Tutorial: Toward Robust Deep Learning against Poisoning Attacks

HUILI CHEN and FARINAZ KOUSHANFAR, University of California, San Diego, USA

Deep Learning (DL) has been increasingly deployed in various real-world applications due to its unprecedented performance and automated capability of learning hidden representations. While DL can achieve high task performance, the training process of a DL model is both time- and resource-consuming. Therefore, current supply chains of the DL models assume the customers obtain pre-trained Deep Neural Networks (DNNs) from the third-party providers that have sufficient computing power. In the *centralized* setting, the model designer trains the DL model using the local dataset. However, the collected training data may contain erroneous or poisoned data points. The model designer might craft malicious training samples and inject a backdoor in the DL model distributed to the users. As a result, the user's model will malfunction. In the *federated learning* setting, the cloud server aggregates local models trained on individual local datasets and updates the global model. In this scenario, the local client could poison the local training set and/or arbitrarily manipulate the local update. If the cloud server incorporates the malicious local gradients in model aggregation, the resulting global model will have degraded performance or backdoor behaviors. In this article, we present a comprehensive overview of contemporary data poisoning and model poisoning attacks against DL models in both centralized and federated learning scenarios. In addition, we review existing detection and defense techniques against various poisoning attacks.

CCS Concepts: • **Computing methodologies → Machine learning**; *Distributed computing methodologies*; • **Security and privacy → Software and application security**; *Systems security;*

Additional Key Words and Phrases: Neural networks, federated learning, robustness, poisoning attacks

ACM Reference format:
Huili Chen and Farinaz Koushanfar. 2023. Tutorial: Toward Robust Deep Learning against Poisoning Attacks. *ACM Trans. Embedd. Comput. Syst.* 22, 3, Article 42 (April 2023), 15 pages.
https://doi.org/10.1145/3574159

1 INTRODUCTION

Deep Learning (**DL**) has enabled the paradigm shift in various application fields such as autonomous driving, medical diagnosis, natural language processing, and healthcare monitoring [14, 32, 34]. Although DL models, particularly **Deep Neural Networks** (**DNNs**), can achieve high accuracy on diverse tasks and removes the need for manual engineering efforts, training a high-performance DNN is both time- and resource-consuming. As such, DNNs are typically

This work was supported by grants NSF-CNS under Award No. 2016737, NSF TILOS AI Institute under Award No. 2112665, Intel PrivateAI Collaborative Research Institute, and ARO MURI Grant No. W911NF-20-S-0009.

Authors' address: H. Chen and F. Koushanfar, University of California, San Diego, 9500 Gilman Drive, La Jolla, California 92093; emails: {huc044, farinaz}@ucsd.edu.

1539-9087/2023/04-ART42 $15.00
https://doi.org/10.1145/3574159

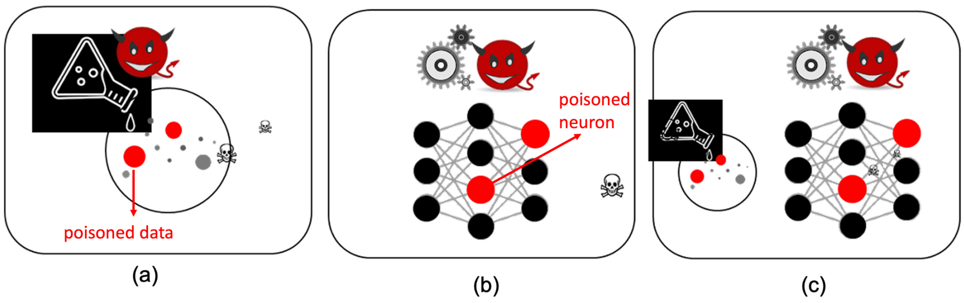

Fig. 1. Demonstration of various poisoning attacks against machine learning models. (a) Data poisoning attack; (b) model poisoning attack; (c) combination of data and model poisoning attack.

trained by third-party tech companies [50] such as Amazon, Microsoft, and Meta that have sufficient computing power. These pre-trained DL models are then distributed to end users or deployed on the cloud, i.e., **Machine Learning as a Service** (**MLaaS**) [2, 31]. However, this supply chain does not guarantee the security/trustworthiness of the provided DL model. Particularly, a malicious model provider may inject stealthy backdoors into the model by data poisoning attacks [12, 28]. Besides model training in the centralized setting, **Federated Learning** (**FL**) is also a popular learning paradigm that trains a global model by aggregating knowledge from local models that are trained on individual local datasets [5, 26, 48]. Prior works have identified that global models obtained via FL are also susceptible to poisoning attacks [3, 43, 46].

In the centralized learning setting, the model developer has full access to the training dataset and sufficient computing power. It is worth noticing that the training data might be provided by a separate party or collected from public resources. This means that the training set may contain "poisoned" or erroneous data points, resulting in *data poisoning attacks* [1, 19, 25, 28]. In the FL scenario, edge users train local models using their private local datasets and upload their local gradients to the cloud server. The server then aggregates all local models and updates the global model for the next round. In this training paradigm, malicious clients may craft adversarial local updates to divert the aggregated global model. This type of attack is called *model poisoning attack* [3, 16, 41, 43]. In a wider sense, the adversary can launch model poisoning attacks by performing *fault injection* into model parameters [6, 29]. These software-level faults can be injected via laser beams [15, 39] or Row-Hammer attacks [10, 21, 35]. Note that both data poisoning attacks and model poisoning attacks can be untargeted (degrading the model's performance on all samples) or targeted (misleading the model to predict the specific class). Figure 1 visualizes three types of poisoning attacks against DL models.

Our Contributions. In this tutorial article, we provide a comprehensive overview of contemporary poisoning attacks against DL systems. More specifically, our taxonomy and systematization of knowledge includes potential threats in all stages throughout the lifecycle of the DL system, from data collection, model training, to deployment in the field. Furthermore, we discuss poisoning attacks and the corresponding defense schemes in both centralized and FL scenarios. Our comprehensive and structured overview of defense techniques against poisoning attacks facilitates the development of robust DL systems as well as future research on this topic.

This article is organized as follows. In Section 2, we introduce existing untargeted and targeted poisoning attacks in the centralized setting. In Section 3, we discuss data and poisoning attacks in the FL scenario. Section 4 and Section 5 categorize detection/defense methods against poisoning attacks. Section 6 concludes the article.

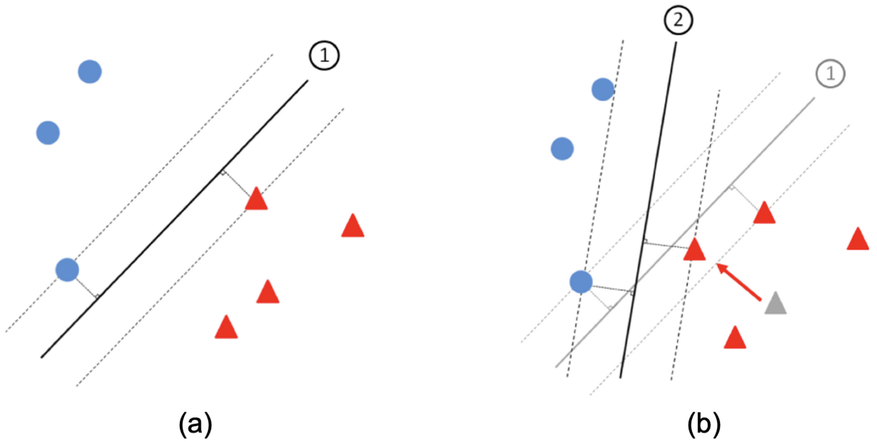

Fig. 2. Illustration of data poisoning attacks on a linear SVM classifier with two classes [22]. (a) Original decision boundary with clean training set. (b) Modified decision boundary when one training sample is changed.

2 POISONING ATTACKS IN CENTRALIZED LEARNING

2.1 Data Poisoning Attacks

Data poisoning attacks exploit the intuition that training samples impact the decision boundary of the resulting DL model. We illustrate an example of data poisoning attacks against a linear **Support Vector Machine** (**SVM**) model in Figure 2. For this two-class classification problem, the decision boundary (solid line) and the margin (dashed line) of the benign SVM are shown in Figure 2(a). When one of the training samples is poisoned (as denoted by the gray triangle) in Figure 2(b), the decision boundary of the resulting model is different from the benign model. While DL models have more complex decision boundaries compared to linear SVMs, they are also susceptible to data poisoning attacks as demonstrated in the prior works [19, 28]. In this section, we introduce existing data poisoning attacks with untargeted and targeted attack objectives.

2.1.1 Untargeted Attacks. In the centralized learning setting, untargeted data poisoning attacks aim to degrade the performance of the trained model on clean samples regardless of the output incorrect class (i.e., reducing the *overall* task accuracy). This attack objective is analogous to untargeted adversarial samples [24, 45]. We introduce the working mechanisms of several data poisoning attacks below.

BadNets. The paper [19] demonstrates an untargeted backdoor attack using two types of trigger design: single-pixel and pattern-based triggers. Particularly, the poisoned images are crafted by adding the single-pixel/trigger pattern to the clean images and change the ground-truth labels to incorrect ones. The adversary randomly selects p percent of training samples and designs poisoned samples correspondingly. For *all-to-all* attacks, the label is changed from i to $i+1$. For *random target attacks*, the adversary changes the label of the poisoned sample to a random incorrect label. The poisoned model is obtained by fine-tuning the benign model using a mixture of clean images and poisoned ones. Experimental results show that BadNets can reduce the classification accuracy of the trained RCNN model on poisoned images to 1.3% when a yellow square is used as the trigger pattern.

Data poisoning against online learning. The paper [44] considers data poisoning in the online learning scenario where the input data comes in a streaming manner. The authors assume the adversary has white-box access to the model and can arbitrarily alter at most K samples in the input data stream. The paper [44] selects **Online Gradient Descent** (**OGD**) algorithm and formulates data poisoning attacks against online learning as an optimization problem for both semi-online and fully online settings. The authors propose a method to quantify the impact of modifying a specific input sample on the attack objective function (classification error in the paper). Since the adversary can perturb up to K samples, the authors present three solutions to strategical searching for proper poisoning positions in the input stream: incremental attack, interval attack, as well as teach-and-reinforce attack. The paper [44] evaluates attack performance on synthetic 2D Gaussian mixture, MNIST, fashion MNIST, and UCI Spambase datasets. Empirical results show that, in the semi-online setting, incremental and interval attacks are overall the most effective, while teach-and-reinforce is more effective in the fully online scenario.

GAN-based poisoning data generation. The paper [47] first evaluates the *gradient-based* method to generate poisoned data against the given victim DNN. Then, the authors propose a new *generative method* to accelerate the generation speed of the poisoned data. More specifically, the paper [47] suggests deploying the victim DNN as the fixed *discriminator* and training a *generator* (an auto-encoder) to update the poisoned data to minimize the attack loss computed by the discriminator. The authors perform experiments on MNIST and CIFAR10 datasets. Empirical results show that while the gradient-based poisoning data generation method is more effective to reduce the overall accuracy of the victim model, the proposed GAN-based method can achieve a similar level of attack performance and improve the generation speed by up to two orders of magnitude.

2.1.2 Targeted Attacks. The goal of targeted poisoning attacks is to divert the prediction of the victim model on the backdoored inputs to the specific attack target class. As such, targeted data poisoning attacks are stealthier compared to the untargeted variant since the poisoned model has normal accuracy on clean inputs. We introduce several examples of targeted data poisoning attacks below.

TrojanNN. The paper [28] proposes a Neural Trojan attack against DNNs with three main stages: Trojan trigger generation, training data generation, and model retraining. The threat model assumes that the adversary does not know the original training set but has white-box access to the victim model. Particularly, the Trojan trigger is generated by updating the value assignments of the pixels within the trigger mask to maximize the activation values of pre-selected neurons. Model inversion is then performed to recover approximate training data. The Trojan is inserted into the model by partial re-training (i.e., only training the layers between the selected neurons and the last layer). Empirical results on computer vision and speech recognition tasks show that TrojanNN can achieve a nearly 100% Trojan success rate on backdoored inputs.

Targeted backdoor attack via data poisoning. The paper [11] presents a black-box backdoor attack with a weak threat model. Particularly, the adversary does not know the internal details of the victim model or its training set. The attacker can only perturb a small number of input samples for data poisoning and the changes shall be hard to notice. The authors of [11] propose two types of backdoor poisoning attacks: input-instance-key attacks and pattern-key attacks. The main difference between these two attack variants is the activation condition (or trigger design) of the backdoor attack. Evaluation results on DeepID and VGGFace datasets show that the poisoned model predicts the attack target class on the backdoor instances with over 90% probability.

Targeted clean-label poisoning attacks. This paper [40] proposes a new type of backdoor attack called clean-label poisoning attacks. Compared to previous backdoor attacks [11, 19, 28], the work [40] does not assume the adversary can control the labels of training data. Given a *base* *instance b* from the correct class and a *target instance t* from the attack target class, the proposed clean-label attack uses an optimization approach to generate the poisoned instance *p* that is close to *b* but produces a similar activation map as *t* in the penultimate layer. This activation alignment in the penultimate layer is called *feature collision* in the paper [40]. This optimization problem is solved using a forward-backward-splitting iterative procedure. Experimental results on the CIFAR10 dataset show that the proposed clean-label attack can achieve a 100% attack success rate with a single poison instance on transfer learning tasks. To embed multiple poison instances, the paper [40] uses a "watermarking" approach. The proposed watermarking-based poisoning attack achieves a 60%–70% success rate in the end-to-end training scenario.

2.2 Model Poisoning Attacks

In this section, we introduce model poisoning attacks that do not rely on model re-training to embed the backdoor. Instead, the backdoor is inserted by modifying a few critical parameters in model weights. We discuss several *fault injection*–based model poisoning attacks below.

We first introduce untargeted model poisoning attacks as follows.

Terminal brain damage. The paper [21] demonstrates the first untargeted **Bit-Flip Attack** (**BFA**) on floating-point DNNs. The paper uses heuristics to find vulnerable weight parameter bits to modify for degrading the overall classification accuracy of the model. The adversary estimates the effectiveness of modifying a specific weight bit for accuracy degradation objective using the accuracy of the poisoned model on a sample validation dataset. In addition, the authors of [21] investigate two different attack scenarios: *surgical* BFA where the attacker can induce a bit flip at the desired location in the victim model's memory, and *blind* BFA where the attacker cannot precisely control the bit-flip position in model weights.

Neural network weight attack. While the paper [21] observes that the exponential bits of the floating-point DNNs are suitable for bit-flip attacks, fixed-pointed DNNs are more widely used in practice due to their efficiency and reduction of storage size. The paper [36] proposes an untargeted BFA on *fixed-pointed* DNNs by searching weight bits with large gradient magnitudes in an iterative in-layer and cross-layers way. Particularly, the in-layer bit search finds the most vulnerable weight bits in this layer via gradient ranking. Then, the cross-layer bit search computes the attack loss values when vulnerable bits in each layer are modified independently. The layer that achieves the largest attack loss is selected as the attack layer and its corresponding vulnerable bits are flipped in this attack iteration.

In the remainder of this section, we discuss model poisoning attacks with a targeted attack objective.

Fault injection attack. The paper [29] investigates two variants of fault injection: **Single-Bias Attack** (**SBA**) and **Gradient-Descent Attack** (**GDA**). SBA only modifies a single-bias parameter in the DNN to enlarge the corresponding bias in the output layer associated with the adversarial class. GDA achieves high stealthiness by preserving the accuracy of the poisoned model on clean samples. Particularly, GDA first performs a layerwise search and finds a subset of parameters to update via gradient descent. Then, the modification parameter set is compressed by repeatedly setting the smallest element to zero to reduce the overall perturbation.

TBT. Targeted Bit Trojan [37] proposes a bi-flip–based neural Trojan attack against **Quantized Neural Networks** (**QNNs**). TBT first deploys **Neural Gradient Ranking** (**NGR**) to identify the vulnerable neurons in the last layer of the victim model associated with the attack target class. Then, the Trojan trigger is generated via gradient descent to stimulate the identified significant neurons to the pre-defined large constant value. In the last step, TBT locates vulnerable bits of

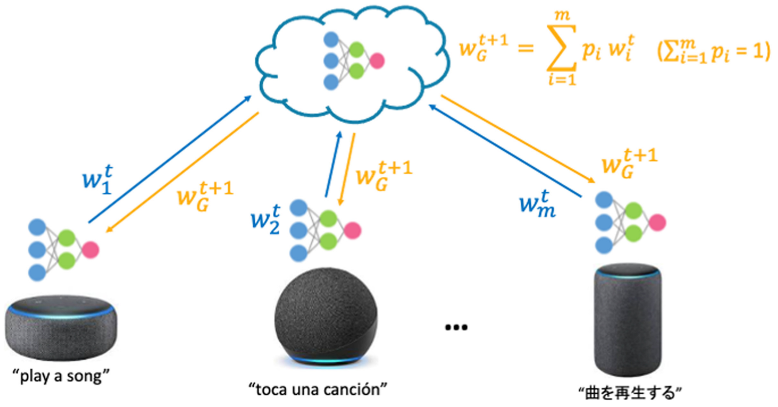

Fig. 3. Demonstration of a FL system for speech recognition tasks.

DNN weight through **Trojan Bit Search** (**TBS**). With the identified significant neurons in the last layer, TBS uses **Stochastic Gradient Descent** (**SGD**) to update the partial weight parameters corresponding to the significant neurons in the last layer with the goal of Trojan insertion and maintaining accuracy on clean data. The vulnerable bits are identified after this partial re-training.

ProFlip. The paper [10] proposes a progressive bit-flip–based Trojan attack against QNNs. There are three key differences between ProFlip [10] and TBT [37]: (i) Trigger generation. The trigger generation of TBT requires the attacker to manually set a hyper-parameter w_b (number of neurons changed). The success rate of TBT is sensitive to w_b while it's not clear how to select w_b. ProFlip removes the manual effort of specifying w_b by presenting a novel adversarial saliency map-based trigger generation method. (ii) ProFlip [10] proposes a new parameter sensitivity analysis method for attack parameter selection. In contrast, TBT always attacks the weight of the last layer, which is not necessarily the optimal attack parameter. (iii) ProFlip proposes a novel progressive critical bit search technique. TBT's bit search only updates a pre-defined number of elements (w_b) in the last layer using partial re-training. Compared to TBT, the critical bit search of ProFlip locates the vulnerable weight bits by first identifying the attack parameter via sensitivity analysis and then selecting the attack element with the largest impact on the attack objective. As a result, ProFlip [10] demonstrates higher attack success rates with fewer bit flips compared to TBT [37].

3 POISONING ATTACKS IN FL

We introduced various data and model poisoning attacks against centralized DNNs in Section 2. In this section, we focus on poisoning attacks in the FL scenario where the global model residing on the cloud server is the victim DNN. Figure 3 shows the workflow of typical FL systems. In round t, edge devices train their local models w_i^t using their local datasets and upload the updated local models to the cloud server. The server then performs model aggregation to update the global model w_G^{t+1}. This global model is then distributed to the selected clients for FL training in the next round. We categorize the FL poisoning attacks into two types based on the attack objective below.

3.1 Byzantine Attacks in FL

Untargeted poisoning attacks can be performed in the FL setting. Byzantine attacks assume that one/multiple clients in the FL system are malicious and upload adversarial local updates to degrade the overall performance of the global model. We discuss three existing Byzantine attacks below.

Local model poisoning attacks to Byzantine-robust FL. The paper [16] performs a systematic study of local model poisoning attacks with the goal of inducing a large test error for the global model. The threat model assumes that the adversary controls several devices in the FL system (less than 50% of total devices) and has full knowledge of the local datasets, local models, as well as training algorithms. Particularly, the authors craft malicious local models by formulating an optimization problem that aims to deviate a global model parameter toward the opposite direction of the global model parameter change without attacks (i.e., *directed deviation goal* [16]). Experimental results on MNIST, Fashion MNIST, CH-MNIST, and Breast Cancer Wisconsin (Diagnostic) show that the proposed local model poisoning attacks can defeat previous Byzantine-robust FL systems that rely on Krum [4], Bulyan [20], trimmed mean, or median aggregation rule [49].

Manipulating the Byzantine. The paper [41] develops a general model poisoning attack framework on FL that outperforms existing Byzantine attacks. The proposed attack considers distinct attack scenarios depending on two factors: the knowledge of local updates from benign clients, and the knowledge of the aggregation rule used by the server. The model poisoning attack works as follows. The adversary first estimates the reference benign aggregate using the knowledge of some benign local updates. Then, he/she generates a malicious perturbation for his/her local model with the goal of maximally diverting the estimated benign aggregate in the malicious direction while maintaining stealthy. Empirical results show that the proposed model poisoning attack achieves 1.5× to 60× more accuracy reduction for the global model compared to the state-of-the-art counterpart [41].

Federated Multi-Armed Bandits Under Byzantine Attacks. The authors of [13] study Byzantine attacks in the **Federated Multi-Armed Bandits (FMAB)** problem. In an FMAB system, a cohort of learners holds heterogeneous local models and plays a multi-armed bandits game. The learners exchange their aggregated feedback with the parameter server in order to learn a global feedback model [13]. The paper proves that if the majority of learners in the cohort are benign, it is feasible to induce a sufficient large cumulative regret with respect to an unavoidable error margin. To mitigate the threats of Byzantine clients, the paper [13] adapts robust statistics and presents a defense named Fed-MoM-UCB using a median-of-means–based estimator.

3.2 Backdoor Attacks in FL

Backdoor attacks in the FL setting also aim to direct the behavior of the global model to a specific attack target when the trigger is present in the input data. We discuss several FL backdoor attacks below.

How to backdoor FL. The paper [3] studies targeted model poisoning attacks against FL systems where the malicious participants use *model replacement* to insert the backdoor payload into the aggregated global model. The goal of the proposed backdoor FL attack is to mislead the global model to predict the attack target class on inputs with trigger patterns.

The threat model assumes that the adversary can adaptively control local training methods and configurations. To achieve targeted backdoor attacks, the attacker (compromised local client) performs model replacement by estimating the poisoned local model when the global model is close to convergence. In addition, the paper [3] proposes two variants of backdoor attacks: *constrain-and-scale* and *train-and-scale*. The first variant uses regularized local loss function to improve the stealthiness of the attack. The second variant estimates the scale factor of the local poisoned model while enforcing an upper bound on the local model magnitude. Experimental results on CIFAR10 and Reddit datasets (word prediction) show that the proposed model replacement-based backdoor attack outperforms training-data poisoning-based backdoor attack.

Attack of tails. The paper [43] presents a new attack called *edge-case backdoors*. The edge-case backdoor aims to divert the global model to misclassify rare inputs (i.e., inputs residing on the tail of input data distribution). In addition, the authors of [43] theoretically show that backdoor attacks are unavoidable if the model is vulnerable to adversarial attacks. The paper proposes three attack methods based on different threat models: black-box attacks, **Projected Gradient Descent** (**PGD**) attacks, and PGD attacks with model replacement. The adversary is assumed to have a set of clean samples and a candidate set of edge-case samples. To characterize the probability distribution of activations in the penultimate layer, the paper [43] uses a **Gaussian Mixture Model** (**GMM**) where the number of clusters is the same as the number of output classes. The GMM (a generative model) is then fitted to the activation map obtained by feeding the benign samples to the victim DNN. This trained GMM is used to identify out-of-distribution samples and construct the set of p edge-case backdoor samples. Experimental results show that the crafted edge-case backdoors can result in over 80% backdoor success rate and impairs the fairness of the defense techniques against edge-case backdoor samples.

Distributed backdoor attacks. DBA [46] presents a new backdoor threat attack against FL systems. The goal of DBA is to poison the shared model on the cloud server with the defined *global trigger*. To this end, DBA decomposes the global trigger into multiple parts as the *local triggers*. The adversarial clients then use the local triggers to design local poisoned models and collaboratively embed the ultimate backdoor into the global model. Empirical results on various datasets and model architectures show that the proposed DBA achieves higher backdoor success rates, faster FL convergence, and better robustness in both single-shot and multiple-shot attack scenarios.

4 POISONING ATTACK DETECTION IN CENTRALIZED LEARNING

We introduced data and model poisoning attacks against centralized DL models in Section 2. In this section, we discuss existing detection methods and categorize them into two types: model-level detection (which aims to inspect whether a trained model has been backdoored) and data-level detection (which aims to determine whether the input sample contains the backdoor trigger).

4.1 Model-Level Detection

In this section, we discuss model-level detection techniques against poisoning attacks. The goal of model-level detection is to ensure the safety and trustworthiness of pre-trained DL models.

Neural Cleanse. The paper [42] proposes a backdoor detection technique using trigger reconstruction. The authors explore the intuition that data poisoning attacks modify the decision boundary of the model via injecting mislabeled data points. The adversary is assumed to have a set of clean samples and white-box access to the pertinent model. Neural Cleanse reverse engineers the hidden triggers by formulating an optimization problem for the backdoor objective and deploys gradient descent to update the trigger. The model backdoor is detected using robust statistics. Particularly, the trigger is recovered for each possible output class and the magnitude (L_1 norm) of the triggers is used as the test statistics. The trained model is determined to be backdoored if there is an outlier detected in the trigger magnitude statistics. Besides backdoor detection, Neural Cleanse also suggests two backdoor mitigation methods: *proactive filtering* and *neuron pruning*. The first defense detects and removes adversarial input samples, and the second defense variant assigns zero output values to the backdoor-related neurons.

ABS. The authors of [27] design a model-level backdoor detection method by **Artificial Brain Stimulation** (**ABS**). The proposed detection technique studies how the output activations change when different levels of stimulation are introduced to hidden neurons. Compromised neurons are identified if the neurons can substantially raise the output activation of a specific class on arbitrary inputs. Then, the neuron is confirmed to be compromised using the reverse-engineered Trojan trigger as guidance. The model is decided to be backdoored if a trigger can be consistently generated to divert the model's predictions to a certain label.

DeepInspect. This paper [9] proposes the first black-box backdoor detection and mitigation technique. Unlike Neural Cleanse, DeepInspect does not make the assumption about benign samples or full access to the model under test. Instead, DeepInspect first uses model inversion to construct a substitution training set and explores a **conditional Generative Adversarial Network** (**cGAN**) to recover potential triggers (for each output class) used by the adversary. The authors explore the observation that the backdoor trigger acts as a "shortcut" across the decision boundary and uses the magnitude of triggers recovered by cGAN as the test statistics. To detect backdoor existence, DeepInspect employs hypothesis testing via **Median Absolute Deviation** (**MAD**). If an outlier is present in the left tail of trigger magnitude statistics, then the model is determined to be backdoored. In addition, DeepInspect proposes a model patching defense to mitigate backdoor threats. The defender can use the trained cGAN to generate perturbed inputs with correct labels and retrain the victim model for robustness enhancement. Experimental results show that DeepInspect attains better detection performance compared to Neural Cleanse [42].

As a summary of the above discussion, Neural Cleanse [42] is effective in detecting Trojaned models given white-box access to the victim model while it may incur high overhead on large DL benchmarks. DeepInspect [9] achieves black-box Trojan detection and mitigation using conditional GAN, demonstrating wider applicability and better scalability compared to Neural Cleanse. ABS [27] analyzes the model's response to inner neurons and requires much fewer input samples for each output label compared to Neural Cleanse.

4.2 Data-Level Detection

In this section, we introduce existing works that aim to detect whether the input sample contains the backdoor trigger for the given DNN. Note that this data-level detection is orthogonal to model-level detection discussed in Section 4.1.

Activation clustering. The paper [8] proposes a poisonous data detection method to prevent backdoor insertion. The intuition of the **Activation Clustering** (**AC**)–based defense [8] is the difference in prediction reasons for inputs in the attack target class. For benign samples in the target class, the model makes the correct prediction since it learns the relevant features. However, for poisonous samples, the model classifies them into the target class since the model "memorizes" the backdoor triggers during training. Therefore, AC obtains the activation maps of the model in the last hidden layer when untrusted samples are fed as inputs. Then, the activations are vectorized and **Independent Component Analysis** (**ICA**) is used for dimensionality reduction. Finally, k-means with $k = 2$ is used to cluster the reduced activations. Experimental results show that AC can achieve close to 100% detection rate of F1 score on MNIST and LISA datasets.

STRIP. The paper [18] designs a backdoored inputs detection approach based on **STRong Intentional Perturbation** (**STRIP**). The key idea of the defense is that poisoned inputs are more robust to perturbations. More specifically, when different perturbing patterns are added to a backdoored sample, the prediction of the Trojaned model remains unchanged (i.e., outputs the attack target class). When perturbations are added to clean samples, the model predictions have a large variance. To leverage this intuition, STRIP proposes an entropy metric to characterize the randomness of model predictions. With this entropy measure, a backdoored input is detected if it yields a low output entropy. STRIP [18] achieves less than 1% false acceptance rates and false rejection rates on MNIST, CIFAR10, and GTSRB datasets.

CLEANN. The paper [23] presents a hardware-assisted solution to online backdoor mitigation. CLEANN defense has two key components: a **Discrete Cosine Transforma (DCT)** analyzer and a feature analyzer. The DCT module extracts the frequency features of input samples and identifies suspicious frequency components abnormal in the clean data. To this end, sparse approximation is applied to the DCT transformation of inputs, and concentration inequality-based outlier detection is used to detect anomalous reconstruction errors [23]. The feature analyzer resides in the penultimate layer of the victim model and inspects suspicious latent features. Within the feature analyzer, dimensionality detection and sparse recovery are performed to detect poisoned inputs. Particularly, the sparse recovery denoises input features and performs anomaly detection based on the reconstruction errors. In addition, CLEANN develops matrix-vector multiplication cores and sparse recovery cores to accelerate Trojan detection of input samples. Experimental results on MNIST, GTSRB, and VGGFace datasets prove that CLEANN can effectively reduce the backdoor success rates and recover the ground-truth labels of the poisoned inputs.

Among the above-mentioned data-level detection techniques [8, 18, 23], CLEANN [23] is the only framework that is equipped with hardware optimizations and demonstration of real-world deployments. The customized acceleration of matrix-vector multiplication and sparse recovery in CLEANN makes it very lightweight and fast for input data checks.

Discussion. We give an overview of model-level and data-level detection of poisoning attacks in this section. The threat models are distinct for these two lines of works. Particularly, data-level detection techniques [8, 18, 23] aim to check if any incoming input data contains the backdoor trigger, which makes an *implicit assumption* that the model under test has been Trojaned during training. While for model-level detection methods [9, 27, 42], the defender does not make this assumption. Instead, model-level detection aims to inspect whether the queried model has been Trojaned or not while the defender does not know the trigger pattern beforehand. We categorize existing works into these two types and show how to characterize model response to input and examine model internals for each type.

5 POISONING ATTACK DETECTION IN FL

We introduce various data and model poisoning attacks (targeted or untargeted) against FL systems in Section 3. In this section, we discuss contemporary detection methods of Byzantine attacks and backdoor attacks in the context of FL.

5.1 Byzantine-Resilient FL

Recall that Byzantine attacks aim to degrade the overall performance of the global model on the test set. The threat model assumes that the compromised devices collaborate and upload malicious local updates to the server and disturb the learning of the global model. We discuss existing detection techniques of FL Byzantine attacks below.

Krum. The paper [4] theoretically shows that any linear combination-based aggregation rules cannot tolerate a single Byzantine participant. To improve the robustness against Byzantine attacks, the paper [4] proposes a Byzantine-resilient aggregation rule named *Krum* that satisfies the resilience property for distributed SGD. Krum combines *majority-based* and *squared distance-based* methods to find the local update that has the smallest distance to its closest $(n-f)$ neighbors (n and f are the number of all workers and malicious works, respectively). Krum guarantees Byzantine resiliency if the condition $2f + 2 < n$ holds. As an extension, the authors of [4] also propose *Multi-Krum* which combines Krum and federated model averaging. As such, Multi-Krum improves FL convergence compared to Krum.

AttestedFL. The paper [30] proposes an untargeted poisoning attack detection technique in FL via behavior attestation. The threat model does not enforce any upper bounds on the ratio of Byzantine workers. The goal of the defense is to assess whether a local worker is trustworthy by observing its training performance across iterations (i.e., maintaining the states). To this end, AttestedFL incorporates three aspects of defense: the first line monitors the convergence local model toward the global one; the second line inspects the angular distance of consecutive local updates of a client during FL training; the third line filters out the suspicious workers if the performance of their local updates does not improve on a quasi-validation dataset compared to the previous round. AttestedFL is shown to be effective in detecting untargeted poisoning attacks.

Coordinatewise median or trimmed mean. The paper [49] develops a provably robust distributed learning algorithm against Byzantine attacks and intends to achieve statistically optimal performance. Particularly, the work [49] proposes two variants of robust distributed gradient descent algorithms based on *coordinatewise median* and *coordinatewise trimmed mean* for global aggregation on the server side. Furthermore, the authors theoretically analyze the statistical error rates of these two Byzantine-robust FL algorithms in different loss scenarios: strongly convex, non-strongly convex, and non-convex loss functions. Particularly, in the case of strongly convex quadratic loss, a median-based one-round algorithm is proved to attain order-optimal statistical rates [49].

FLTrust. The paper [7] proposes a Byzantine-robust FL algorithm based on trust bootstrapping on the server [7]. The cloud server is assumed to have access to a small set of clean samples relevant to the learning task (*root dataset*). Also, the server maintains a *server model* as the basis to bootstrap trust. More specifically, in each round, the cloud server computes a trust score for each local update based on the angular deviation of the local model and the global one. A larger deviation of model update directions corresponds to a smaller trust score. The server then normalizes the magnitudes of all local updates. In the last step, the server performs model aggregation by trust score-weighted model averaging and updates the global model.

Evaluation results on MNIST, Fashion MNIST, CIFAR10, and **Human Activity Recognition** (**HAR**) datasets show that FLTrust is robust against existing poisoning attacks and adaptive attacks with less than 100 samples in the root dataset.

To summarize, Krum [4] and Coordinatewise median/trimmed mean [49] leverage robust statistics to design Byzantine-resilient FL systems. The statistical view of FL training allows them to derive a theoretical guarantee on the effectiveness of the defense scheme. AttestedFL [30] and FLTrust [7] propose to build a "trusted anchor" as the benign reference for Byzantine robustness. However, this type of approaches are heuristic-based and cannot provide guaranteed robustness. In addition, the construction of the trusted anchor requires a "clean" dataset for the pertinent task, which may not be feasible for data-sensitive tasks.

5.2 Backdoor Detection in FL

In this section, we discuss contemporary backdoor defense techniques in the context of FL.

FLAME. The paper [33] presents FLAME, a backdoor-resilient aggregation framework that mitigates the impact of potential backdoored local updates while preserving the performance on benign inputs. The FLAME design has three key components: *model clustering*, dynamic weight clipping, and **differential privacy** (**DP**)–based adaptive noising. The first module deploys HDB-SCAN clustering to identify the majority cluster of local updates and filter out suspicious ones. The second module computes the median of local update magnitudes and uses it as a dynamic threshold to clip the uploaded local weights. The third module injects random Gaussian noise into the aggregated global model where the noise variance is adaptively tuned based on the statistics of local updates. It is worth noticing that the first two components of FLAME help to reduce the amount of DP noise required to eliminate the backdoor influence on the global model. Evaluation results on various data modalities and DNN architectures prove the effectiveness of FLAME.

FoolsGold. This work [17] develops a backdoor defense named FoolsGold that identifies poisoned participants based on the diversity of local updates in FL. The threat model of FoolsGold is different from prior works since it does not assume an upper bound on the number of compromised clients and does not require auxiliary information about client data. Given the received local updates from clients, FoolsGold adaptively assigns the aggregation weights to different participants based on the distribution of their update directions. More specifically, FoolsGold computes the pairwise *update similarity* based on the aggregate historical vector and the indicative features. The intuition of FoolsGold is that the angle between the aggregated update vectors of backdoor participants is smaller than the angle between honest clients and backdoored clients. This similarity-aware model aggregation can mitigate the adversarial influence of backdoor adversaries on the global model.

DeepSight. The paper [38] proposes an FL backdoor detection method based on model inspection and weight clipping. DeepSight develops a *voting-based model filtering* technique that combines clustering-based estimation of model similarity and a meta classifier. A new metric called *threshold exceedings* is introduced to characterize the parameter update of the last layer and measure the homogeneity of training data. The ensemble of clustering algorithms is used to group local updates with similar training data distribution. Furthermore, DeepSight proposes two new methods, **Division Differences** (**DDifs**) and **NormalizEd UPdate energies** (**NEUPs**), to quantify the fine-grained differences between the internals/outputs of multiple DNNs.

In summary, FoolsGold [17] uses an adaptive aggregation rule based on angular similarity to defend against FL backdoor attacks, while it might be ineffective when detecting adversaries that deliberately hide their angular deviation. DeepSight [38] designs new metrics and trains a meta-classifier to detect malicious local updates, while FLAME [33] combines differential privacy and angular/magnitude footprints of backdoored models for attack mitigation. As such, FLAME demonstrates stronger defense compared to DeepSight thanks to the integration of DP-based adaptive noising.

Discussion. We survey existing poisoning defense in the FL scenario in the above discussion. Compared to centralized defense in Section 4, protecting DL models in the distributed setting is more challenging since the defender needs to analyze multiple local models for identifying malicious updates from edge users. Furthermore, there might be multiple adversaries collaboratively attacking the model, making the detection of poisoned updates more difficult in the FL setting. The defense overview in Section 5 demonstrates how the defender can leverage techniques such as robust statistics, assistance of trusted anchors, clustering analysis, and/or adaptive noising to detect backdoor attacks in FL.

6 CONCLUSION

DL models are widely used in various safety-critical applications due to their superior performance and capability of learning features automatically. In this article, we present a comprehensive study of contemporary poisoning attacks and defense techniques. Our taxonomy covers attacks and corresponding defenses in different settings (i.e., centralized and FL scenarios), as well as in diverse stages of DL applications (i.e., data collection, training, and inference). We also discuss poisoning attacks with different attack objectives. Our systematization of knowledge sheds light on the vulnerability of DL models in the entire lifecycle. This tutorial can facilitate the development of robust DL systems and risk assessment of DL models for both research communities and industrial practitioners. For future works, we believe a holistic, end-to-end defense framework is critical to protect the DL system throughout its lifecycle. For this purpose, the defender needs to take an algorithm/software/hardware co-design approach for developing an effective and lightweight defense solution that supports real-time detection of poisoning attacks. Particularly, the defense scheme shall be implemented with hardware optimizations when deployed in the field to ensure minimal defense overhead and applicability on resource-constrained systems.

REFERENCES

[1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. 2016. Data poisoning attacks against autoregressive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[2] Amazon. 2022. Amazon Amazon Web Services (AWS). https://aws.amazon.com/. Accessed: 2022-06-02.

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

[4] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems* 30 (2017), 118–128.

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems* 1 (2019), 374–388.

[6] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. 2018. Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2204–2206.

[7] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. 2020. Fltrust: Byzantine-robust federated learning via trust bootstrapping. arXiv preprint arXiv:2012.13995 (2020). https://arxiv.org/pdf/2012.13995.pdf.

[8] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2018. Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728 (2018). https://arxiv.org/pdf/1811.03728.pdf.

[9] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2019. DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks. In *IJCAI*, Vol. 2. 8.

[10] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2021. Proflip: Targeted trojan attack with progressive bit flips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7718–7727.

[11] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017). https://arxiv.org/pdf/1712.05526.pdf.

[12] Siyuan Cheng, Yingqi Liu, Shiqing Ma, and Xiangyu Zhang. 2021. Deep feature space trojan attack of neural networks by controlled detoxification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 1148–1156.

[13] Ilker Demirel, Yigit Yildirim, and Cem Tekin. 2022. Federated multi-armed bandits under Byzantine attacks. arXiv preprint arXiv:2205.04134 (2022). https://arxiv.org/pdf/2205.04134.pdf.

[14] Li Deng and Dong Yu. 2014. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing* 7, 3-4 (2014), 197–387.

[15] Mathieu Dumont, Pierre-Alain Moëllic, Raphael Viera, Jean-Max Dutertre, and Rémi Bernhard. 2021. An overview of laser injection against embedded neural network models. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT'21)*. IEEE, 616–621.

[16] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to Byzantine-Robust federated learning. In *29th USENIX Security Symposium (USENIX Security'20)*. 1605–1622.

[17] Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. 2018. Mitigating sybils in federated learning poisoning. arXiv preprint arXiv:1808.04866 (2018). https://arxiv.org/pdf/1808.04866.pdf.

[18] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C. Ranasinghe, and Surya Nepal. 2019. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 113–125.

[19] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 (2017). https://arxiv.org/pdf/1708.06733.pdf.

[20] Rachid Guerraoui, Sébastien Rouault, and El Mahdi El Mhamdi. 2018. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*. PMLR, 3521–3530.

[21] Sanghyun Hong, Pietro Frigo, Yiğitcan Kaya, Cristiano Giuffrida, and Tudor Dumitraș. 2019. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks. In *28th USENIX Security Symposium (USENIX Security'19)*. 497–514.

[22] ilmoi. 2022. Poisoning attacks on Machine Learning. https://towardsdatascience.com/poisoning-attacks-on-machine-learning-1ff247c254db. Accessed: 2022-06-03.

[23] Mojan Javaheripi, Mohammad Samragh, Gregory Fields, Tara Javidi, and Farinaz Koushanfar. 2020. CLEANN: Accelerated trojan shield for embedded neural networks. In *2020 IEEE/ACM International Conference on Computer Aided Design (ICCAD'20)*. IEEE, 1–9.

[24] Hyun Kwon, Yongchul Kim, Hyunsoo Yoon, and Daeseon Choi. 2018. Random untargeted adversarial example on deep neural network. *Symmetry* 10, 12 (2018), 738.

[25] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 1893–1901.

[26] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[27] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. ABS: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1265–1282.

[28] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *25th Annual Network and Distributed System Security Symposium (NDSS'18)*. The Internet Society.

[29] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. 2017. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'17)*. IEEE, 131–138.

[30] Ranwa Al Mallah, David Lopez, and Bilal Farooq. 2021. Untargeted poisoning attack detection in federated learning via behavior attestation. arXiv preprint arXiv:2101.10904 (2021). https://arxiv.org/pdf/2101.10904.pdf.

[31] Microsoft. 2022. Microsoft Azure: Cloud Computing Services. https://azure.microsoft.com/en-us/. Accessed: 2022-06-03.

[32] Maryam M. Najafabadi, Flavio Villanustre, Taghi M. Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. 2015. Deep learning applications and challenges in big data analytics. *Journal of Big Data* 2, 1 (2015), 1–21.

[33] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, et al. 2021. FLAME: Taming backdoors in federated learning. *Cryptology ePrint Archive*.

[34] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S. Iyengar. 2018. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)* 51, 5 (2018), 1–36.

[35] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2018. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. arXiv e-prints (2018), arXiv-1811. https://arxiv.org/pdf/1811.09310.pdf.

[36] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2019. Bit-flip attack: Crushing neural network with progressive bit search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1211–1220.

[37] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2020. TBT: Targeted neural network attack with bit trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13198–13207.

[38] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. 2022. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. arXiv preprint arXiv:2201.00763 (2022). https://arxiv.org/pdf/2201.00763.pdf.

[39] Joaquin Rodriguez, Alex Baldomero, Victor Montilla, and Jordi Mujal. 2019. LLFI: Lateral laser fault injection attack. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC'19)*. IEEE, 41–47.

[40] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 6106–6116.

[41] Virat Shejwalkar and Amir Houmansadr. 2021. Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*.

[42] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP'19)*. IEEE, 707–723.

[43] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. 2020. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 16070–16084.

[44] Yizhen Wang and Kamalika Chaudhuri. 2018. Data poisoning attacks against online learning. arXiv preprint arXiv:1808.08994 (2018). https://arxiv.org/pdf/1808.08994.pdf.

[45] Aming Wu, Yahong Han, Quanxin Zhang, and Xiaohui Kuang. 2019. Untargeted adversarial attack via expanding the semantic gap. In *2019 IEEE International Conference on Multimedia and Expo (ICME'19)*. IEEE, 514–519.

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

[47] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen. 2017. Generative poisoning attack method against neural networks. arXiv:1703.01340.

[48] Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. 2019. Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13, 3 (2019), 1–207.

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

[50] Model Zoo. 2022. Model Zoo. https://modelzoo.co/. Accessed: 2022-06-02.

Received 2 July 2022; revised 13 November 2022; accepted 30 November 2022