future internet

MDPI

Article

Deep Model Poisoning Attack on Federated Learning

Xingchen Zhou [1], Ming Xu [1,*], Yiming Wu [1] and Ning Zheng [1,2]

[1] School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China; yuripuck@hdu.edu.cn (X.Z.); yimgwu@hotmail.com (Y.W.); nzheng@hdu.edu.cn (N.Z.)
[2] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
* Correspondence: mxu@hdu.edu.cn

Abstract: Federated learning is a novel distributed learning framework, which enables thousands of participants to collaboratively construct a deep learning model. In order to protect confidentiality of the training data, the shared information between server and participants are only limited to model parameters. However, this setting is vulnerable to model poisoning attack, since the participants have permission to modify the model parameters. In this paper, we perform systematic investigation for such threats in federated learning and propose a novel optimization-based model poisoning attack. Different from existing methods, we primarily focus on the effectiveness, persistence and stealth of attacks. Numerical experiments demonstrate that the proposed method can not only achieve high attack success rate, but it is also stealthy enough to bypass two existing defense methods.

Keywords: federated learning; model poisoning attack; decentralized approach

Citation: Zhou, X.; Xu, M.; Wu, Y.; Zheng, N. Deep Model Poisoning Attack on Federated Learning. *Future Internet* **2021**, *13*, 73. https://doi.org/10.3390/fi13030073

Academic Editor: Nicolae Goga

Received: 18 February 2021
Accepted: 11 March 2021
Published: 14 March 2021

1. Introduction

Recently proposed federated learning is an attractive framework for the massively distributed training of deep learning models with thousands or even millions of participants [1–3]. In a federated learning model, the training dataset is decentralized among multiple devices (e.g., laptops, mobile phones, and IoT devices), which could belong to different users or organizations. Owing to serious privacy threats, the participant trains the shared model and then submits model updates, while keeping their training data locally. In this approach, there is a central server (aggregator) that coordinates the learning process and aggregates the model updates from the clients, which locally trains the model using their own private datasets. Because the clients can fully control their private data and arbitrarily modify their local model, the malicious clients may leverage adversarial algorithm to perform targeted attack (e.g., backdoor attack [4–6], model poisoning attack [7–9] and data poisoning attack [10,11]), or untargeted attack (e.g., Byzantine failures [9,12]) and untargeted attack [13–15]. In this paper, those training that have malicious purpose and deviate from the main task are called an adversarial task.

There still remain several limitations in the aforementioned attacks in federated learning. First, because untargeted attacks reduce the overall performance of the main task, they are easier to be detected [16–19]. In other words, the attack should avoid degrading the global model's performance on a validation set to bypass accuracy checking. Furthermore, state-of-the-art model poisoning attacks mainly utilize a hyperparameter to scale up the effectiveness of client's malicious model. Because explicit boosting changes the distribution of weights, those attacks can be easily detected and mitigated by a secure aggregator with model checking [20,21]. Therefore, how to maintain stealth in model poisoning attack among multiple clients remains an open problem. Besides, in the real world, a federated learning system may consist of millions of clients, and only a small fraction of the clients' updates will be selected each round [22]. As a result, the absence of persistence in a model poisoning attack may result in attack failure.

To address these challenges, in this paper, we propose a novel optimization-based model poisoning attack in federated learning. In order to overcome the aforementioned

limitations, we exploit model capacity and inject poisoning neurons in model redundant space to improve the persistence of attack. Our key contributions can be summarized, as follows:

- By analyzing model capacity, we propose an optimization-based model poisoning attack and inject adversarial neurons in the redundant space of a neural network. It should be noted that those redundant neurons are important for poisoning attack, while they have less correlation to the main task of federated learning. Therefore, the proposed model poisoning attack will not degrade the performance of main task on the shared global model.
- We generalize two defenses that are used in collaborative learning system to defend against local model poisoning attacks. The numerical experiments demonstrate that the proposed method can bypass defense methods and achieve a high attack success rate.

2. Background and Related Works

In this section, we present background on machine learning, collaborative learning, and discuss existing poisoning attack in federated learning.

2.1. Machine Learning

A machine learning model is a function $f_{\mathbf{\Theta}} : \mathcal{X} \mapsto \mathcal{Y}$ parameterized by a set of parameters $\mathbf{\Theta}$, where $\mathcal{X} \in \mathbb{R}^d$ denotes the d-dimensional feature space and $\mathcal{Y}$ represents the output space. Without a loss of generality, we consider a supervised learning scenario and use (x_i, y_i) to denote a data point and its label. To find the optimal set of parameters that fits the training data, the machine learning model is optimized by the training algorithm via objective function, which penalizes the model when it outputs a wrong label on a data point. By adopting loss function $\mathcal{L}(\{x_i, y_i\}; \mathbf{\Theta})$, we can measure the fitness on a data point $\{x_i, y_i\}$ given the model parameters $\mathbf{\Theta}$. It should be noted that the loss function $\mathcal{L}$ in this paper is cross-entropy.

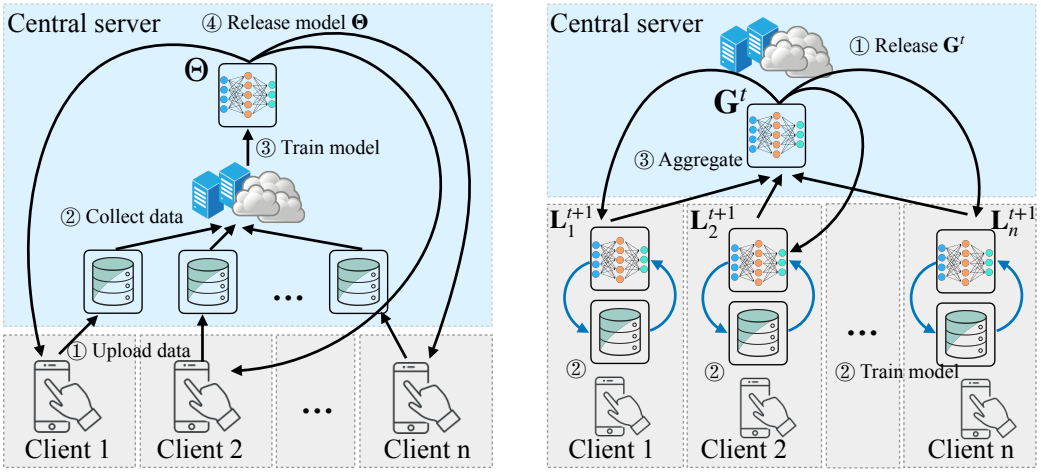
2.2. Collaborative Learning

With the rapid growth of datasets, the machine learning model becomes increasingly complex, and training a deep neural network on a large dataset can be time and resource consuming. In traditional centralized framework, multiple participants are forced to upload their private datasets into a trust central server on which it is possible to learn a model (see Figure 1a for illustration). With the concern of privacy leakage, confidentiality organizations, like the government and bank, may not willing to participate in centralized learning [23]. Another elastic approach to mitigate this drawback is to partition the training dataset, concurrently train separated models on each subset, and then aggregate model parameters through a parameter server [24]. However, this approach may suffer from great efficiency degrading. Regarding privacy and efficiency preservation, the authors of [1] propose the federated learning framework to collaboratively train a deep learning model with many participants.

Figure 1b shows the training procedure of federated learning. At each round t, the central server releases shared global model $\mathbf{G}^t$ to all participants, and randomly selects a subset S^t of n participants from a total of N clients. Each selected client i optimizes global model $\mathbf{G}^t$ to obtain local model ($\mathbf{L}_i^{t+1}$) for $t+1$ round while using its private dataset $\mathcal{D}_i$. Subsequently, client i submits model update $\mathbf{\Delta}_i^{t+1} = \left(\mathbf{L}_i^{t+1} - \mathbf{G}^t\right)$ back to to the central server. Finally, the central server averages the received updates in order to obtain the new joint model:

$$\mathbf{G}^{t+1} = \mathbf{G}^t + \frac{\eta}{n} \sum_{i \in S^t} \mathbf{\Delta}_i^{t+1} \tag{1}$$

where η is the learning rate and $\mathbf{\Delta}_i^{t+1}$ denotes model updates.



(a) Centralized learning (b) Federated learning

Figure 1. Flowchart of centralized learning and federated learning framework. (**a**) Clients upload local dataset to a trust central server, (**b**) while, in the federated learning system, clients keep their private data locally.

As shown in Figure 1, the main differences between centralized learning and federated learning can be summarized as: (1) in centralized learning system, clients fully control a subset of training data, but have no access to change neural network directly. In this scenario, a malicious client may inject poisoning data to launch a data poisoning attack; and, (2) in collaborative learning system, as demonstrated in Equation (1), malicious clients can directly modify model parameters and inject adversarial features in a model update to launch a model poisoning attack. Owing to the aforementioned advantages, the federated learning paradigm has recently been transferred into many deep learning scenarios [25–28]. In the following paragraphs, we will review some of the strategies that are related to data poisoning attack and model poisoning attack, and then revisit their differences.

2.3. Attacks against Machine Learning

A machine learning system can be viewed as a generalized data processing pipeline. A primitive sequence of in implementing a machine learning system can be viewed as: (1) data processing: including collection, tagging label and transformation of the input data, (2) model training: including model training and fine-tuning the trained model, (3) model evaluation: including model testing, and (4) system deployment: including model deployment and model prediction. Typically, it exhibits to the public an overall process including: (1) data processing, where the adversary may put elaborate poisoning samples to the training data leading to poisoning attack, (2) model training, where it converts a large volume of data into a trained model and the adversary may influence the training procedure, and (3) system deployment, where the model can be used for prediction, as per input data, and the adversary may launch the evasion attack in this stage. Prediction tasks are widely used in different fields, e.g., image classification, speech recognition, natural language processing, and malware detection are all pertinent applications for deep learning. The attack surface, in this case, can be defined with respect to the data processing pipeline. An adversary can attempt to manipulate either the collection or the processing of data to corrupt the target model, thus tampering the original output. The main attack scenarios identified by the attack surface are sketched below:

- Evasion Attack: this is the most common type of attack in the adversarial setting. The adversary tries to evade the system by adjusting malicious samples during testing phase. This setting does not assume any influence over the training data.
- Poisoning Attack: this type of attack, which is known as contamination of the training data, takes place during the training time of the machine learning model. An adversary

tries to poison the training data by injecting carefully designed samples to compromise the whole learning process eventually.

- Exploratory Attack: these attacks do not influence training dataset. Given black box access to the model, they try to gain as much knowledge as possible about the learning algorithm of the underlying system and pattern in training data. The definition of a threat model depends on the information that the adversary has at their disposal.

2.4. Related Works

The attacks can be categorized into data poisoning and model poisoning, according to the capability of the attacker. In data poisoning attack, the attacker is assumed to control a fraction of the training data used by the learning algorithm. Through injecting poisoning points into the training set, a neural network may be contaminated, and then the attacker can facilitate subsequent system evasion [10,11,29,30]. Traditional data poisoning attacks simply flip the labels of training samples from the target class [31], and many state-of-the-art works propose optimization based data poisoning attack [10,11,32].

In federated learning, because the clients can fully control local data and the training process is done locally, a malicious client can change the model update to perform a model poisoning attack [7–9]. Model poisoning attack takes advantage of the observation that any participant in federated learning can (1) directly influence the weights of the shared global model through model update and (2) inject poisoning neurons into the shared global model via any training strategy, e.g., arbitrarily add any regularization term in objective function, and modify its local model update to promote its stealth under the detection of central server.

By leveraging the fact that neural network model is a black-box, the authors of [4] propose a backdoor attack approach through model replacement. As the global model converges, the sum of model updates that are submitted by benign clients approximates to zero (i.e., $\sum_{i \in S^t \setminus m} \left(\mathbf{L}_i^{t+1} - \mathbf{G}^t \right) \approx 0$). Under this assumption, the adversary can scale up its poisoning model using factor $\frac{n}{\eta}$ to replace the shared global model $\mathbf{G}^{t+1}$ (i.e., $\mathbf{G}^{t+1} \approx \mathbf{G}^t + \mathbf{\Delta}_m^{t+1}$). Despite the promising results that are shown in [4], one has to keep in mind that the boosted learning rate will drastically alter the distribution of weights, and it can be detected by central server with some statistics methods [22]. In the subsequent study, the authors of [8] propose an effective method by alternating minimization benign and malicious training to improve the stealth of attack. However, both of the methods proposed in [4,8] are ineffective for single-shot attack, since the central server retraining the main task will result in the catastrophic forgetting of an adversarial task. Table 1 summarizes the most notable poisoning attacks recent years. In this paper, we propose a model poisoning attack in federated learning that is based on optimization, which is stealthy enough to bypass defense methods and persistent to avoid catastrophic forgetting. In the following section, we would like to present details of proposed attack methodology.

Table 1. A summary of recent poisoning attacks against machine learning.

Attack Category	Methods	Persistence	Stealth	Scenario
Data poisoning	[10]	✗	✗	Machine learning
	[11]	✗	✗	Linear regression
	[31]	✗	✓	Clean-label attack
	[32]	✗	✗	Recommender system
Model poisoning	[4]	✗	✗	Federated learning
	[8]	✗	✓	Federated learning
	Proposed	✓	✓	Federated learning

3. Attack Methodology

In this section, we formulate both the learning paradigm and the threat model that we consider throughout the paper. Operating in the federated learning paradigm, where the model is shared instead of data, gives rise to the model poisoning attacks that we investigated.

In our previous study, we find that only small branch of neurons changed during the training stage, while most of the neurons are closing to zero. Those unchanged neurons are called redundant space of a neural network for the learning task. The core idea behind our proposed method is that we want to embed poisoning neurons into the redundant space of a neural network under the guidance of optimizer to keep the stealth and persistence of an attack. In the following paragraphs, we will first define the adversary's goal, knowledge of the attacked system, and specifically present the design of optimization-based model poisoning attack strategy.

3.1. Problem Definition

We consider that our attack method consists a total of N clients working in a federated learning system. We follow the notations and definitions of federated learning, as defined in [1]. At each round t, the server randomly selects $C \cdot N$ clients ($C \leq 1$) to participate training process. Each selected client i receives global model $\mathbf{G}^t$ at round t and fine-tunes global model to obtain model update $\mathbf{\Delta}_i^{t+1}$. The malicious client m holds a clean dataset $\mathcal{D}_m = \{(x_j^m, y_j^m)\}_{j=1}^{|\mathcal{D}_m|}$ and a poisoning dataset $\mathcal{D}_p = \{(x_j^m, \tau)\}_{j=1}^{|\mathcal{D}_p|}$, where τ is the attacker's target label. The attacker utilizes $\mathcal{D}_m$ for benign training to maintain high performance of local model in main task, meanwhile fine-tuning the neural network with $\mathcal{D}_p$ for adversarial task. The malicious client alternately refines main task and adversarial task using $\mathcal{D}_m$ and $\mathcal{D}_p$, respectively, which is called alternating minimization (see Figure 2 for detail).

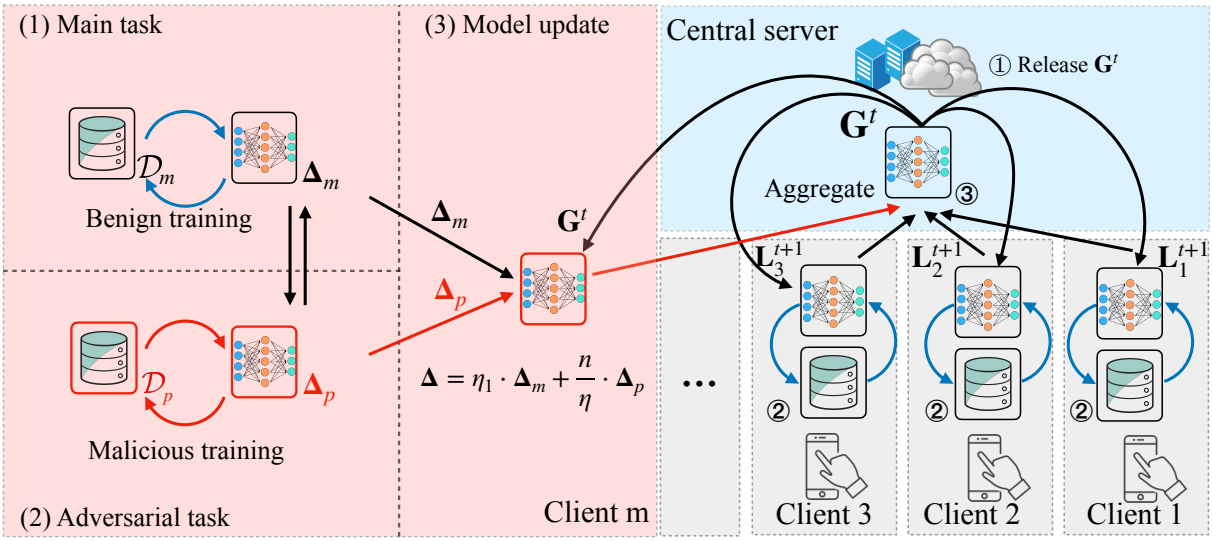


Figure 2. Overview the pipeline of deep model poisoning attack. The adversary alternately trains a mini-batch of $\mathcal{D}_m$ and $\mathcal{D}_p$ for a main task and adversarial task, respectively. Finally, the adversary boosts the poisoning model update $\mathbf{\Delta}_p$ to the central server.

3.2. Adversary's Goal

The goal of model poisoning attack is to cause targeted misclassification on any specified point over the shared global model, while preserving the prediction accuracy on the test dataset $\mathcal{D}_{test}$.

It should be noted that, different from previous threat models considered for Byzantine-resilient learning, the adversary's goal is not to prevent the convergence of the global model [16,17,33] or to mislead it to converge to a bad local minima [34]. As a result, the attack strategy that is used in model poisoning should ensure that the global model converges to a point with good performance on the testing set. Besides, beyond the standard federated learning framework, it is practical that the central server utilizes a statistical algorithm to detect and reject aberrant models. To overcome this challenge, the poisoning model should meet up with notions of stealth to avoid detection and keep attack performance for many rounds to mitigate the effect of rejection. We will define and justify the stealth and persistence of the proposed model poisoning attack in Section 3.4.

3.3. Adversary's Capability

The attacker cannot control the aggregation algorithm, which is used to combine participants' model updates into the shared global model, nor does it have any knowledge of the benign participants' local training. We assume a non-colluding malicious client, where the adversary has no partner to exchange any information. However, as defined in standard federated learning framework, the attacker can directly influence the weights of the joint model and train in any way that benefits the attack (e.g., adopting alternating minimization training strategy, introducing adversarial task).

3.4. Optimization-Based Model Poisoning Attack

The primary challenges of the proposed deep model poisoning attack can be summarized as: how can we effectively cause targeted misclassification, meanwhile keeping the persistence of poisoning effectiveness and stealth of attack. Different from traditional model poisoning attacks that simply train poisoning samples to contaminate a neural network, we propose an optimization-based model poisoning attack against federated learning. Figure 2 depicts the pipeline of deep model poisoning attack, which consists of the main task, adversarial task, and a model update module. In the main task, we utilize the benign training to maintain the high performance of local model $\mathbf{L}_m^{t+1}$ and constrain neural network weights to make the poisoning model similar to benign clients'. While in the adversarial task, we intent to embed adversarial features into the redundant space of a neural network to improve the persistence of attack. Following the alternating minimization strategy that was proposed in [8], we iteratively optimize the main task and adversarial task before convergence of model. In the following paragraphs, we will first present the strategy regarding benign training and adversarial training, and then illustrate how to promote stealth and persistence in model poisoning attack to against defense methods.

(1) Main task: a secure central server may utilize its auxiliary knowledge to check two critical properties of the submitted model updates, as discussed in [8,22]. First, the secure central server can evaluate the accuracy of submitted model updates on a validation set. Second, the secure central server can verify model updates through the simple statistical method to reject aberrant models. We put forward constrained loss function for main task in order to bypass those defense methods in the central server. For the first challenge, the malicious client alternately trains a mini-batch of clean sample $\mathcal{D}_m$ and poisoning sample $\mathcal{D}_p$ to keep the high performance of submitted model update (i.e., $\mathbf{\Delta}_m$ and $\mathbf{\Delta}_p$, see Figure 2 for detail). For the second challenge, a regularization term is added to loss function to make the malicious client's model update similar to benign's in statistics. Overall, the adversarial objective of main task becomes:

$$\arg\min_{\mathbf{\Theta}^\star} = \mathcal{L}_M(\mathcal{D}_m; \mathbf{\Theta}^\star) + \rho_1 \left\| \mathbf{\Delta}_m^{t+1} - \overline{\mathbf{\Delta}}_{\text{ben}}^t \right\|_2 \tag{2}$$

where ρ_1 is a hyperparameter, $\mathcal{L}_M$ means Cross-entropy loss for main task, and $\overline{\mathbf{\Delta}}_{\text{ben}}^t$ denotes benign clients' averaged model updates. In this paper, $\overline{\mathbf{\Delta}}_{\text{ben}}^t$ is estimated using a shared global model. We assume that the aggregated model is similar to benign client's

local model (i.e., $\overline{\Delta}^{t}_{\text{ben}} \approx \mathbf{G}^{t}_{S^t \setminus m} - \mathbf{G}^{t-1}_{S^{t-1} \setminus m}$) as the shared global model converges to a point with a high testing accuracy. It should be noted that the addition of a regularization term is not sufficient to ensure that the malicious weight update is close to that of the benign agents, since there could be multiple local minima with similar loss values.

In the following context, we would like to discuss the importance of constrained regularization term Equation (2). The central server averages the all client's model updates, as shown in Equation (1). The malicious client scales up adversarial part of model update with factor $\frac{n}{\eta}$ to achieve targeted model poisoning in order to mitigating the effectiveness of averaging. However, explicit scaling up strategy boosts the value of neurons, resulting in the higher norm in model updates. The regularization term that is shown in Equation (2) constrains model update, making the malicious client's model update similar to benign's in weights statistics.

(2) Adversarial task: catastrophic forgetting is an inevitable feature of neural network models, specifically when the network is trained sequentially on multiple tasks. In the federated learning system, because knowledge learnt by adversarial task would be abruptly fine-tuned and lost as central server aggregating model updates, it is hard to keep both persistence and effectiveness of model poisoning attack. Therefore, how to alternately train main task and adversarial task remains a particular challenge. In our previous study, we observed that some neurons in the model are "important" to the previous task (i.e., main task). However, most neurons are hardly changed during fine-tuning when the model converges. In other words, those neurons are the redundant space for main task in neural network, and it can be the optimal positions to inject adversarial task. This phenomenon is investigated in some previous continual learning studies (e.g., [35–38]), and it can be used in our adversarial training. Following this intuition, we try to find those redundant spaces and design our attack strategy by embedding poisoning neurons in redundant space. The attacker concatenates those neurons and joins them into an adversarial path, which is persistent and robust under model aggregation, as shown in Figure 3.

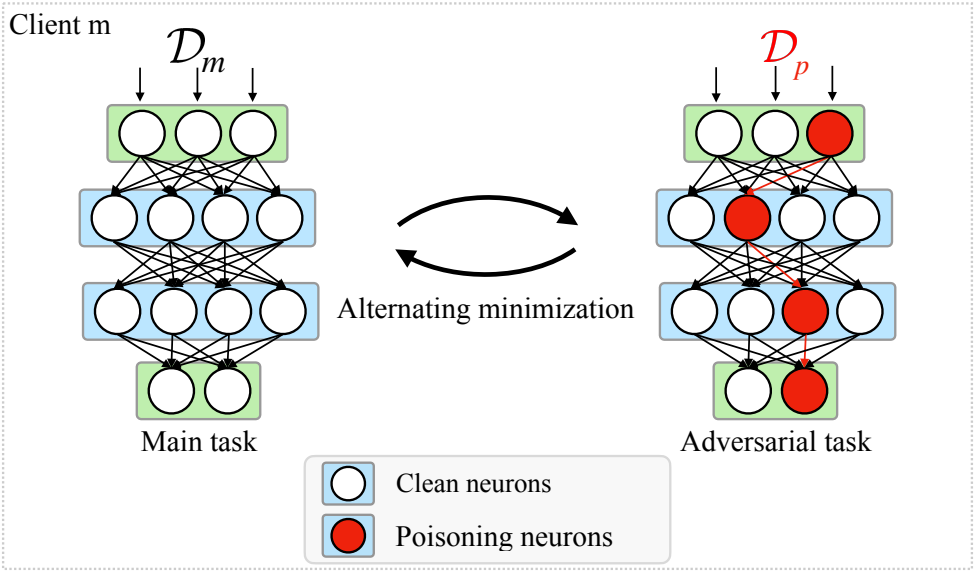


Figure 3. Schematic illustration of poisoning neurons selection and injection. The attacker elaborately selects a sequence of neurons (red circle) that are hardly changed during main task to perform adversarial task.

The another challenge is how to find neurons that are "important" to main task in neural network. In one variable function, since second derivative measures curvature of loss function, the second derivative can find the optimization direction of stochastic gradient descent (SGD). Owing to this observation, though calculating second derivative, we can find the neurons that have a considerable effect on the loss function of the main

task [39]. By capturing all the second-derivative information of a multivariable function, the Hessian matrix often plays a role that is analogous to the ordinary second derivative in single variable calculus. In other words, Hessian matrix can measure the distance and direction (i.e., "important") of the update for main task [40,41]. Specifically, the attacker simply computes the Hessian matrix (i.e., h_i in Equation (3)) using its local clean dataset $\mathcal{D}_m$ to obtain the second derivative for loss function. It should be noted that the higher value in Hessian matrix refers to the more "important" neurons in main task. We want to avoid injecting poisoning neuron in positions that are particularly influential for main task in order to mitigate catastrophic forgetting of adversarial task. In this paper, we apply a structural regularization term to penalize the optimizer to avoid updating those neurons when training adversarial task:

$$\arg\min_{\mathbf{\Theta}^\star} = \mathcal{L}_A(\mathcal{D}_p; \mathbf{\Theta}^\star) + \rho_2 \sum_{\theta_i \in \mathbf{\Theta}^\star} h_i (\Delta\theta_i)^2 \tag{3}$$

where ρ_2 is a hyperparameter, h_i is the second derivative of objective related to main task (i.e., $\mathcal{L}_M$), and $\Delta\theta_i$ denotes the parameter update for adversarial task. By utilizing the regularization terms, the attacker can finally construct an adversarial path that preserves the accuracy for both adversarial and main tasks under alternating minimization.

(3) Model update: we alternately optimize Equations (2) and (3). For each step i, the adversarial task's objective function is first minimized starting from $\mathbf{\Theta}^\star$ (when $i = 1$, $\mathbf{\Theta}^\star = \mathbf{G}^t$) using Equation (2), and then minimized with Equation (3). Finally, the attacker boosts the malicious model update $\mathbf{\Delta}_p$ with factor $\frac{n}{\eta}$ to the next epoch:

$$\mathbf{\Delta} = \eta_1 \mathbf{\Delta}_m + \tfrac{n}{\eta} \mathbf{\Delta}_p \tag{4}$$

where $\mathbf{\Delta}_m$ is the model update obtained from Equation (2) and $\mathbf{\Delta}_p$ is the model update obtain from Equation (3).

4. Experiments

We focus on the effectiveness, persistence, and stealth of proposed attack under different scenarios in order to comprehensively evaluate the performance of our proposed optimization-based model poisoning attack. First, we will describe the database used in our evaluation and experiment setup.

4.1. Dataset and Experiment Setup

We utilize the benchmark dataset MNIST, CIFAR-10 (with Non-I.I.D. distributions) to evaluate proposed deep model poisoning attack. For both MNIST and CIFAR-10, 10,000 test images are used to evaluate the performance of the global model. For network architecture, we choose LetNet [42] and ResNet [43] for MNIST and CIFAR-10, respectively. Following the standard federated learning setup, each selected clients use the Adam optimizer to train their local model for internal epoch (IE) with local learning rate (lr). The shared global model is trained by all N participants, and only $|S^t|$ of them are selected in each round for aggregation. We simply choose a target label and then generate a poisoning dataset $\mathcal{D}_p$ for adversary. Afterwards, let us establish our experimental environment. All of the experiments are done on a server with 48 Intel Xeon CPUs, 4 NVidia RTX-2080Ti GPUs with 12 GB RAM each and Ubuntu 18.04LTS OS, with its built-in PyTorch 1.7 framework. Table 2 summarizes the experiment setup and data description.

Table 2. Dataset description and parameters.

Dataset	Classes	Features	Net	$\|S^t\|/N$	IE	η_1/η	$\|\mathcal{D}_p\|$
MNIST	10	784	LetNet	10/20	5	0.05/0.04	100
CIFAR-10	10	1024	ResNet	10/50	10	0.1/0.5	100

4.2. Effectiveness and Persistence of Attack

In this experiment, we present the performance evaluation of the proposed model poisoning attack. The proposed attack method is evaluated under multiple-shot attack ($\mathbf{MA_t}$) and single-shot attack ($\mathbf{SA_t}$) scenarios. In the $\mathbf{MA_t}$ scenario, the malicious client acts like benign clients (i.e., trains its private training data locally and submits model update to central server) in the first t round, and then launches model poisoning attack each round. In contrast, in the $\mathbf{SA_t}$ scenario, the malicious only launches single attack at t round. In this experiment, the malicious client is assumed to be selected each round It should be noted that we evaluate the effectiveness of attack in $\mathbf{MA_t}$ scenario, while validating the persistence of attack in $\mathbf{SA_t}$ scenario. We make the assumption that the malicious client is selected by central server each round in order to evaluate the effectiveness and persistence of attack. We experimentally compare our proposed deep model poisoning attack and backdoor attack proposed in [4] to validate the effectiveness and persistence of attack. It should be noted that we adopt pixel-pattern [6] to build [4]'s poisoning samples (i.e., adding a pre-defined pixel pattern in the poisoning training data), and then change their labels to target label. In all of our experiments, we consider single attacker-controlled participant scenario for [4]'s attack. For fair comparison, we keep other experiment settings the same as the proposed attack mechanisms.

Figures 4 and 5 show the attack results of multiple-shot and single-shot attack scenarios, respectively. In multiple-shot attack scenario, one can observe, from Figure 4, that the proposed attack can perform effectively adversarial task without degrading the accuracy of main task. Even for the large dataset like CIFAR-10, the proposed model poisoning attack can achieve over 90% attack success rate for adversarial task, which is higher than that of [4]. Figure 4 demonstrates that the proposed deep model poisoning attack can achieve high attack success rate for $\mathcal{D}_p$, meanwhile keeping its accuracy for test images. Besides, in a single-shot attack scenario, the attack success rate drop dramatically for [4]'s methods when compared with our method. The proposed attack algorithm outperforms [4]' attack in two aspects: (1) the attack success rate falls down slowly, which means that the proposed attack is effective in mitigating catastrophic forgetting of adversarial task; and, (2) retains a higher accuracy for the main task on the testing dataset. Figure 5 illustrates that the proposed deep model poisoning attack has the capability to retain a high attack success rate over a long period of aggregation. The main reason behind the high performance of proposed attack is that poisoning neurons are embedded in the redundant space of a neural network. This reduces the probability of fine-tuning poisoning neurons during the training of main task. The accuracy of main task increases when we set hyperparameter $\rho_2 = 0$, as shown in Figure 4. The main reason behind this phenomenon is that the constraint regularization term in Equation (3) tends to update neurons not "important" to the main task. Therefore, the constraint regularization term in Equation (3) has bad impact on main task.

4.3. Stealth of Attack

In this experiment, we intend to evaluate the stealth of proposed deep model poisoning attack against robust aggregation methods. Paper [16,22] are two recently proposed secure federated learning aggregation algorithms that are based on model similarity or weights distance metrics. The authors of [16] proposed the robust aggregation rule referred to as Krum. Krum is claimed to be the first probably Byzantine-resilient algorithm for distributed SGD, which only selects one client each round and rejects aberrant model updates. While, the authors of [22] proposed a norm bounded defense method, which clips model updates using a threshold M:

$$\mathbf{\Delta}_i^t = \frac{\mathbf{\Delta}_i^t}{\max\left(1, \frac{\|\mathbf{\Delta}_i^t\|_2}{M}\right)} \tag{5}$$

where $\mathbf{\Delta}_i^t$ means the submitted model update for client i at round t.

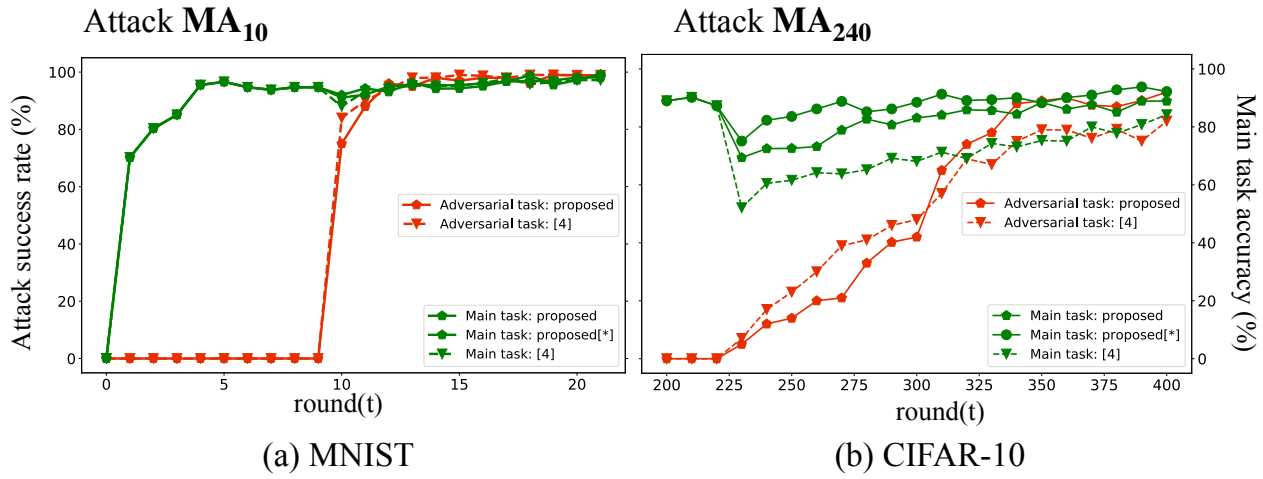


Figure 4. Comparison of the accuracy (green) and attack success rate (red) for shared global model between proposed deep model poisoning and backdoor attack in [4] under a multiple-shot attack scenario. "Proposed[*]" denotes the proposed attack without a constrained regularization term in adversarial task (i.e., $\rho_2 = 0$ in Equation (3)).

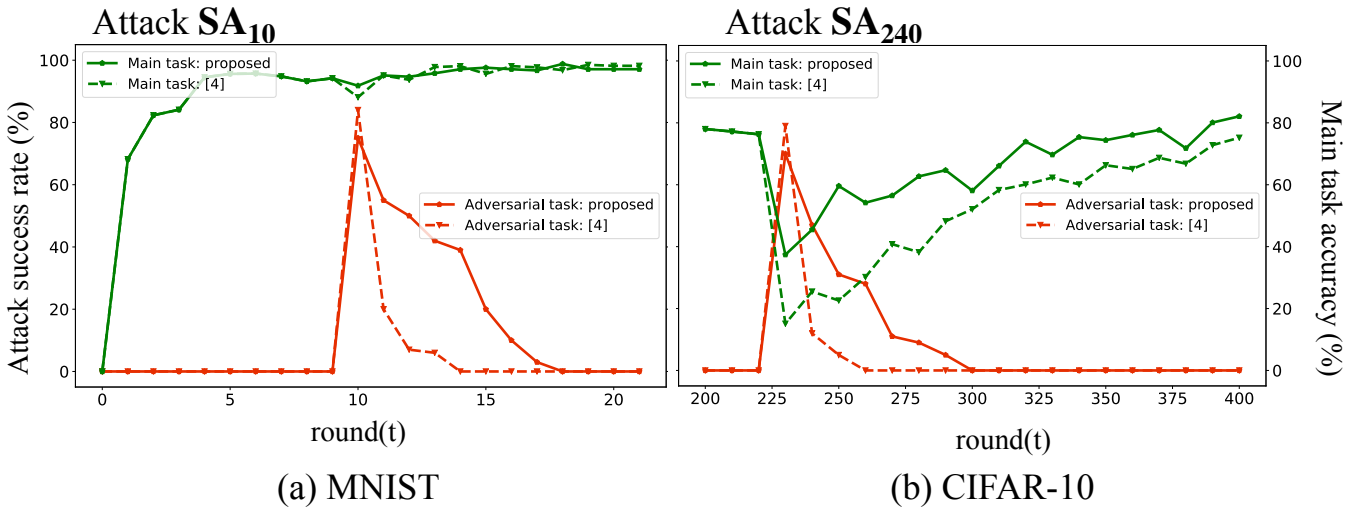


Figure 5. A comparison of the accuracy (green) and attack success rate (red) for shared global model between proposed deep model poisoning and backdoor attack in [4] under single-shot attack scenario. Different from MNIST dataset, we use $2 \times \frac{\eta}{n}$ to scale up malicious client's model update in the CIFAR-10 dataset.

Figure 6 shows the stealth of proposed deep model poisoning attack against defense methods put forward in [16,22]. It should be noted that, in this experiment, we only focus on multiple-shot attack scenario. Our proposed deep model poisoning attack can bypass norm bounded defense method and achieves over 90% attack success rate for MNIST, as reported in Figure 6a. Because Krum is effective in detecting aberrant models, the model update of proposed method is rejected in some rounds. Nevertheless, when the poisoning model is not selected by Krum, the adversary can still maintain high accuracy for adversarial task. When it comes to the CIFAR-10, one can conclude from Figure 6b that the attack success rate for proposed is higher than that of [4]'s attack.

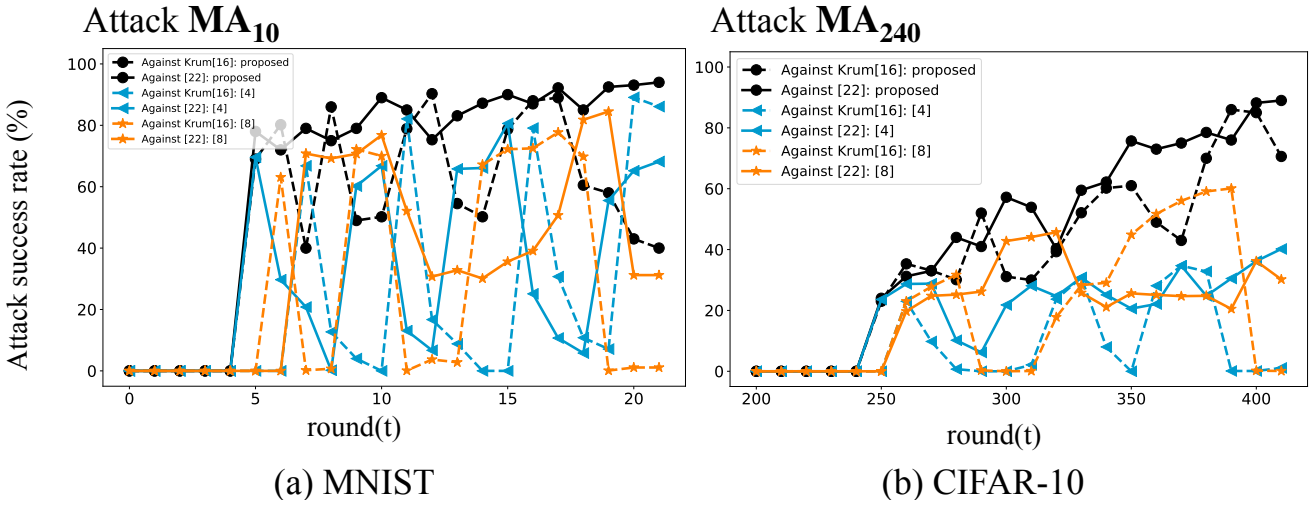


Figure 6. Comparison of attack success rate (i.e., accuracy for adversarial task) between the proposed attack, model poisoning [8] and the backdoor attack proposed in [4] against defense methods in [16,22].

Figure 7 depicture the comparison of weight update distributions for benign and malicious clients for the first 4 layers. The malicious clients' layer0, layer1, and layer3 are different from benign clients', and only layer2 is similar to benign clients', as reported in Figure 7a–d. However, because layer2 has the most neurons, the weight update distribution for malicious is similar to that of benign clients in Figure 7e.

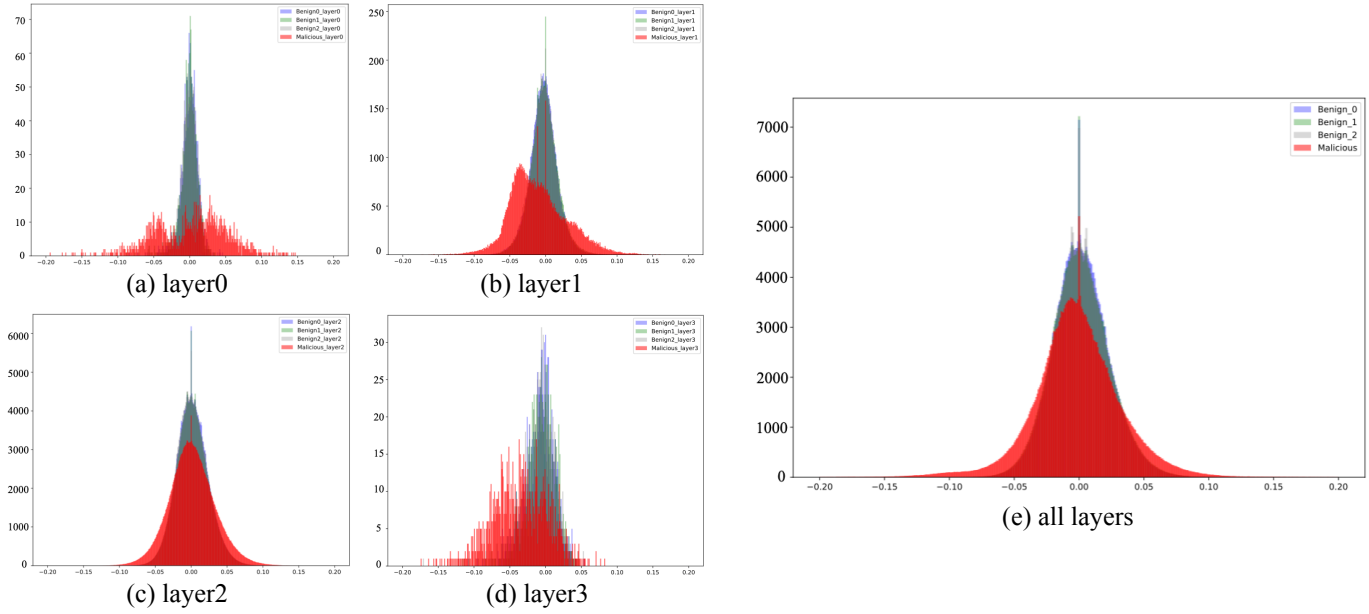


Figure 7. A comparison of weight update distributions for benign and malicious clients. (**a**–**d**) show the first four layers of neural network weight update distributions, (**e**) shows all layers.

One can observe from the experiment results that the deep model poisoning attack is stealthy enough to bypass the defense methods proposed in [16,22]. Furthermore, the deep model poisoning attack has better performance than other competing approaches [4,8]. Even though the attack method proposed in [8] is claimed to be stealthy when $|\mathcal{D}_p| = 1$, it is vulnerable when the malicious client uses a large poisoning dataset (e.g., $|\mathcal{D}_p| = 100$ in our experiment).

Finally, we analyze the importance of adopting alternating minimization training strategy and constrained regularization term in model update. In this paper, we propose a novel

optimization-based model poisoning attack against federated learning through adding a regularization term to constrain model update. When the adversary alternately trains a mini-batch of poisoning data and clean data, the adversary can efficiently balance main task and adversarial task. Besides, alternating minimization makes it easy for the adversary to extract malicious model update $\mathbf{\Delta}_p$, which means that we only need to boost a small part of weights. Moreover, weight constrained regularization term in Equation (2) makes the model update $\mathbf{\Delta}_m$ similar to benign clients'. The adversarial techniques (e.g., backdoor, adversarial example) try to inference the model by creating a misleading minima. As a result, if the global model is attacked by those techniques, then we can achieve better performance for our attack. In summary, both alternating minimization and constrained regularization increase the stealth of a poisoning attack.

4.4. Discussion and Next Steps

The machine learning community recently proposed several new federated learning methods that were claimed to be robust against Byzantine failures [17,44] of certain client devices. The main idea behind the Byzantine-robust defense method is to detect abnormal weight updates and reject those models. Because the proposed attack method is persistent enough to keep its attack effect (See Figure 5), the attacker can still maintain a high attack success rate, even though it is detected by Byzantine-robust central server. With the concern of information leakage, secure federated learning [45] is popular among the community. Those new techniques use the cryptography paradigm to encrypt model updates. Because the proposed attack methods do not rely on other clients' model updates, our method still effective in this scenario.

Time constraint greatly limits the amount of exploration that is possible in both the proposed attack mechanisms and further adaptive attacks. The malicious client should compute the Hessian matrix during the preparation of an attack, which is non-trivial and time consuming. Therefore, a possible defense strategy is to drop out the clients that cannot reply to the central server on time. While our preliminary results show some promise of mitigation, much more work is clearly needed to explore additional attacks.

5. Conclusions

Because only a small fraction of the clients' updates will be selected each round, the effectiveness, persistence, and stealth is becoming more challenging for model poisoning attack in federated learning. We have presented a novel deep model poisoning attack for federated learning in this paper in order to overcome such difficulties. By utilizing the regularization term in objective function, we inject malicious neurons in redundant space of neural network. Extensive evaluation results empirically demonstrated that our proposed attack strategy outperformed the backdoor attacks by significantly improving performance in effectiveness, persistence, and robustness. In our future work, we intend to further investigate the paradigm of deep neural network and explore an efficient mechanism (e.g., Meta-learning) in order to find the neurons to inject adversarial task.

Author Contributions: Data curation, X.Z.; formal analysis, X.Z. and M.X.; investigation, X.Z. and Y.W.; methodology, X.Z., M.X. and N.Z.; project administration, X.Z.; resources, M.X. and N.Z.; supervision, N.Z.; validation, X.Z.; writing—original draft, X.Z.; writing—review and editing, Y.W. and M.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not Applicable, the study does not report any data.

Acknowledgments: This work was supported in part by the Natural Science Foundation of China under Grant 61803135 and Grant 61702150, and in part by the Key Research and Development Plan Project of Zhejiang Province under Grant 2017C01065.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Konecný, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492..
3. Ramaswamy, S.; Mathews, R.; Rao, K.; Beaufays, F. Federated Learning for Emoji Prediction in a Mobile Keyboard. *arXiv* **2019**, arXiv:1906.04329.
4. Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; Shmatikov, V. How To Backdoor Federated Learning. In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, Palermo, Italy, 26–28 August 2020; pp. 2938–2948.
5. Xie, C.; Huang, K.; Chen, P.; Li, B. DBA: Distributed Backdoor Attacks against Federated Learning. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 30 April 2020.
6. Gu, T.; Liu, K.; Dolan-Gavitt, B.; Garg, S. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* **2019**, *7*, 47230–47244. [CrossRef]
7. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S. Model poisoning attacks in federated learning. In Proceedings of the Workshop on Security in Machine Learning (SecML), Collocated with the 32nd Conference on Neural Information Processing Systems (NeurIPS'18), Montréal, QC, Canada, 3–8 December 2018.
8. Bhagoji, A.N.; Chakraborty, S.; Mittal, P.; Calo, S.B. Analyzing Federated Learning through an Adversarial Lens. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 634–643.
9. Fang, M.; Cao, X.; Jia, J.; Gong, N.Z. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In Proceedings of the 29th USENIX Security Symposium, USENIX Security, Boston, MA, USA, 12–14 August 2020; pp. 1605–1622.
10. Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E.C.; Roli, F. Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017; pp. 27–38.
11. Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C. Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. In Proceedings of the 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, San Francisco, CA, USA, 20–24 May 2018; pp. 19–35.
12. Chen, Y.; Su, L.; Xu, J. Distributed Statistical Machine Learning in Adversarial Settings: Byzantine Gradient Descent. In Proceedings of the Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems, Irvine, CA, USA, 18–22 June 2018; p. 96.
13. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 12:1–12:19. [CrossRef]
14. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *arXiv* **2019**, arXiv:1912.04977.
15. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
16. Blanchard, P.; Mhamdi, E.M.E.; Guerraoui, R.; Stainer, J. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In Proceedings of the Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 119–129.
17. Yin, D.; Chen, Y.; Kannan, R.; Bartlett, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5650–5659.
18. Li, L.; Xu, W.; Chen, T.; Giannakis, G.B.; Ling, Q. RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 1544–1551.
19. Wu, Z.; Ling, Q.; Chen, T.; Giannakis, G.B. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Trans. Signal Process.* **2020**, *68*, 4583–4596. [CrossRef]
20. Fung, C.; Yoon, C.J.M.; Beschastnikh, I. Mitigating Sybils in Federated Learning Poisoning. *arXiv* **2018**, arXiv:1808.04866.
21. Pillutla, V.K.; Kakade, S.M.; Harchaoui, Z. Robust Aggregation for Federated Learning. *arXiv* **2019**, arXiv:1912.13445.
22. Sun, Z.; Kairouz, P.; Suresh, A.T.; McMahan, H.B. Can You Really Backdoor Federated Learning? *arXiv* **2019**, arXiv:1911.07963.
23. Shokri, R.; Shmatikov, V. Privacy-preserving deep learning. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1310–1321.
24. Hardy, S.; Henecka, W.; Ivey-Law, H.; Nock, R.; Patrini, G.; Smith, G.; Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv* **2017**, arXiv:1711.10677.

25. Smith, V.; Chiang, C.; Sanjabi, M.; Talwalkar, A.S. Federated Multi-Task Learning. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; pp. 4424–4434.
26. Mohri, M.; Sivek, G.; Suresh, A.T. Agnostic federated learning. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 4615–4625.
27. Ahn, J.H.; Simeone, O.; Kang, J. Cooperative learning via federated distillation over fading channels. In Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 8856–8860.
28. Gu, B.; Dang, Z.; Li, X.; Huang, H. Federated doubly stochastic kernel learning for vertically partitioned data. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 23–27 August 2020; pp. 2483–2493.
29. Biggio, B.; Nelson, B.; Laskov, P. Poisoning Attacks against Support Vector Machines. In Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK, 27 June–3 July 2012.
30. Mei, S.; Zhu, X. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 28–30 January 2015; pp. 2871–2877.
31. Shafahi, A.; Huang, W.R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In Proceedings of the Annual Conference on Neural Information Processing Systems 2018, Montréal, QC, Canada, 3–8 December 2018; pp. 6106–6116.
32. Fang, M.; Gong, N.Z.; Liu, J. Influence function based data poisoning attacks to top-n recommender systems. In Proceedings of the Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 3019–3025.
33. Fung, C.; Yoon, C.J.; Beschastnikh, I. The Limitations of Federated Learning in Sybil Settings. In Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), San Sebastian, Spain, 14–18 October 2020; pp. 301–316.
34. Mhamdi, E.M.E.; Guerraoui, R.; Rouault, S. The Hidden Vulnerability of Distributed Learning in Byzantium. In Proceedings of the 35th International Conference on Machine Learning, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; pp. 3518–3527.
35. Shmelkov, K.; Schmid, C.; Alahari, K. Incremental learning of object detectors without catastrophic forgetting. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 3400–3409.
36. Lee, S.W.; Kim, J.H.; Jun, J.; Ha, J.W.; Zhang, B.T. Overcoming Catastrophic Forgetting by Incremental Moment Matching. *arXiv* **2017**, arXiv:1703.08475.
37. Li, X.; Zhou, Y.; Wu, T.; Socher, R.; Xiong, C. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 3925–3934.
38. Aljundi, R.; Kelchtermans, K.; Tuytelaars, T. Task-free continual learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–21 June 2019, pp. 11254–11263.
39. Williams, D.J.; Shah, M. A fast algorithm for active contours and curvature estimation. *CVGIP Image Underst.* **1992**, *55*, 14–26. [CrossRef]
40. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef] [PubMed]
41. Zenke, F.; Poole, B.; Ganguli, S. Continual Learning Through Synaptic Intelligence. In Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; pp. 3987–3995.
42. Zbontar, J.; LeCun, Y. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.* **2016**, *17*, 2287–2318.
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
44. Cao, X.; Fang, M.; Liu, J.; Gong, N.Z. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. *arXiv* **2020**, arXiv:2012.13995.
45. Cao, X.; Jia, J.; Gong, N.Z. Provably Secure Federated Learning against Malicious Clients. *arXiv* **2021**, arXiv:2102.01854.