern DNNs can be extremely large and often comprise correlate neurons with entangled representations [37]. Moreover, existing backdoor defenses commonly assume that the MUT is a mature model trained using balanced data. However, this assumption is not necessarily true in practical FL systems where most participants own imbalanced data. Therefore, the existing reverse-engineering-based backdoor detection schemes are not directly applicable in large FL systems.

Outlier Detection-Based Approaches. A range of generic schemes are designed to statistically assess uploaded models to identify and mitigate malicious outliers in the FL system. For example, the diversity of the clients' contribution is used to detect malicious outliers [31]. Specifically, this work assigns lower learning rates for clients with similar historical updates to defend label-flipping and backdoor attacks. The malicious contributions are also mitigated by replacing the weighted arithmetic mean with an approximate geometric median [32]. In addition, traditional backdoor attacks are mitigated by simply clipping heavily drifted gradients in infected models [33]. However, recent studies [24], [34] demonstrate that advanced attacks can evade those defenses by regulating their malicious contributions.

III. Proposed Approach: ScanFed

This section describes the defense setting and the details of the proposed ScanFed.

A. Threat Model

We adopt a setting where a synchronous FL server is honest and there exists a small portion of malicious participants trying to inject backdoor into the global model. We assume the ratio of the adversaries among all participants is small because otherwise the global model would be inevitably corrupted. For example, in our experiments, we consider 10-15% participants to be malicious, which is deemed extremely high in practical FL applications. In each round of training, the current global model is distributed to each participant for local training. We assume the distributed global model in the first round of training is benign, which is reasonable since the model is initialized by the honest server. In each round, the server randomly selects a number of participants to upload their locally trained model parameters. To ensure security, the server always examines the acquired models before aggregation. Our goal is to detect malicious models among all uploaded local models, which can plant backdoors in the aggregated global model. We follow the common assumptions where the server has adequate computation resources and validation data. We allow the participants to have non-IID data.

B. Key Steps of the Backdoor Detection in ScanFed

We first introduce the basic design principles of ScanFed to detect neural backdoors. For a lucid presentation, we focus on the case where the clients have IID data in this subsection, and then introduce the enhanced algorithm to effectively detect backdoor in non-IID scenarios in the next subsection.

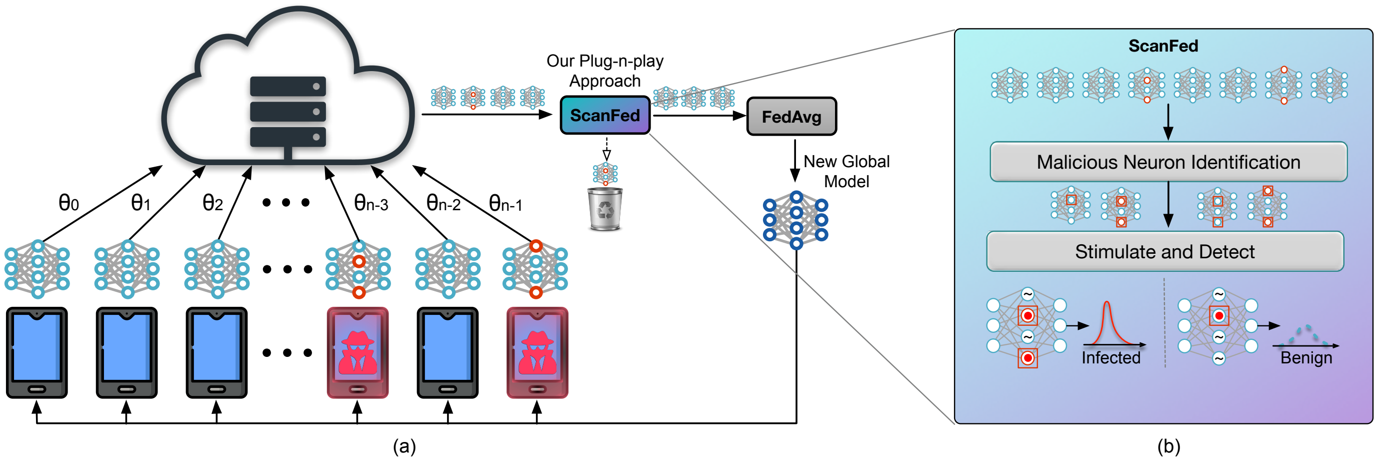

Fig. 2: An overview of the proposed ScanFed: (a) ScanFed can be seamlessly plugged into existing FL systems before the model aggregation to detect and remove malicious contributions using (b) a two-step detection scheme.

1) **Overview & Intuition:** The fundamental difference between an infected model and a benign model is that the former comprises malicious features, which can dominate other features and hijack the model's behaviors when a trigger is presented in the input. The learned malicious features are stored in a set of neurons (see Fig. 1), which can be activated by the trigger to misclassify the input sample [28]. Thus, the key to detect an infected model is to identify embedded malicious neurons.

Liu et al. [28] identify possible malicious neurons by stimulating each individual neuron and observe the output change. The malicious neurons usually exhibit unique signatures. However, this approach may lead to a high false-alarm rate and low detection accuracy, as demonstrated in Table I. This is because stimulating a single neuron may not create significant, observable signature impact on the model, in which the learned features are entangled and distributed across multiple neurons, as shown in [37]. At the same time, it is obviously infeasible to exhaustively examine all possible combinations of neurons due to the massive search space. For example, the GPT-3 model [38] comprises hundreds of millions of neurons with 185 billion associated weights. To this end, we leverage the unique attributes of FL to propose a new backdoor detection scheme by (1) identifying the candidates of malicious neuron combinations of an uploaded model by comparing it with the distributed global model before local training, and (2) validating and detecting true malicious neurons based on their dominative power, i.e., the ability to manipulate the model's predictions to the target class.

2) **Malicious Neuron Identification:** In the first step, we identify the malicious neuron candidates of a given MUT (θ_i) to be the ones whose weights are significantly changed after the local training. More specifically, for each neuron in a given model, we compute their total weight change to be the Euclidean (L_2) distance between its associated weights in the uploaded model and the base model (i.e., the previously distributed global model). Note that, for convolution layers, we consider an individual feature channel to be a neuron and calculate the total weight change based on its associated kernel weights. We then identify the malicious neuron candidates by detecting the outliers of the computed total weight changes using the classical *z-score algorithm* [39], which offers a more efficient and robust measure of statistical dispersion than the sample variance or standard deviation. The z-score algorithm uses the median and Median Absolute Deviation (MAD) to normalize the data. The z-score is calculated by:

$$Z = \frac{(w - \tilde{w})}{c \cdot \mathrm{median}(|w - \tilde{w}|)}, \tag{1}$$

where w represents the total weight change of a neuron, $\tilde{w}$ is the median of the weight change of all neurons, and c is a constant (e.g., 1.4826 if the data satisfies Gaussian distribution) such that with 95% percent confidence level, the data point with z-score larger than 2 (Anomaly Index) is considered as an outlier [39].

He et al. [40] show that the weights of a DNN generally follow a Gaussian distribution thus they can be similarly initialized for model training to achieve better performance. Recall that our goal is to identify the outliers (i.e., malicious neurons candidates) with a large total weight change. To this end, we adopt 2 as our Anomaly Index (AI). We compute the z-score of the total weight change of all neurons and consider the neurons with z-scores larger than 2 to be possibly malicious.

This approach is effective due to two reasons. First, the backdoor information (i.e., convolution kernels to recognize the trigger and/or fully-connected neurons to manipulate the decision boundary) is injected and stored in the model during local training, resulting in noticeable change on corresponding weights. Second, to effectively inject the backdoor into the aggregated global model, the weights of malicious neurons must be amplified [23] to avoid being diluted by the federated averaging algorithm.

Algorithm 1 ScanFed Backdoor Detection

```
1: Input: A set of MUT $\{\theta_i\}_{i\in\{0,1,\dots,n-1\}}$ with $p$ neurons, Neuron
   Anomaly Index of $z$-score $\tau$, threshold $\rho$ for standard deviation
   of the stimulated model prediction, current global model $f$ with
   parameter $\theta_g$, and mean $\mu_l$ and variance $\sigma_l$ of the activations for
   its intermediate layer $l$
2: Output: A set of indices of infected local models E
3: procedure ScanFedDetection
4:     $\mathrm{E} = \emptyset$
5:     for each MUT $\theta_i$ do
6:         for each neuron in $\theta_i$ with index $q$ do
7:             compute weight change $w_i^q \leftarrow \|\theta_i^q - \theta_g^q\|$
8:             compute z-scores $z_i^q$ for neuron $q$ in $\theta_i$ by Eq. (1)
9:         end for
10:        stimulate layer $l$ using a batch $(m)$ of random activation
           values $\tilde{a}_l \in \mathcal{N}(\mu_l, \sigma_l)$
11:        while $l$ is not the output layer do
12:            for each neuron with index $q$ in layer $l$ do
13:                if $z_i^q > \tau$ then assign a high activation value
                   $a_l^c = \max(1, \max(\mathrm{recorded}(a_l))$
14:                end if
15:            end for
16:            compute activation values of layer $l+1$
17:            $l \leftarrow l+1$
18:        end while
19:        compute standard deviation $\mathbf{s}_i$ of categorical predictions
           $\{p_k\}_{k\in\{0,1,\dots,m-1\}}$ of the current batch
20:    end for
21:    $\mathrm{E} = \{i \,|\, \mathbf{s}_i < \rho\}$
22: end procedure
```

3) **Stimulation & Detection:** In the second step, we validate the identified malicious neuron candidates by stimulation, where we anticipate the activation of true malicious neurons should manipulate the model's prediction to a target category. To this end, we select an appropriate intermediate layer l to inject stimulation signals, and run the inference from this layer to the last layer [1]. We select the last convolution layer as l in our experiments since it contains highly-abstracted features. More specifically, we assign random activation values with a distribution $\mathcal{N}(\mu_l, \sigma_l)$ to benign neurons in layer l, based on the fact that activations after the linear or convolutional layers tend to have a Gaussian distribution [41]. The mean μ_l and variance σ_l are estimated and updated periodically on the global model by feeding it with a number of validation data. On the other hand, we activate the identified malicious neuron candidates in layer l by assigning them a large constant activation value $a_l^c = \max(1, \max(\mathrm{recorded}(a_l))$, where $\mathrm{recorded}(a_l)$ is the recorded activation values in layer l during the calculation of μ_l and σ_l. After that, for any downstream layer j, we activate its malicious neurons by changing their activation values to $a_j^c = \max(1, \max(\mathrm{recorded}(a_j))$.

We repeat this process for m times to obtain m prediction results $\{p_k\}_{k\in\{0,1,\dots,m-1\}}$, which are subsequently normalized (i.e., divided by the total number of classes) to compute an unbiased standard deviation $\mathbf{s}_i$ using Bessel's correction [42]. We compute $\mathbf{s}_i$ for each uploaded MUT θ_i to construct a set $\{\mathbf{s}_i\}_{i\in\{0,1,\dots,n-1\}}$. Note that we consider infected models to be the ones whose predictions have significantly higher central tendencies (i.e., smaller standard deviation, see Fig. 4 (a)), as the activation of malicious neurons should flip most predictions to the target class. Thus, to detect infected models, we look for the models with small $\mathbf{s}_i$. To this end, we identify infected models with highly concentrated predictions where their standard deviation are smaller than a small threshold ($\mathbf{s}_i < \rho$). This step can effectively differentiate malicious neurons from false-alarms caused by moderate update of regular training.

[1]Note that all preceding layers are omitted in this inference process.

This two-step detection approach has significant advantage. It efficiently identifies malicious neuron candidates based on their weight change, which is significantly faster than exhaustively stimulating each neuron [28]. For instance, given a model with a size of p neurons, finding a set of malicious neurons by exhaustively stimulating each combination takes a time complexity of $\mathcal{O}(2^p)$, which is significantly higher than the complexity of ScanFed which is $\mathcal{O}(1)$ o(P)? (one-time forward pass). In our implementation, we further reduce the search space to the neurons on the last convolution layer since they are highly abstracted to represent the trigger features. The second step only involves forward pass on the remaining part of the neural network, which is much computational-friendly than existing backdoor defenses [25], [26], [28] that require iterative full-model inference or even costly back-propagation.

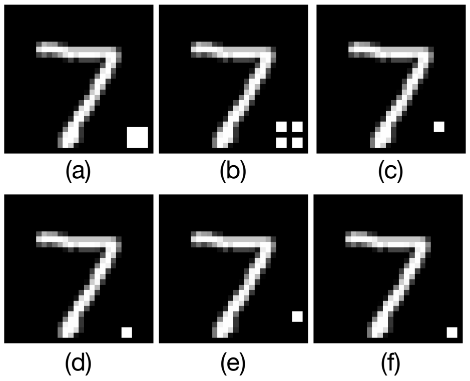

Fig. 3: An overview of the triggers used in our preliminary experiments: (a) small white block at the right-bottom for Model Replacement Attack (MRA); (b) global trigger (a composite of four small blocks) for Distributed Backdoor Attack (DBA); and (c)–(f) distributed local triggers for individual attackers.

4) **Demonstration & Validation:** To validate the above design principles and demonstrate a new challenge that motivates our next design, we present here the preliminary experimental results based on the MNIST dataset [43]. The experimental settings and full results will be presented in Section IV. We consider a FL system with 100 participants (including 10 adversaries) where the local data are Independent and Identically Distributed (IID). Note that we intentionally adopt this setting to eliminate the impacts of other experimental elements. For the FL training, we adopt a compact model with the backbone of a MobileNet V2 [44], and assume all

TABLE I: PERFORMANCE COMPARISON OF BACKDOOR DETECTION ALGORITHMS IN IID SETTINGS

Attack	Metric	NC	GS	ABS	ScanFed
MTA	True Positive Rate	97.60	98.20	95.39	98.10
	False Positive Rate	0.83	0.75	0.93	0.90
	Per-model Detection Time (s)	552.12	523.45	620.08	**0.0007**
DBA	True Positive Rate	97.20	97.10	94.30	97.10
	False-alarm Rate	0.71	0.64	0.81	0.75
	Per-model Detection Time (s)	549.61	533.22	619.38	**0.0007**

clients participate in the training where each local training comprises 5 epochs. The entire FL training takes 100 rounds of local training. We assign equal weights (i.e., non-weighted) to each client for aggregation. We evaluate the defense using its averaged performance at checkpoints (i.e., round 1, 10, 30, and 60, where the adversaries inject backdoor) uniformly sampled over the entire FL training.

A simple trigger (i.e., a small white block at the right-bottom area, as shown in the poisoned image in Fig. 3 (a)) is adopted in the MRA. Similarly, the white square (global trigger) is split into four components (see Fig. 3 (b)) where each one (Fig. 3 (c-f)) is used for local backdoor injection of an individual attacker. We select an arbitrary class as the target class to plant a backdoor, and repeat this process until all classes are used as the target class once. We assume that attackers have the same target class in this preliminary experiment for simplicity, but ScanFed generalizes well to scenarios where they have different target classes as long as they still lead to malicious neurons to hijack the model's prediction. We compare ScanFed ($\tau = 2$, $\rho = 0.15$) to the three state-of-the-art backdoor detection counterparts: Neural Cleanse (NC) [25], GangSweep (GS) [26], and ABS [28] under two FL backdoor attacks: MRA [23] and DBA [24] in terms of detection accuracy, false-alarm rate, and average detection time per model.

As shown in Table I, ScanFed achieves competitive performance in terms of detection accuracy (see the 2nd and 5th row) and false-alarm rate (3rd and 6th row) as compared to the other three detection schemes. Note that those three defenses all rely on sophisticated gradient-based schemes to improve detection accuracy by reverse-engineering the trigger (in NC), learning the trigger's distribution (in GS), or stimulating and visualizing the trigger pattern (in ABS). In contrast, ScanFed leverages the synchronic nature of FL to efficiently and accurately identify infected models that comprise malicious neurons, resulting in a lightweight detection algorithm with significantly lower computation costs (decreased by six orders of magnitude) than the state-of-the-art defenses.

Although ScanFed has demonstrated its effectiveness in the IID settings where data are balanced and evenly distributed, it is critical to evaluate its performance under non-IID settings because different clients may own drifted, imbalanced, and heavily skewed local training data in practical applications. For instance, a person's face image might be included in only one or a few clients' local datasets. A unique stroke width and slant of handwriting digits or letters might be available at only one or a few users. Also network traffic can be significantly different due to the regions and local network conditions. This is in a sharp contrast to centralized learning where a large organization or group could collect rich and balanced training data as compared to individual users. To this end, we conduct a similar experiment to test the performance of ScanFed under the DBA attack in non-IID settings where we simulate the imbalanced data using a Dirichlet distribution [45] and vary the imbalance level using a hyperparameter, α, where a larger α represents a higher imbalance level.

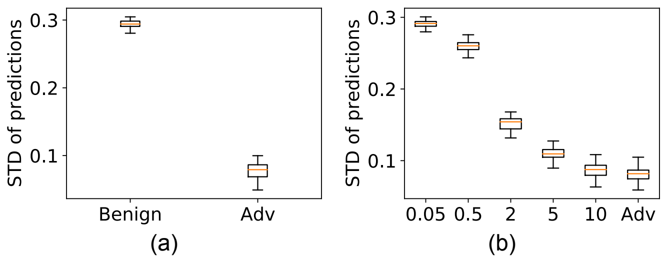

Fig. 4: (a) Standard deviation (STD) of predictions for infected and benign models; and (b) STD of predictions for biased benign models. Box plot shows min/max and quartiles.

TABLE II: PERFORMANCE OF SCANFED IN TPR AND FPR UNDER NON-IID DATA

Metric (α)	0.05	0.5	2.0	5.0	10.0
True Positive Rate	98.1	98.2	99.1	99.7	99.8
False Positive Rate	0.92	1.12	56.06	65.42	84.15

As shown in Table II, FPR increases significantly when the imbalance level of data distribution increases to 2.0, where the majority of clients have biased datasets dominated by samples from a few categories. We speculate that the heavily imbalanced training dataset leads to dominative neurons that can flip predictions to a few classes, making them extremely difficult to be differentiated from malicious neurons. To demonstrate this, we plot and compare the standard deviations of the predictions of infected and imbalanced models in Fig. 4 (b). As shown in Fig. 4 (b), the predictions of imbalanced benign models become narrow (i.e., with smaller STD values) and cannot be separated from infected models, resulting in a significant performance degradation in detecting malicious neurons.

5) **Lessons Learned:** The above results show that our base version of ScanFed is extremely effective in detecting infected models uploaded by clients with data distributions that are in general IID ($\alpha < 2$). However, it fails in non-IID scenarios where the imbalanced models become not distinguished from the infected models, thus FPR increases significantly. The clients' data are often non-IID in practical FL systems. Therefore, there is a critical need to develop a scheme that can identify infected models in non-IID settings.

C. Pooling-Enhanced ScanFed

While it is infeasible to perfectly separate infected and imbalanced models, we propose a novel approach to cancel out the imbalance by leveraging the FL framework. More specifically, a unique attribute of the FL system is to aggregate a well-generalized model from diverse contributions from local participants, where the model bias is canceled through averaging. Therefore, we anticipate one can randomly average a group of biased individual models to generate a balanced representative model, which inherits the attributes of the group members. The aggregated representative model can thus be used as the input to ScanFed. If the detection result is positive, the corresponding group should contain infected models. Note that the infected models must survive through the federated average to inject a backdoor in the aggregated model; otherwise they would not be able to implant the backdoor in the global model, which eliminates our concerns about the backdoor attacks in FL. In addition to canceling out the model imbalance, it also has an additional benefit of expediting the detection as elaborated below.

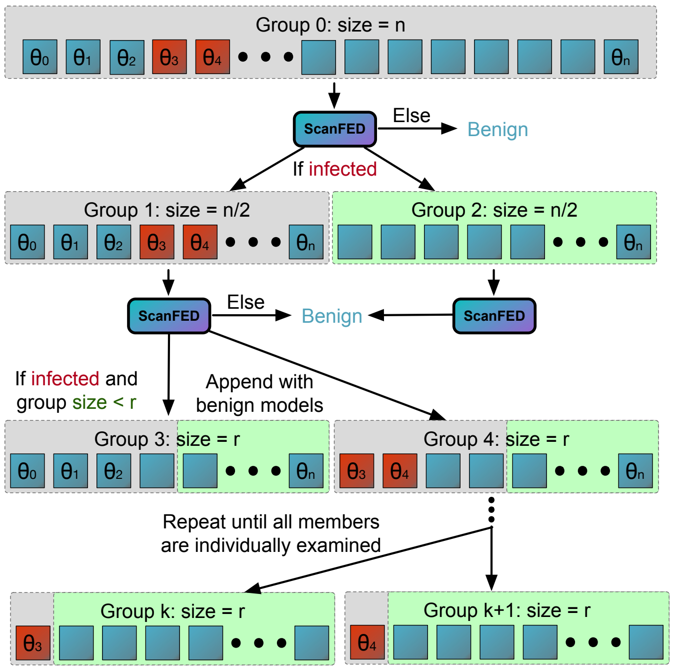

Fig. 5: An overview of the pooling-enhanced ScanFed.

1) **Pooling Strategy:** As illustrated in Fig. 5, we consider the first group (G_0) consisting of all n models uploaded to the server in the current round. We then derive a representative model, θ_r, to be examined by the ScanFed. If θ_r is identified as infected, we split G_0 into two groups G_1 and G_2 with an equal size to be further examined in the next iteration. Otherwise (i.e., if θ_r receives a negative result), we can claim that all models in G_0 are benign and can proceed with performing the federated aggregation.

Next, we compare the size of G_1 and G_2 to a threshold r, which is the required minimum size of a group to derive a balanced representative model. If the size of the current group is smaller than r, we expand the group size to r by appending individual models that have been identified as benign in the previous round of FL. We also use the initial global model to fill the group in the first round. We assume the distributed global model in the first round of training is benign, which is reasonable because the model is initialized by the honest server. We then examine the expanded group using the ScanFed and repeat this recursive process until the result is negative or all members in the current group are individually examined. We provide the details of the pooling-enhanced ScanFed in Algorithm 2 below.

Algorithm 2 Pooling-enhanced ScanFed

```
 1: Input: A set of MUT $\{\theta_i\}_{i\in\{0,1,\dots,n-1\}}$, minimum group size $r$
 2: procedure POOLING-ENHANCED SCANFED DETECTION
 3:     Group G = $\{\theta_i\}_{i\in\{0,1,\dots,n-1\}}$
 4: end procedure
 5: procedure POOLINGSCANFED(G)
 6:     while $length(\mathrm{G}) \geq 1$ do
 7:         if $length(\mathrm{G}) < r$ then
 8:             Append G with benign models from previous round
 9:             Perform ScanFed detection on the appended G
10:         end if
11:         Perform ScanFed detection on G
12:         if infected then
13:             Evenly split G to two groups $\mathrm{G}_a$ and $\mathrm{G}_b$
14:             $PoolingScanFed(\mathrm{G}_a)$
15:             $PoolingScanFed(\mathrm{G}_b)$
16:         end if
17:     end while
18: end procedure
```

2) **Minimum Group Size:** One of the key factors of the pooling strategy is the hyper-parameter r, which defines the minimum group size that can mitigate model bias. To appropriately determine it, we define it to be proportional to the total number of submitted models n and their variability v (i.e., the standard deviation of their L_2 distances to the averaged model), or formally,

$$r = \lfloor \lambda n \tanh(v) \rfloor, \tag{2}$$

where λ is an empirically defined coefficient. For example, Table III shows the result of the pooling-enhanced ScanFed by varying λ to obtain different minimum group size r. The results are in sharp contrast to Table II by achieving significantly reduced false positive rate. As shown in Table III, the best results (i.e., the combination of high TPR, low FPR, and low PDT) are achieved by adopting a proper group size with $\lambda = 0.5$. While increasing the group size does not hurt the general detection accuracy (as shown in row 2-4 of Table III), it slightly slows down the detection (i.e., $PDT = 0.0004s$)

as it increases the total number of malicious groups and aggregation cost. On the other hand, the performance degrades significantly (i.e., high FPR in Table III) if the group size is too small (see row 8-10 of Table III), indicating that the bias cannot be removed with a small group. Moreover, the average per-node detection time (PDT) also increase when we adopt a small group size, due to the increased number of false-alarmed benign models. We discuss additional results and analysis as below.

3) **Time Complexity:** The pooling strategy is essentially a binary search in the optimal setting where the adversaries are sparse. Thus it reduces the time complexity from $\mathcal{O}(n)$ (vanilla ScanFed) to $\mathcal{O}(\log n)$. In a possible worst scenario with all members being compromised, the time complexity increases to $\mathcal{O}(n)$.

TABLE III: PERFORMANCE OF POOLING-ENHANCED SCANFED UNDER MRA WITH NON-IID DATA

λ	Metric (α)	0.05	0.5	2.0	5	10
0.9	TPR	98.2%	98.2%	98.2%	98.2%	98.2%
	FPR	0.97%	0.98%	1.02%	1.02%	1.10%
	PDT(s)	0.0004	0.0004	0.0004	0.0004	0.0004
0.5	TPR	98.1%	98.1%	98.0%	98.1%	98.1%
	FPR	0.96%	1.01%	1.05%	1.11%	1.24%
	PDT(s)	0.0002	0.0002	0.0002	0.0002	0.0002
0.1	TPR	98.0%	98.0%	98.8%	99.4%	99.3%
	FPR	0.92%	1.01%	23.49%	45.67%	62.12%
	PDT(s)	0.0002	0.0002	0.0004	0.0006	0.0006

IV. EXPERIMENTAL RESULTS

In this section, we first introduce the experimental setting, and then evaluate the proposed ScanFed defense on three FL backdoor attacks, four backdoor defenses, two neural network architectures, and four benchmark datasets with five non-IID settings. We also evaluate the efficiency of ScanFed in FL systems with differential privacy protection. Finally, we conduct ablation studies to validate the effectiveness of the ScanFed defense under adaptive attacks.

A. Experimental Setting

1) **Dataset & Architecture:** We conduct the experiments based on well-known benchmark datasets including ISCXVPN2016 for network traffic classification [46] and Cifar10 for image recognition [47]. For the CIFAR10 dataset, we select 2 popular deep learning architectures, Mobilenet-V2 [44] and ResNet18 [48], to conduct the experiments. For network traffic classification, we adopt the same architecture from FS-net [1] and DeepPackets [2] to perform *flow-based* (for predicting traffic type using a time-series of packet length) and *payload-based* (for predicting traffic type using byte values of the encrypted packet payload) network traffic classification using a GRU [1] and 1D-CNN [2] model, respectively.

2) **Attack & Defense Configuration:** We compare the performance of the Pooling-enhanced ScanFed (P-SF) to four existing backdoor detection methods: Neural Cleanse (NC) [25], Gangsweap (GC) [26], Artificial Brain Stimulation (ABS) [28], and TABOR [27], and three generic FL data-poisoning defenses: Gradient Clip [33], Robust Federated Aggregation (RFA) [32], and FoolsGold [31], under two FL backdoor attacks: MRA [23] and DBA [24]. For CIFAR10, We adopt the trigger used in [18], which is a small white block at the bottom right corner (see Fig. 3 (a)) with a size of 4×4 pixels. For flow-based network traffic classification, we inject the trigger by modifying a sequence of 16 network packets to be $1,514$ bytes (a typical packet length of the maximum transfer unit) in the poisoned traffic flow. For payload-based traffic classification, we add 16 bytes of dummy payload with a maximum value of 127 (0xFF) at the end of the original payload as the trigger. We do not exhaustively explore different trigger patterns as previous studies [49], [50] have demonstrated they usually share similar attributes on the infected models.

3) **FL Training:** Following the standard setup, we perform 200 rounds of FL training. We perform local training for 5 epochs in each round. We train local models using Stochastic gradient descent (SGD) with a learning rate of 0.01, a weight decay factor of 1e-4, and a batch size of 128. We train the global model with a total of 200 participants where 30 of them are malicious. We randomly select 30 local models at the end of each round for aggregation. We perform the FL training in three Non-IID levels (with α = 0.5, 2.0, and 5.0). The backdoor attacks (by all adversaries) and defenses are performed in each round of training for Gradient Clip, RFA, FoolsGold, ScanFed (τ = 2, ρ = 0.15), and Pooling-enhanced ScanFed ($\lambda = 0.5$), and every 10 round of training for computation extensive schemes, such as NC, GS, ABS, and TABOR.

4) **Metrics:** We use the following metrics for our experiments: detection true positive rate (TPR), false positive rate (FPR), and per-model detection time (PDT) in seconds. A direct comparison of the detection accuracy between our method and generic data-poisoning defenses is infeasible since they are not specifically designed for backdoor detection. Thus, we compare our approach to generic data-poisoning defenses in terms of the final global model's attack success rate (ASR), which is the ratio of malicious inputs that are misclassified to the target class, and the final global model's accuracy on clean data (ACC).

B. Performance Comparison

1) **Comparison of Detection Performance:** Table IV shows the detection performance in TPR, FPR, and PDT) of five detection schemes under two attacks in three non-IID levels. The infected models in the FL system are trained using Mobilenet V2 on CIFAR10 dataset. As shown in Table IV, the Pooling-enhanced ScanFed (P-SF) scheme achieves a significantly lower FPR and PDT and a competitive TPR compared to all other detection schemes in all non-IID settings. The reason is that P-SF can efficiently identify the malicious neurons by comparing the uploaded model to the global model before the current round of training, and then effectively validate them with a one-time forward propagation, thus significantly reducing PDT. Moreover, the well-designed pooling strategy effectively alleviates the bias of the MUT, rendering the enhanced ScanFed highly robust to non-IID data.

On the other hand, although other backdoor detection schemes show high TPRs on identifying infected models, they also result in notably high FPR, especially under the non-IID settings where data are heavily imbalanced. The reason is that the imbalanced dataset leads to a few dominative categories with similar properties as targeted classes of an infected model, rendering them indistinguishable.In addition, their PDTs are also significantly higher than our approach because they all need to enumerate each class/neuron to iteratively reverse-engineer backdoor triggers using forward-pass and back-propagation.

TABLE IV: Performance Comparison of Backdoor Detection in Non-IID Setting with Mobilenet-V2 Architecture using CIFAR10 Dataset.

Attack	Detection	Non-IID (α)	TPR	FPR	PDT (s)
MRA	NC	0.5	96.5%	23.97%	558.5
		2.0	98.3%	61.61%	565.9
		5.0	99.5%	83.23%	582.1
	GS	0.5	95.3%	32.0%	521.3
		2.0	97.7%	65.98%	515.9
		5.0	99.1%	92.14%	512.6
	ABS	0.5	97.6%	71.57%	625.5
		2.0	99.3%	79.43%	625.7
		5.0	99.5%	90.08%	630.1
	TABOR	0.5	96.39%	34.28%	492.2
		2.0	96.8%	57.17%	492.6
		5.0	99.8%	97.03%	501.2
	P-SF	0.5	98.2%	**1.34%**	**0.0003**
		2.0	98.0%	**1.81%**	**0.0003**
		5.0	98.1%	**2.52%**	**0.0003**
DBA	NC	0.5	95.39%	23.27%	558.6
		2.0	97.7%	53.22%	559.7
		5.0	98.8%	85.83%	565.6
	GS	0.5	97.6%	46.95%	526.0
		2.0	98.3%	61.05	525.4
		5.0	99.7%	93.64%	524.4
	ABS	0.5	97.39%	39.32%	624.4
		2.0	97.89%	60.91%	620.1
		5.0	98.8%	81.87%	619.5
	TABOR	0.5	95.7%	37.8%	495.2
		2.0	98.6%	71.27%	494.9
		5.0	99.5%	94.94%	499.1
	P-SF	0.5	98.09%	**1.22%**	**0.0003**
		2.0	97.9%	**2.19%**	**0.0003**
		5.0	98.3%	**2.36%**	**0.0003**

2) **Overall Defense Performance on the Final Global Model:** In addition to the detection performance in a given round of FL training, we evaluate the overall performance of the backdoor defense by examining the final global model (i.e., the final aggregated model after the FL training). More specifically, we evaluate the ACC and ASR on the final model which has been protected using three defenses (RFA, FG, and P-SF) under the same attack (DBA) in the context of network traffic classification. We do not compare with traditional backdoor detection schemes since they are computation-extensive thus cannot be performed on all the uploaded models to complete all rounds of the FL training.

We notice that P-SF delivers the highest ACC and lowest ASR in all experimental settings (including two non-IID levels on two types of network traffic classifiers). The reason is that DBA attacks the FL system by distributing its poisonous power using multiple collaborative adversaries. It exploits the heterogeneity of the FL system to make infected models indistinguishable from the benign and biased models in individual outlier detection, and thus is more effective to attack FL with higher non-IID levels. Gradient Clip, RFA and FG cannot defend the DBA attack because they essentially try to remove outlier models/weights, leading to high ASR and low ACC on the final model. In contrast, P-SF is extremely effective in defending the DBA attack, yielding an ASR of around 2%. This is because although the poisonous power has been distributed by the DBA attack, its infected models still comprises malicious neurons to hijack the model's behavior, rendering them detectable by P-SF.

3) **Performance of P-SF on Differential Privacy (DP)-protected Models:** While we have demonstrated the P-SF's superior performance in vanilla FL, we further test its efficiency in more advanced FL systems protected by differential privacy (DP). DP is a widely used technique to protect the user's privacy by injecting random noise to prevent a malicious server or adversarial insiders from inferring the user's private data. It has been embraced for modern FL systems. We conduct an experiment to validate P-SF on an FL system protected by a widely adopted DP algorithm [51], where each client injects a certain amount of noise to its model weights before uploading the weights to the server. We perform the DP-protected FL training for payload-based network traffic classification on the ISCXVPN2016 dataset in five different non-IID levels under the DBA attack. For infected models, we perform an additional round of backdoor injection to maintain their poisonous power, aiming to infect the global model. As shown in Table VII, P-SF demonstrates robust effectiveness in backdoor detection in DP-protected models over a range of non-IID levels.

4) **Performance of Defending Adaptive Attacks:** It is critical to evaluate P-SF under adaptive attacks because an adversary may be aware the defense and thus construct more advanced attacks to evade detection. To mimic this scenario, we assume an adversary has the full knowledge of the deployed P-SF scheme including the overall algorithm and critical hyper-parameters, such as the stimulation layer index and thresholds. Recall that P-SF examines a model by stimulating a portion of the entire model, from the stimulation layer l to the output layer. Therefore, the attacker can hide the backdoor by keeping the layers after l benign by fixing their weights, and only injects the backdoor to the upstream layers. Of course, this leads to lower ASR since the backdoor is injected to fewer layers with limited capacity. To this end, an alerted server may actively reduce the number of injectable layers (thus further reduce its capacity) by selecting an l that

TABLE V: PERFORMANCE COMPARISON OF BACKDOOR DEFENSES IN NON-IID SETTING UNDER DBA

Model	Non-IID (α)	Metric	Gradient Clip [33]	Robust FA [32]	FoolsGold [31]	P-SF
Flow-based	0.5	ACC	98.11%	97.69%	97.44%	**98.66%**
		ASR	78.21%	64.51%	71.2%	**1.22%**
	2.0	ACC	95.90%	95.62%	94.93%	**96.47%**
		ASR	82.77%	83.9%	80.78%	**2.15%**
Payload-based	0.5	ACC	98.22%	98.13%	97.9%	**99.1%**
		ASR	81.12%	81.06%	76.55%	**2.08%**
	2.0	ACC	97.96%	97.42%	97.2%	**98.77%**
		ASR	87.8%	85.9%	89.48%	**1.93%**

TABLE VI: PERFORMANCE UNDER ADAPTIVE ATTACKS

Attack	Metric	Stimulation Layer Index				
		1	5	9	13	17
MRA-adaptive	PDT (s)	0.12	0.06	0.02	0.046	0.0003
	ASR	0.13%	1.1%	2.72%	32.6%	86.23%
DBA-adaptive	PDT (s)	0.32	0.21	0.13	0.045	0.0003
	ASR	0.11%	1.35%	2.52%	39.32%	89.74%

TABLE VII: PERFORMANCE COMPARISON OF POOLING-ENHANCED SCANFED ON DP-PROTECTED MODELS FOR PAYLOAD-BASED NETWORK TRAFFIC CLASSIFICATION UNDER THE DBA

Metric	Non-IID level (α)				
	0.1	0.5	2.0	5.0	10
TPR	97.9%	98.0%	98.0%	97.8%	97.8%
FPR	1.2%	1.26%	1.14%	1.34%	1.40%

is closer to the input layer to mitigate the adaptive attack. To understand how P-SF performs in this cat-and-mouse game, we conduct an experiment to compare ASR of the final global model under the adaptive attack with five different stimulation layers. More specifically, we perform the FL training on the CIFAR10 dataset using a ResNet18 model that has 18 layer, which are indexed from 1 to 18 where a smaller index indicates a layer that is closer to the input layer. We do not compare TPR and FPR in each training round because the adaptive attack does not guarantee to produce an infected local model caused by the additional regulation.

As shown in Table VI, ASR drastically decreases when a shallower stimulation layer is selected. On the other hand, PDT increases as it introduces higher computation cost to examine more layers. Therefore, a properly selected stimulation layer is required to effectively defend adaptive attacks. For example, the 9th layer is a good option as it reduces ASR to 2.7% while maintaining a small PDT.

V. CONCLUSION

We have proposed a new defense scheme, namely *ScanFed*, to protect Federated Learning (FL) from neural backdoor attacks. It leverages the synchronic nature of FL to effectively filter out malicious neuron candidates and further identify the backdoor models. ScanFed is a *plug-n-play* scheme that can be seamlessly integrated into existing FL systems. It is extremely *computation-friendly*, achieving a six-orders-of-magnitude speedup compared with the state-of-the-art backdoor defenses, rendering it most suitable for large-scale FL systems. It is *robust to biased models* uploaded by clients with non-IID (Independent and Identically Distributed) data, which is very common in practical FL systems. To the best of our knowledge, this is the first work that mitigates neural backdoor attack in non-IID settings. We have implemented a well engineered prototype in PyTorch and fully tested it on three datasets and four neural network architectures. It has been compared against seven existing backdoor defenses and evaluated under two FL backdoor attacks. The experiments have demonstrated the effectiveness of ScanFed under various settings, achieving competitive accuracy with high true positive and low false positive rates and minimal detection time.

REFERENCES

[1] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1171–1179, 2019.

[2] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[3] W. Zheng, C. Gou, L. Yan, and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *Proceedings of The Web Conference (WWW)*, pp. 13–22, 2020.

[4] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for dl-based traffic classifier update," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pp. 397–405, 2020.

[5] R. Ning, C. Xin, and H. Wu, "TrojanFlow: A neural backdoor attack to deep learning-based network traffic classifiers," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2022.

[6] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$deep-full-range$: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[7] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proceedings of the ACM*

International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc), pp. 231–240, 2018.

[8] L. Nie, D. Jiang, S. Yu, and H. Song, "Network traffic prediction based on deep belief network in wireless mesh backbone networks," in *2017 Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–5, 2017.

[9] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[10] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[11] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[12] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," in *Federated learning*, pp. 240–254, Springer, 2020.

[13] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 74–90, Springer, 2019.

[14] R. Kumar, A. A. Khan, J. Kumar, A. Zakria, N. A. Golilarz, S. Zhang, Y. Ting, C. Zheng, and W. Wang, "Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging," *IEEE Sensors Journal*, 2021.

[15] I. Dayan, H. R. Roth, A. Zhong, A. Harouni, A. Gentili, A. Z. Abidin, A. Liu, A. B. Costa, B. J. Wood, C.-S. Tsai, *et al.*, "Federated learning for predicting clinical outcomes in patients with covid-19," *Nature medicine*, vol. 27, no. 10, pp. 1735–1743, 2021.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[17] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[18] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[19] A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," *arXiv preprint arXiv:1910.00033*, 2019.

[20] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[21] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," in *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2018.

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

[23] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 2938–2948, 2020.

[24] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[25] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proceedings of IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, 2019.

[26] L. Zhu, R. Ning, C. Wang, C. Xin, and H. Wu, "GangSweep: Sweep out neural backdoors by GAN," in *Proceedings of ACM International Conference on Multimedia (MM)*, 2020.

[27] W. Guo, L. Wang, X. Xing, M. Du, and D. Song, "Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems," *arXiv preprint arXiv:1908.01763*, 2019.

[28] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1265–1282, 2019.

[29] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang, "Practical detection of trojan neural networks: Data-limited and data-free cases," *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[30] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[31] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

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

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

[34] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *arXiv preprint arXiv:2007.05084*, 2020.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

[36] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[37] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning gan for pose-invariant face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1415–1424, 2017.

[38] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[39] F. R. Hampel, "The influence curve and its role in robust estimation," *Journal of the american statistical association*, vol. 69, no. 346, pp. 383–393, 1974.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015.

[41] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, pp. 972–981, 2017.

[42] R. W. Farebrother, *Fitting linear relationships: A history of the calculus of observations 1750-1900.* Springer Science & Business Media, 1999.

[43] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[44] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

[45] T. Minka, "Estimating a dirichlet distribution," 2000.

[46] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407–414, 2016.

[47] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[49] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[50] A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," *arXiv preprint arXiv:2010.08138*, 2020.

[51] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," *arXiv preprint arXiv:1710.06963*, 2017.