Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks

Jinyuan Jia, Xiaoyu Cao, Neil Zhenqiang Gong
Duke University
{jinyuan.jia, xiaoyu.cao, neil.gong}@duke.edu

Abstract

In a data poisoning attack, an attacker modifies, deletes, and/or inserts some training examples to corrupt the learnt machine learning model. Bootstrap Aggregating (bagging) is a well-known ensemble learning method, which trains multiple base models on random subsamples of a training dataset using a base learning algorithm and uses majority vote to predict labels of testing examples. We prove the intrinsic certified robustness of bagging against data poisoning attacks. Specifically, we show that bagging with an arbitrary base learning algorithm provably predicts the same label for a testing example when the number of modified, deleted, and/or inserted training examples is bounded by a threshold. Moreover, we show that our derived threshold is tight if no assumptions on the base learning algorithm are made. We evaluate our method on MNIST and CIFAR10. For instance, our method achieves a certified accuracy of 91.1% on MNIST when arbitrarily modifying, deleting, and/or inserting 100 training examples. Code is available at: https://github.com/jjy1994/BaggingCertifyDataPoisoning.

Introduction

Machine learning models trained on user-provided data are vulnerable to *data poisoning attacks* (Nelson et al. 2008; Biggio, Nelson, and Laskov 2012; Xiao et al. 2015; Li et al. 2016; Steinhardt, Koh, and Liang 2017; Shafahi et al. 2018), in which malicious users carefully poison (i.e., modify, delete, and/or insert) some training examples such that the learnt model is corrupted and makes predictions for testing examples as an attacker desires. In particular, the corrupted model predicts incorrect labels for a large fraction of testing examples indiscriminately (i.e., a large testing error rate) or for some attacker-chosen testing examples. Unlike adversarial examples (Szegedy et al. 2014; Carlini and Wagner 2017), which carefully perturb each testing example such that a model predicts an incorrect label for the perturbed testing example, data poisoning attacks corrupt the model such that it predicts incorrect labels for many clean testing examples. Like adversarial examples, data poisoning attacks pose severe security threats to machine learning systems.

To mitigate data poisoning attacks, various defenses (Cretu et al. 2008; Barreno et al. 2010; Suciu et al. 2018; Tran, Li, and Madry 2018; Feng et al. 2014; Jagielski et al. 2018; Ma, Zhu, and Hsu 2019; Wang et al. 2020; Rosenfeld et al. 2020) have been proposed in the literature. Most of these defenses (Cretu et al. 2008; Barreno et al. 2010; Suciu et al. 2018; Tran, Li, and Madry 2018; Feng et al. 2014; Jagielski et al. 2018) achieve *empirical* robustness against certain data poisoning attacks and are often broken by strong adaptive attacks. To end the cat-and-mouse game between attackers and defenders, *certified defenses* (Ma, Zhu, and Hsu 2019; Wang et al. 2020; Rosenfeld et al. 2020) were proposed. We say a learning algorithm is certifiably robust against data poisoning attacks if it can learn a classifier that provably predicts the same label for a testing example when the number of poisoned training examples is bounded. For instance, (Ma, Zhu, and Hsu 2019) showed that a classifier trained with differential privacy certifies robustness against data poisoning attacks. (Wang et al. 2020) and (Rosenfeld et al. 2020) leveraged *randomized smoothing* (Cao and Gong 2017; Cohen, Rosenfeld, and Kolter 2019), which was originally designed to certify robustness against adversarial examples, to certify robustness against data poisoning attacks that modify labels and/or features of existing training examples.

However, these certified defenses suffer from two major limitations. First, they are only applicable to limited scenarios, i.e., (Ma, Zhu, and Hsu 2019) is limited to learning algorithms that can be differentially private, while (Wang et al. 2020) and (Rosenfeld et al. 2020) are limited to data poisoning attacks that only modify existing training examples. Second, their certified robustness guarantees are loose, meaning that a learning algorithm is certifiably more robust than their guarantees indicate. We note that (Steinhardt, Koh, and Liang 2017) derives an approximate upper bound of the loss function for data poisoning attacks. However, their method cannot certify that the learnt model predicts the same label for a testing example.

We aim to address these limitations in this work. Our approach is based on a well-known ensemble learning method called *Bootstrap Aggregating (bagging)* (Breiman 1996). Given a training dataset, we create a random *subsample* with k training examples sampled from the training dataset uniformly at random with replacement. Moreover, we use a deterministic or randomized base learning algorithm to learn a base classifier on the subsample. Due to the randomness in sampling the subsample and the (randomized) base learning algorithm, the label predicted for a testing example $\mathbf{x}$ by the

learnt base classifier is random. Therefore, we define p_j as the probability that the learnt base classifier predicts label j for $\mathbf{x}$, where $j = 1, 2, \cdots, c$. We call p_j *label probability*. In bagging, the *ensemble classifier* essentially predicts the label with the largest label probability for $\mathbf{x}$.

Our first major theoretical result is that we prove the ensemble classifier in bagging predicts the same label for a testing example when the number of poisoned training examples is no larger than a threshold. We call the threshold *certified poisoning size*. Our second major theoretical result is that we prove our derived certified poisoning size is tight (i.e., it is impossible to derive a certified poisoning size larger than ours) if no assumptions on the base learning algorithm are made. Note that the certified poisoning sizes may be different for different testing examples.

Our certified poisoning size for a testing example is the optimal solution to an optimization problem, which involves the testing example's largest and second largest label probabilities predicted by the bagging's ensemble classifier. However, it is computationally challenging to compute the exact largest and second largest label probabilities, as there are an exponential number of subsamples with k training examples. To address the challenge, we propose a Monto Carlo algorithm to simultaneously estimate a lower bound of the largest label probability and an upper bound of the second largest label probability for multiple testing examples via training N base classifiers on N random subsamples. Moreover, we design an efficient algorithm to solve the optimization problem with the estimated largest and second largest label probabilities to compute certified poisoning size.

We empirically evaluate our method on MNIST and CIFAR10. For instance, our method can achieve a certified accuracy of 91.1% on MNIST when 100 training examples are arbitrarily poisoned, where $k = 100$ and $N = 1,000$. Under the same attack setting, (Ma, Zhu, and Hsu 2019), (Wang et al. 2020), and (Rosenfeld et al. 2020) achieve 0 certified accuracy on a simpler MNIST 1/7 dataset. Moreover, we show that training the base classifiers using transfer learning can significantly improve the certified accuracy.

Our contributions are summarized as follows:

- We derive the first intrinsic certified robustness of bagging against data poisoning attacks and prove the tightness of our robustness guarantee.
- We develop algorithms to compute the certified poisoning size in practice.
- We evaluate our method on MNIST and CIFAR10.

All our proofs are shown in our technical report (Jia, Cao, and Gong 2020).

Certified Robustness of Bagging

Assuming we have a training dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\}$ with n examples, where $\mathbf{x}_i$ and y_i are the feature vector and label of the ith training example, respectively. Moreover, we are given an arbitrary deterministic or randomized base learning algorithm $\mathcal{A}$, which takes a training dataset $\mathcal{D}$ as input and outputs a classifier f, i.e., $f = \mathcal{A}(\mathcal{D})$. $f(x)$ is the predicted label for a testing example $\mathbf{x}$. For convenience, we jointly represent the training and testing processes as $\mathcal{A}(\mathcal{D}, \mathbf{x})$, which is $\mathbf{x}$'s label predicted by a classifier that is trained using algorithm $\mathcal{A}$ and training dataset $\mathcal{D}$.

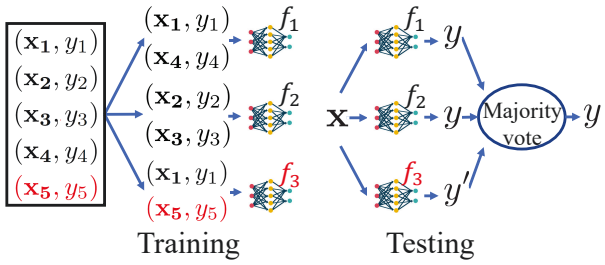


Figure 1: An example to illustrate why bagging is robust against data poisoning attacks, where ($\mathbf{x}_5$,y_5) is the poisoned training example. Base classifiers f_1 and f_2 are trained using clean training examples and bagging predicts the correct label for a testing example after majority vote among the three base classifiers.

Data poisoning attacks: In a data poisoning attack, an attacker poisons the training dataset $\mathcal{D}$ such that the learnt classifier makes predictions for testing examples as the attacker desires. In particular, the attacker can carefully *modify*, *delete*, and/or *insert* some training examples in $\mathcal{D}$ such that $\mathcal{A}(\mathcal{D}, \mathbf{x}) \neq \mathcal{A}(\mathcal{D}', \mathbf{x})$ for many testing examples $\mathbf{x}$ or some attacker-chosen $\mathbf{x}$, where $\mathcal{D}'$ is the poisoned training dataset. We note that modifying a training example means modifying its feature vector and/or label. We denote the set of poisoned training datasets with at most r poisoned training examples as follows:

$$B(\mathcal{D}, r) = \{\mathcal{D}' | \max\{|\mathcal{D}|, |\mathcal{D}'|\} - |\mathcal{D} \cap \mathcal{D}'| \leq r\}. \quad (1)$$

Intuitively, $\max\{|\mathcal{D}|, |\mathcal{D}'|\} - |\mathcal{D} \cap \mathcal{D}'|$ is the minimum number of modified/deleted/inserted training examples that can change $\mathcal{D}$ to $\mathcal{D}'$.

Bootstrap aggregating (Bagging) (Breiman 1996): Bagging is a well-known ensemble learning method. Roughly speaking, bagging creates many subsamples of a training dataset with replacement and trains a base classifier on each subsample. For a testing example, bagging uses each base classifier to predict its label and takes majority vote among the predicted labels as the label of the testing example. Figure 1 shows a toy example to illustrate why bagging certifies robustness against data poisoning attacks. When the poisoned training examples are minority in the training dataset, a majority of the subsamples do not include any poisoned training examples. Therefore, a majority of the base classifiers and the bagging's predicted labels for testing examples are not influenced by the poisoned training examples.

Next, we describe a probabilistic view of bagging, which makes it possible to theoretically analyze its certified robustness against data poisoning attacks. Specifically, we denote by $g(\mathcal{D})$ a random subsample, which is a list of k examples that are sampled from $\mathcal{D}$ with replacement uniformly at random. We use the base learning algorithm $\mathcal{A}$ to learn

a base classifier on $g(\mathcal{D})$. Due to the randomness in sampling the subsample $g(\mathcal{D})$ and the (randomized) base learning algorithm $\mathcal{A}$, the label $\mathcal{A}(g(\mathcal{D}), \mathbf{x})$ predicted by the base classifier learnt on $g(\mathcal{D})$ for $\mathbf{x}$ is random. We denote by $p_j = \text{Pr}(\mathcal{A}(g(\mathcal{D}), \mathbf{x}) = j)$ the probability that the learnt base classifier predicts label j for $\mathbf{x}$, where $j = 1, 2, \cdots, c$. We call p_j *label probability*. The *ensemble classifier* h in bagging essentially predicts the label with the largest label probability for $\mathbf{x}$, i.e., we have:

$$h(\mathcal{D}, \mathbf{x}) = \underset{j \in \{1,2,\cdots,c\}}{\text{argmax}} \; p_j, \tag{2}$$

where $h(\mathcal{D}, \mathbf{x})$ is the predicted label for $\mathbf{x}$ when the ensemble classifier h is trained on $\mathcal{D}$.

Certified robustness of bagging: We prove the certified robustness of bagging against data poisoning attacks. In particular, we show that the ensemble classifier in bagging predicts the same label for a testing example when the number of poisoned training examples is no larger than some threshold (called *certified poisoning size*). Formally, we aim to show $h(\mathcal{D}', \mathbf{x}) = h(\mathcal{D}, \mathbf{x})$ for $\forall \mathcal{D}' \in B(\mathcal{D}, r^*)$, where r^* is the certified poisoning size. For convenience, we define the following two random variables:

$$X = g(\mathcal{D}), Y = g(\mathcal{D}'), \tag{3}$$

where X and Y are two random subsamples with k examples sampled from $\mathcal{D}$ and $\mathcal{D}'$ with replacement uniformly at random, respectively. $p_j = \text{Pr}(\mathcal{A}(X, \mathbf{x}) = j)$ and $p'_j = \text{Pr}(\mathcal{A}(Y, \mathbf{x}) = j)$ are the label probabilities of label j for testing example $\mathbf{x}$ when the training dataset is $\mathcal{D}$ and its poisoned version $\mathcal{D}'$, respectively. For simplicity, we use Ω to denote the joint space of X and Y, i.e., each element in Ω is a subsample of k examples sampled from $\mathcal{D}$ or $\mathcal{D}'$ uniformly at random with replacement.

Suppose the ensemble classifier predicts label l for $\mathbf{x}$ when trained on the clean training dataset, i.e., $h(\mathcal{D}, \mathbf{x}) = l$. Our goal is to find the maximal poisoning size r such that the ensemble classifier still predicts label l for $\mathbf{x}$ when trained on the poisoned training dataset with at most r poisoned training examples. Formally, our goal is to find the maximal poisoning size r such that the following inequality is satisfied for $\forall \mathcal{D}' \in B(\mathcal{D}, r)$:

$$h(\mathcal{D}', \mathbf{x}) = l \Longleftrightarrow p'_l > \max_{j \neq l} p'_j. \tag{4}$$

However, it is challenging to compute p'_l and $\max_{j \neq l} p'_j$ due to the complicated base learning algorithm $\mathcal{A}$. To address the challenge, we aim to derive a lower bound of p'_l and an upper bound of $\max_{j \neq l} p'_j$, where the lower bound and upper bound are independent from the base learning algorithm $\mathcal{A}$ and can be easily computed for a given r. In particular, we derive the lower bound and upper bound as the probabilities that the random variable Y is in certain regions of the space Ω via the Neyman-Pearson Lemma (Neyman and Pearson 1933). Then, we can find the maximal r such that the lower bound is larger than the upper bound for any $\mathcal{D}' \in B(\mathcal{D}, r)$, and such maximal r is our certified poisoning size r^*.

Next, we show the high-level idea of our approach to derive the lower and upper bounds (details are in Supplemental Material). Our key idea is to construct regions in the space Ω such that the random variables X and Y satisfy the conditions of the Neyman-Pearson Lemma (Neyman and Pearson 1933), which enables us to derive the lower and upper bounds using the probabilities that Y is in these regions. Next, we discuss how to construct the regions. Suppose we have a lower bound $\underline{p_l}$ of the largest label probability p_l and an upper bound $\overline{p}_s$ of the second largest label probability p_s when the ensemble classifier is trained on the clean training dataset. Formally, $\underline{p_l}$ and $\overline{p}_s$ satisfy:

$$p_l \geq \underline{p_l} \geq \overline{p}_s \geq p_s = \max_{j \neq l} p_j. \tag{5}$$

We use the probability bounds instead of the exact label probabilities p_l and p_s, because it is challenging to compute them exactly. We first divide the space Ω into three regions $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{E}$, which include subsamples with k examples sampled from $\mathcal{D}$, $\mathcal{D}'$, and $\mathcal{D} \cap \mathcal{D}'$, respectively. Then, we can find a region $\mathcal{B}' \subseteq \mathcal{E}$ such that we have $\text{Pr}(X \in \mathcal{B} \cup \mathcal{B}') = \underline{p_l} - \delta_l$, where $\delta_l = \underline{p_l} - (\lfloor \underline{p_l} \cdot n^k \rfloor)/n^k$ is a small residual. We have the residual δ_l because $\text{Pr}(X \in \mathcal{B} \cup \mathcal{B}')$ is an integer multiple of $\frac{1}{n^k}$. The reason we assume we can find such region $\mathcal{B}'$ is that we aim to derive a sufficient condition. Similarly, we can find $\mathcal{C}_s \subseteq \mathcal{E}$ such that we have $\text{Pr}(X \in \mathcal{C}_s) = \overline{p}_s + \delta_s$, where $\delta_s = (\lceil \overline{p}_s \cdot n^k \rceil)/n^k - \overline{p}_s$ is a small residual. Given these regions, we leverage the Neyman-Pearson Lemma (Neyman and Pearson 1933) to derive a lower bound of p'_l and an upper bound of $\max_{j \neq l} p'_j$ as follows:

$$p'_l \geq \text{Pr}(Y \in \mathcal{B} \cup \mathcal{B}'), \tag{6}$$

$$\max_{j \neq l} p'_j \leq \text{Pr}(Y \in \mathcal{C} \cup \mathcal{C}_s), \tag{7}$$

where the lower bound $\text{Pr}(Y \in \mathcal{B} \cup \mathcal{B}')$ and upper bound $\text{Pr}(Y \in \mathcal{C} \cup \mathcal{C}_s)$ can be easily computed for a given r. Finally, we find the maximal r such that the lower bound is still larger than the upper bound, which is our certified poisoning size r^*. The following Theorem 1 formally summarizes our certified robustness guarantee of bagging.

Theorem 1 (Certified Poisoning Size of Bagging)**.** *Given a training dataset $\mathcal{D}$, a deterministic or randomized base learning algorithm $\mathcal{A}$, and a testing example $\mathbf{x}$. The ensemble classifier h in bagging is defined in Equation (2). Suppose l and s respectively are the labels with the largest and second largest label probabilities predicted by h for $\mathbf{x}$. Moreover, the probability bounds $\underline{p_l}$ and $\overline{p}_s$ satisfy (5). Then, h still predicts label l for $\mathbf{x}$ when the number of poisoned training examples is bounded by r^*, i.e., we have:*

$$h(\mathcal{D}', \mathbf{x}) = l, \forall \mathcal{D}' \in B(\mathcal{D}, r^*), \tag{8}$$

where r^* is the solution to the following optimization problem:

$$\begin{aligned} r^* = \underset{r}{\text{argmax}} \; & r \\ \textit{s.t.} \; & \max_{n-r \leq n' \leq n+r} (\frac{n'}{n})^k - 2 \cdot (\frac{\max(n, n') - r}{n})^k \\ & + 1 - (\underline{p_l} - \overline{p}_s - \delta_l - \delta_s) < 0, \end{aligned} \tag{9}$$

where $n = |\mathcal{D}|$, $n' = |\mathcal{D}'|$, $\delta_l = \underline{p_l} - (\lfloor \underline{p_l} \cdot n^k \rfloor)/n^k$, and $\delta_s = (\lceil \overline{p}_s \cdot n^k \rceil)/n^k - \overline{p}_s$.

Given Theorem 1, we have the following corollaries.

Corollary 1. *Suppose a data poisoning attack only modifies existing training examples. Then, we have $n' = n$ and the solution to optimization problem (9) is $r^* = \lceil n \cdot (1 - \sqrt[k]{1 - \frac{\underline{p_l} - \overline{p}_s - \delta_l - \delta_s}{2}}) - 1 \rceil$.*

Corollary 2. *Suppose a data poisoning attack only deletes existing training examples. Then, we have $n' = n - r$ and $r^* = \lceil n \cdot (1 - \sqrt[k]{1 - (\underline{p_l} - \overline{p}_s - \delta_l - \delta_s)}) - 1 \rceil$.*

Corollary 3. *Suppose a data poisoning attack only inserts new training examples. Then, we have $n' = n + r$ and $r^* = \lceil n \cdot (\sqrt[k]{1 + (\underline{p_l} - \overline{p}_s - \delta_l - \delta_s)} - 1) - 1 \rceil$.*

The next theorem shows that our derived certified poisoning size is tight.

Theorem 2 (Tightness of the Certified Poisoning Size)**.** *Assuming we have $\underline{p_l} + \overline{p}_s \leq 1$, $\underline{p_l} + (c-1) \cdot \overline{p}_s \geq 1$, and $\delta_l = \delta_s = 0$. Then, for any $r > r^*$, there exist a base learning algorithm $\mathcal{A}^*$ consistent with (5) and a poisoned training dataset $\mathcal{D}'$ with r poisoned training examples such that $h(\mathcal{D}', \mathbf{x}) \neq l$ or there exist ties.*

We have several remarks about our theorems.

Remark 1: Our Theorem 1 is applicable for any base learning algorithm $\mathcal{A}$, i.e., bagging with any base learning algorithm is provably robust against data poisoning attacks.

Remark 2: For any lower bound $\underline{p_l}$ of the largest label probability and upper bound $\overline{p}_s$ of the second largest label probability, Theorem 1 derives a certified poisoning size. Moreover, our certified poisoning size is related to the gap between the two probability bounds. If we can estimate tighter probability bounds, then the certified poisoning size may be larger.

Remark 3: Theorem 2 shows that when no assumptions on the base learning algorithm are made, it is impossible to certify a poisoning size that is larger than ours.

Computing the Certified Poisoning Size

Given a base learning algorithm $\mathcal{A}$, a training dataset $\mathcal{D}$, subsampling size k, and e testing examples in $\mathcal{D}_e$, we aim to compute the label l_i predicted by the ensemble classifier and the corresponding certified poisoning size r_i^* for each testing example $\mathbf{x}_i$. For a testing example $\mathbf{x}_i$, our certified poisoning size relies on a lower bound $\underline{p_{l_i}}$ of the largest label probability and an upper bound $\overline{p}_{s_i}$ of the second largest label probability. We design a Monte-Carlo algorithm to estimate the probability bounds for the e testing examples simultaneously via training N base classifiers. Next, we first describe estimating the probability bounds. Then, we describe our efficient algorithm to solve the optimization problem in (9) with the estimated probability bounds to compute the certified poisoning sizes.

Computing the predicted label and probability bounds for one testing example: We first discuss estimating the predicted label l_i and probability bounds $\underline{p_{l_i}}$ and $\overline{p}_{s_i}$ for one testing example $\mathbf{x}_i$. We first randomly sample N subsamples $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_N$ from $\mathcal{D}$ with replacement, each of which has k training examples. Then, we train a base classifier f_o for each subsample $\mathcal{L}_o$ using the base learning algorithm $\mathcal{A}$, where $o = 1, 2, \cdots, N$. We use the base classifiers to predict labels for $\mathbf{x}_i$, and we denote by N_j the frequency of label j, i.e., N_j is the number of base classifiers that predict label j for $\mathbf{x}_i$. We estimate the label with the largest frequency as the label l_i predicted by the ensemble classifier h for $\mathbf{x}_i$. Moreover, based on the definition of label probability, the frequency N_j of the label j among the N base classifiers follows a binomial distribution with parameters N and p_j. Therefore, given the label frequencies, we can use the Clopper-Pearson (Clopper and Pearson 1934) based method called SimuEM (Jia et al. 2020a) to estimate the following probability bounds simultaneously:

$$\underline{p_{l_i}} = \text{Beta}(\frac{\alpha}{c}; N_{l_i}, N - N_{l_i} + 1) \tag{10}$$

$$\overline{p}_j = \text{Beta}(1 - \frac{\alpha}{c}; N_j, N - N_j + 1), \forall j \neq l_i, \tag{11}$$

where $1 - \alpha$ is the confidence level and $\text{Beta}(\beta; \lambda, \theta)$ is the βth quantile of the Beta distribution with shape parameters λ and θ. One natural method to estimate $\overline{p}_{s_i}$ is that $\overline{p}_{s_i} = \max_{j \neq l_i} \overline{p}_j$. However, this bound may be loose. For example, $\underline{p_{l_i}} + \overline{p}_{s_i}$ may be larger than 1. Therefore, we estimate $\overline{p}_{s_i}$ as $\overline{p}_{s_i} = \min(\max_{j \neq l_i} \overline{p}_j, 1 - \underline{p_{l_i}})$.

Computing the predicted labels and probability bounds for e testing examples: One way of estimating the predicted labels and probability bounds for e testing examples is to apply the above process for each testing example separately. However, such process requires training N base classifiers for each testing example, which is computationally intractable. To address the challenge, we propose a method to estimate them for e testing examples simultaneously via training N base classifiers in total. Our key idea is to divide the confidence level among the e testing examples such that we can estimate their predicted labels and probability bounds using the same N base classifiers with a simultaneous confidence level at least $1 - \alpha$. Specifically, we still use the N base classifiers to predict the label for each testing example as we described above. Then, we follow the above process to estimate the probability bounds $\underline{p_{l_i}}$ and $\overline{p}_{s_i}$ for a testing example $\mathbf{x}_i$ via replacing α as α/e in Equation (10) and (11). Based on the *Bonferroni correction*, the simultaneous confidence level of estimating the probability bounds for the e testing examples is at least $1 - \alpha$.

Computing the certified poisoning sizes: Given the estimated probability bounds $\underline{p_{l_i}}$ and $\overline{p}_{s_i}$ for a testing example $\mathbf{x}_i$, we solve the optimization problem in (9) to obtain its certified poisoning size r_i^*. We design an efficient binary search based method to solve r_i^*. Specifically, we use binary search to find the largest r such that the constraint in (9) is satisfied. We denote the left-hand side of the constraint as $\max_{n-r \leq n' \leq n+r} L(n)$. For a given r, a naive way to check whether the constraint $\max_{n-r \leq n' \leq n+r} L(n') < 0$ holds is to check whether $L(n') < 0$ holds for each n' in the range $[n - r, n + r]$, which could be inefficient when r is large. To reduce the computation cost, we derive an analytical form of n' at which $L(n')$ reaches its maximum value. Our analytical form enables us to only check whether $L(n') < 0$ holds for at most two different n' for a given r. The details of deriving the analytical form are shown in Supplemental Material.

Algorithm 1 CERTIFY

Input: $\mathcal{A}$, $\mathcal{D}$, k, N, $\mathcal{D}_e$, α.
Output: Predicted label and certified poisoning size for each testing example.
$f_1, f_2, \cdots, f_N \leftarrow$ TRAINUNDERSAMPLE$(\mathcal{A}, \mathcal{D}, k, N)$
for $\mathbf{x}_i$ **in** $\mathcal{D}_e$ **do**
  counts$[j] \leftarrow \sum_{o=1}^{N} \mathbb{I}(f_o(\mathbf{x}_i) = j), j \in \{1, 2, \cdots, c\}$
  $l_i, s_i \leftarrow$ top two indices in counts (ties are broken uniformly at random).
  $\underline{p_{l_i}}, \overline{p}_{s_i} \leftarrow$ SIMUEM(counts, $\frac{\alpha}{e}$)
  if $\underline{p_{l_i}} > \overline{p}_{s_i}$ **then**
    $r_i^* \leftarrow$ BINARYSEARCH$(\underline{p_{l_i}}, \overline{p}_{s_i}, k, |\mathcal{D}|)$
  else
    $l_i, r_i^* \leftarrow$ ABSTAIN, ABSTAIN
  end if
end for
return $l_1, l_2, \cdots, l_e$ and $r_1^*, r_2^*, \cdots, r_e^*$

Complete certification algorithm: Algorithm 1 shows our certification process to estimate the predicted labels and certified poisoning sizes for e testing examples in $\mathcal{D}_e$. The function TRAINUNDERSAMPLE randomly samples N subsamples and trains N base classifiers. The function SIMUEM estimates the probability bounds $\underline{p_{l_i}}$ and $\overline{p}_{s_i}$ with confidence level $1 - \frac{\alpha}{e}$. The function BINARYSEARCH solves the optimization problem in (9) using the estimated probability bounds $\underline{p_{l_i}}$ and $\overline{p}_{s_i}$ to obtain the certified poisoning size r_i^* for testing example $\mathbf{x}_i$.

Since the probability bounds are estimated using a Monte Carlo algorithm, they may be estimated incorrectly, i.e., $\underline{p_{l_i}} > p_{l_i}$ or $\overline{p}_{s_i} < p_{s_i}$. When they are estimated incorrectly, our algorithm CERTIFY may output an incorrect certified poisoning size. However, the following theorem shows that the probability that CERTIFY returns an incorrect certified poisoning size for at least one testing example is at most α.

Theorem 3. *The probability that* CERTIFY *returns an incorrect certified poisoning size for at least one testing example in $\mathcal{D}_e$ is at most α, i.e., we have:*

$$Pr(\cap_{\mathbf{x}_i \in \mathcal{D}_e}((\forall \mathcal{D}' \in B(\mathcal{D}, r_i^*), h(\mathcal{D}', \mathbf{x}_i) = l_i) | l_i \neq \textit{ABSTAIN})) \geq 1 - \alpha. \quad (12)$$

Experiments

Experimental Setup

Datasets and classifiers: We use MNIST and CIFAR10 datasets. The base learning algorithm is neural network, and we use the example convolutional neural network architecture and ResNet20 (He et al. 2016) in Keras for MNIST and CIFAR10, respectively. The number of training examples in the two datasets are $60,000$ and $50,000$, respectively, which are the training datasets that we aim to certify. Both datasets have 10,000 testing examples, which are the $\mathcal{D}_e$ in our algorithm. When we train a base classifier, we adopt the example data augmentation in Keras for both datasets.

Evaluation metric: We use *certified accuracy* as our evaluation metric. In particular, we define the certified accuracy at r poisoned training examples of a classifier as the fraction of testing examples whose labels are correctly predicted by the classifier and whose certified poisoning sizes are at least r. Formally, we have the certified accuracy CA_r at r poisoned training examples as follows:

$$CA_r = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}_e} \mathbb{I}(l_i = y_i) \cdot \mathbb{I}(r_i^* \geq r)}{|\mathcal{D}_e|}, \quad (13)$$

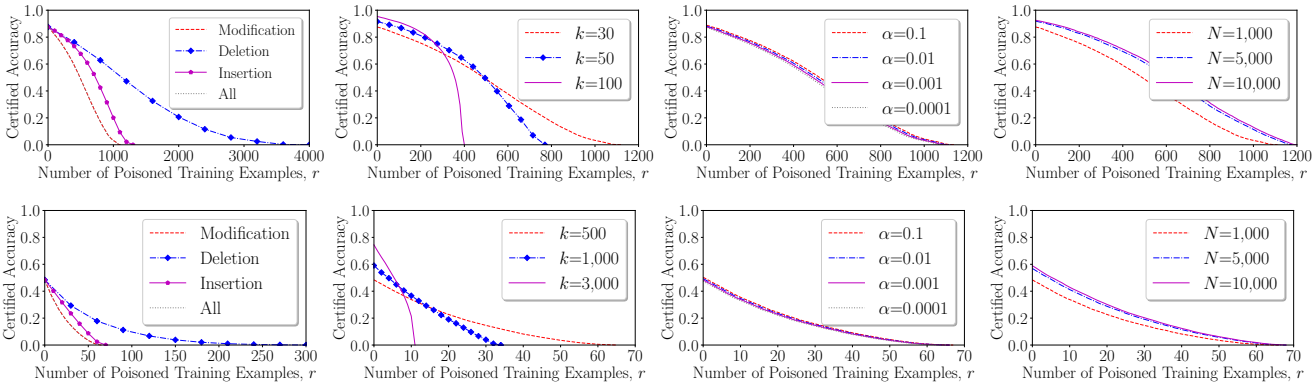
where y_i is the ground truth label for testing example $\mathbf{x}_i$, and l_i and r_i^* respectively are the label predicted by the classifier and the corresponding certified poisoning size for $\mathbf{x}_i$. Intuitively, CA_r of a classifier means that, when the number of poisoned training examples is r, the classifier's testing accuracy for $\mathcal{D}_e$ is at least CA_r no matter how the attacker manipulates the r poisoned training examples. Based on Theorem 3, the CA_r computed using the predicted labels and certified poisoning sizes outputted by our CERTIFY algorithm has a confidence level $1 - \alpha$.

Parameter setting: Our method has three parameters, i.e., k, α, and N. Unless otherwise mentioned, we adopt the following default settings for them: $\alpha = 0.001$, $N = 1,000$, $k = 30$ for MNIST, and $k = 500$ for CIFAR10. We will study the impact of each parameter while setting the remaining parameters to their default values. Note that training the N base classifiers can be easily parallelized. We performed experiments on a server with 80 CPUs@2.1GHz, 8 GPUs (RTX 6,000), and 385 GB main memory.

Experimental Results

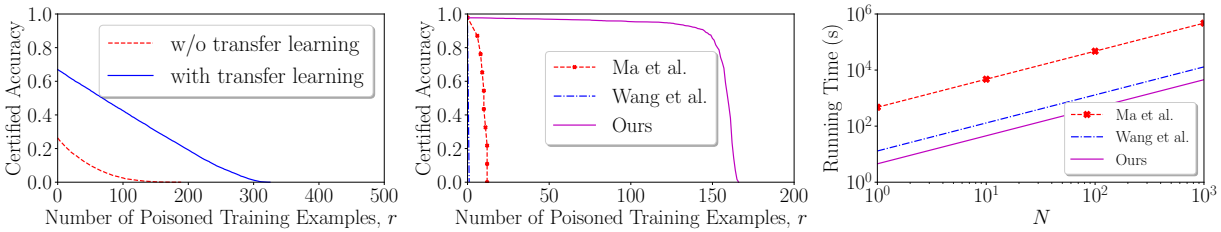
Comparing different data poisoning attacks: An attacker can modify, delete, and/or insert training examples in data poisoning attacks. We compare the certified accuracy of our method when an attacker only modifies, deletes, or inserts training examples. Our Corollary 1-3 show the certified poisoning sizes for such attacks. Figure 2(a) shows the comparison results, where "All" corresponds to the attacks that can use modification, deletion, and insertion. Our method achieves the best certified accuracy for attacks that only delete training examples. This is because deletion simply reduces the size of the clean training dataset. The curves corresponding to Modification and All overlap and have the lowest certified accuracy. This is because modifying a training example is equivalent to deleting an existing training example and inserting a new one. In the following experiments, we use the All attacks unless otherwise mentioned.

Impact of k, α, and N: Figure 2 shows the impact of k, α, and N on the certified accuracy of our method. As the results show, k controls a tradeoff between accuracy under no poisoning and robustness. Specifically, when k is larger, our method has a higher accuracy when there are no data poisoning attacks (i.e., $r = 0$) but the certified accuracy drops more quickly as the number of poisoned training examples increases. The reason is that a larger k makes it more likely to sample poisoned training examples when creating the subsamples in bagging. The certified accuracy increases as α or N increases. The reason is that a larger α or N produces tighter estimated probability bounds, which make the certified poisoning sizes larger. We also observe that the certified accuracy is relatively insensitive to α.



(a) Different attacks (b) Impact of k (c) Impact of α (d) Impact of N

Figure 2: (a) Comparing different data poisoning attacks. (b)-(d) Impact of k, α, and N on the certified accuracy of our method. The first row is the result on MNIST and the second row is the result on CIFAR10.



(a) Transfer learning (b) Comparing certified accuracy (c) Comparing running time

Figure 3: (a) Transfer learning improves our certified accuracy on CIFAR10. Comparing our method with existing methods with respect to (b) certified accuracy and (c) running time on the MNIST 1/7 dataset.

Transfer learning improves certified accuracy: Our method trains multiple base classifiers and each base classifier is trained using k training examples. Improving the accuracy of each base classifier can improve the certified accuracy. We explore using transfer learning to train more accurate base classifiers. Specifically, we use the Inception-v3 classifier pretrained on ImageNet to extract features and we use a public implementation[1] to train our base classifiers on CIFAR10. Figure 3(a) shows that transfer learning can significantly increase our certified accuracy, where $k = 100$, $\alpha = 0.001$, and $N = 1,000$. Note that we assume the pre-trained classifier is not poisoned in this experiment.

Comparing with (Ma, Zhu, and Hsu 2019), (Wang et al. 2020), and (Rosenfeld et al. 2020): Since these methods are not scalable because they train N classifiers on the entire training dataset, we perform comparisons on the MNIST 1/7 dataset that just includes digits 1 and 7. This subset includes 13,007 training examples and 2,163 testing examples. Note that our above experiments used the entire MNIST dataset.

- **(Ma, Zhu, and Hsu 2019).** Ma et al. showed that a classifier trained with differential privacy achieves certified robustness against data poisoning attacks. Suppose ACC_r is the testing accuracy for $\mathcal{D}_e$ of a differentially private classifier trained on a poisoned training dataset with r poisoned training examples. Based on the Theorem 3 in (Ma, Zhu, and Hsu 2019), we have the expected testing accuracy $E(ACC_r)$ is lower bounded by a certain function of $E(ACC)$, r, and (ϵ, δ) (the function can be found in their Theorem 3), where $E(ACC)$ is the expected testing accuracy of a differentially private classifier that is trained using the clean training dataset and (ϵ, δ) are the differential privacy parameters. The randomness in $E(ACC_r)$ and $E(ACC)$ are from differential privacy. This lower bound is the certified accuracy that the method achieves. A lower bound of $E(ACC)$ can be further estimated with confidence level $1 - \alpha$ via training N differentially private classifiers on the entire clean training dataset. For simplicity, we estimate $E(ACC)$ as the average testing accuracies of the N differentially private classifiers, which gives advantages for this method. We use DP-SGD (Abadi et al. 2016) implemented in TensorFlow to train differentially private classifiers. Moreover, we set $\epsilon = 0.3$ and $\delta = 10^{-5}$ such that this method and our method achieve comparable certified accuracies when $r = 0$.

- **(Wang et al. 2020) and (Rosenfeld et al. 2020).** Wang et al. proposed a randomized smoothing based method to certify robustness against backdoor attacks via randomly flipping features and labels of training examples as well as features of testing examples. Rosenfeld et al. leveraged randomized smoothing to certify robustness against label flipping attacks. Both methods can be generalized to certify robustness against data poisoning attacks that modify

[1] https://github.com/alexisbcook/keras_transfer_cifar10

both features and labels of existing training examples via randomly flipping features and labels of training examples. Moreover, the two methods become the same after such generalization. Therefore, we only show results for (Wang et al. 2020). In particular, we binarize the features to apply this method. We train N classifiers to estimate the certified accuracy with a confidence level $1-\alpha$. Unlike our method, when training a classifier, they flip each feature/label value in the training dataset with probability β and use the entire noisy training dataset. When predicting the label of a testing example, this method takes a majority vote among the N classifiers. We set $\beta = 0.3$ such that this method and our method achieve comparable certified accuracies when $r = 0$. We note that this method certifies the number of poisoned features/labels in the training dataset. We transform this certificate to the number of poisoned training examples as $\lfloor \frac{F}{d+1} \rfloor$, where F is the certified number of features/labels and $d+1$ is the number of features/label of a training example (d features + one label). We have $d = 784$ for MNIST.

Figure 3(b) shows the comparison results, where $k = 50$, $\alpha = 0.001$, and $N = 1,000$. To be consistent with previous work, we did not use data augmentation when training the base classifiers for all three methods in these experiments. Our method significantly outperforms existing methods. For example, our method can achieve 96.95% certified accuracy when the number of poisoned training examples is $r = 50$, while the certified accuracy is 0 under the same setting for existing methods. Figure 3(c) shows that our method is also more efficient than existing methods. This is because our method trains base classifiers on a small number of training examples while existing methods train classifiers on the entire training dataset. Ma et al. outperforms Wang et al. and Rosenfeld et al. because differential privacy directly certifies robustness against modification/deletion/insertion of training examples while randomized smoothing was designed to certify robustness against modifications of features/labels.

Related Work

Data poisoning attacks carefully modify, delete, and/or insert some training examples in the training dataset such that a learnt model makes incorrect predictions for many testing examples indiscriminately (i.e., the learnt model has a large testing error rate) or for some attacker-chosen testing examples. For instance, data poisoning attacks have been shown to be effective for Bayes classifiers (Nelson et al. 2008), SVMs (Biggio, Nelson, and Laskov 2012), neural networks (Yang et al. 2017; Muñoz-González et al. 2017; Suciu et al. 2018; Shafahi et al. 2018), linear regression models (Mei and Zhu 2015b; Jagielski et al. 2018), PCA (Rubinstein et al. 2009), LASSO (Xiao et al. 2015), collaborative filtering (Li et al. 2016; Yang, Gong, and Cai 2017; Fang et al. 2018; Fang, Gong, and Liu 2020), clustering (Biggio et al. 2013, 2014), graph-based methods (Zügner, Akbarnejad, and Günnemann 2018; Wang and Gong 2019; Jia et al. 2020b; Zhang et al. 2020), federated learning (Fang et al. 2020; Bhagoji et al. 2019; Bagdasaryan et al. 2020), and others (Mozaffari-Kermani et al. 2014; Mei and Zhu 2015a; Koh, Steinhardt, and Liang 2018; Zhu et al. 2019). We note that backdoor attacks (Gu et al. 2019; Liu et al. 2018) also poison the training dataset. However, unlike data poisoning attacks, backdoor attacks also inject perturbation (i.e., a trigger) to testing examples.

One category of defenses (Cretu et al. 2008; Barreno et al. 2010; Suciu et al. 2018; Tran, Li, and Madry 2018) aim to detect the poisoned training examples based on their negative impact on the error rate of the learnt model. Another category of defenses (Feng et al. 2014; Jagielski et al. 2018) aim to design new loss functions, solving which detects the poisoned training examples and learns a model simultaneously. For instance, (Jagielski et al. 2018) proposed to jointly optimize the selection of a subset of training examples with a given size and a model that minimizes the loss function; and the unselected training examples are treated as poisoned ones. (Steinhardt, Koh, and Liang 2017) assumes that a model is trained only using examples in a feasible set and derives an approximate upper bound of the loss function for any data poisoning attacks under these assumptions. However, all of these defenses cannot certify that the learnt model predicts the same label for a testing example under data poisoning attacks.

(Ma, Zhu, and Hsu 2019) shows that differentially private models certify robustness against data poisoning attacks. (Wang et al. 2020) proposes to use randomized smoothing to certify robustness against backdoor attacks, which is also applicable to certify robustness against data poisoning attacks. (Rosenfeld et al. 2020) leverages randomized smoothing to certify robustness against label flipping attacks. However, these defenses achieve loose certified robustness guarantees. Moreover, (Ma, Zhu, and Hsu 2019) is only applicable to learning algorithms that can be differentially private, while (Wang et al. 2020) and (Rosenfeld et al. 2020) are only applicable to data poisoning attacks that modify existing training examples. (Biggio et al. 2011) proposed bagging as an empirical defense against data poisoning attacks. However, they did not derive the certified robustness of bagging. We note that a concurrent work (Levine and Feizi 2021) proposed to certify robustness against data poisoning attacks via partitioning the training dataset using a hash function. However, their results are only applicable to deterministic learning algorithms.

Conclusion

Data poisoning attacks pose severe security threats to machine learning systems. In this work, we show the intrinsic certified robustness of bagging against data poisoning attacks. Specifically, we show that bagging predicts the same label for a testing example when the number of poisoned training examples is bounded. Moreover, we show that our derived bound is tight if no assumptions on the base learning algorithm are made. We also empirically demonstrate the effectiveness of our method using MNIST and CIFAR10. Our results show that our method achieves much better certified robustness and is more efficient than existing certified defenses. Interesting future work includes: 1) generalizing our method to other types of data, e.g., graphs, and 2) improving our method by leveraging meta-learning.

Acknowledgments

We thank the anonymous reviewers for insightful reviews. This work was supported by NSF grant No. 1937786.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 308–318.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, 2938–2948.

Barreno, M.; Nelson, B.; Joseph, A. D.; and Tygar, J. D. 2010. The security of machine learning. *Machine Learning* 81(2): 121–148.

Bhagoji, A. N.; Chakraborty, S.; Mittal, P.; and Calo, S. 2019. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, 634–643.

Biggio, B.; Corona, I.; Fumera, G.; Giacinto, G.; and Roli, F. 2011. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *International workshop on multiple classifier systems*, 350–359. Springer.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 1467–1474.

Biggio, B.; Pillai, I.; Rota Bulò, S.; Ariu, D.; Pelillo, M.; and Roli, F. 2013. Is data clustering in adversarial settings secure? In *Proceedings of the 2013 ACM workshop on Artificial intelligence and security*, 87–98.

Biggio, B.; Rieck, K.; Ariu, D.; Wressnegger, C.; Corona, I.; Giacinto, G.; and Roli, F. 2014. Poisoning behavioral malware clustering. In *Proceedings of the 2014 workshop on artificial intelligent and security workshop*, 27–36.

Breiman, L. 1996. Bagging predictors. *Machine learning* 24(2): 123–140.

Cao, X.; and Gong, N. Z. 2017. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, 278–287.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. IEEE.

Clopper, C. J.; and Pearson, E. S. 1934. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 26(4): 404–413.

Cohen, J.; Rosenfeld, E.; and Kolter, Z. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 1310–1320. PMLR.

Cretu, G. F.; Stavrou, A.; Locasto, M. E.; Stolfo, S. J.; and Keromytis, A. D. 2008. Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE Symposium on Security and Privacy*, 81–95. IEEE.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. Z. 2020. Local model poisoning attacks to Byzantine-robust federated learning. In *Usenix Security Symposium*, 1605–1622.

Fang, M.; Gong, N. Z.; and Liu, J. 2020. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, 3019–3025.

Fang, M.; Yang, G.; Gong, N. Z.; and Liu, J. 2018. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference*, 381–392.

Feng, J.; Xu, H.; Mannor, S.; and Yan, S. 2014. Robust logistic regression and classification. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 1*, 253–261.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7: 47230–47244. doi:10.1109/ACCESS.2019.2909068.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jagielski, M.; Oprea, A.; Biggio, B.; Liu, C.; Nita-Rotaru, C.; and Li, B. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy*, 19–35. IEEE.

Jia, J.; Cao, X.; and Gong, N. 2020. Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks. In *arXiv: Cryptography and Security*. URL https://arxiv.org/pdf/2008.04495.pdf.

Jia, J.; Cao, X.; Wang, B.; and Gong, N. Z. 2020a. Certified Robustness for Top-k Predictions against Adversarial Perturbations via Randomized Smoothing. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=BkeWw6VFwr.

Jia, J.; Wang, B.; Cao, X.; and Gong, N. Z. 2020b. Certified Robustness of Community Detection against Adversarial Structural Perturbation via Randomized Smoothing. In *Proceedings of The Web Conference 2020*, 2718–2724.

Koh, P. W.; Steinhardt, J.; and Liang, P. 2018. Stronger Data Poisoning Attacks Break Data Sanitization Defenses. volume abs/1811.00741. URL https://arxiv.org/pdf/1811.00741.pdf.

Levine, A.; and Feizi, S. 2021. Deep Partition Aggregation: Provable Defenses against General Poisoning Attacks. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=YUGG2tFuPM.

Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *Advances in neural information processing systems*, 1885–1893.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2018. Trojaning Attack on Neural Networks. In *25nd Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-221, 2018*. The Internet Society.

Ma, Y.; Zhu, X.; and Hsu, J. 2019. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *International Joint Conference on Artificial Intelligence*.

Mei, S.; and Zhu, X. 2015a. The security of latent dirichlet allocation. In *Artificial Intelligence and Statistics*, 681–689. PMLR.

Mei, S.; and Zhu, X. 2015b. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2871–2877.

Mozaffari-Kermani, M.; Sur-Kolay, S.; Raghunathan, A.; and Jha, N. K. 2014. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE journal of biomedical and health informatics* 19(6): 1893–1905.

Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E. C.; and Roli, F. 2017. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 27–38.

Nelson, B.; Barreno, M.; Chi, F. J.; Joseph, A. D.; Rubinstein, B. I.; Saini, U.; Sutton, C. A.; Tygar, J. D.; and Xia, K. 2008. Exploiting Machine Learning to Subvert Your Spam Filter. *LEET* 8: 1–9.

Neyman, J.; and Pearson, E. S. 1933. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 231(694-706): 289–337.

Rosenfeld, E.; Winston, E.; Ravikumar, P.; and Kolter, Z. 2020. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, 8230–8241. PMLR.

Rubinstein, B. I.; Nelson, B.; Huang, L.; Joseph, A. D.; Lau, S.-h.; Rao, S.; Taft, N.; and Tygar, J. D. 2009. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, 1–14.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 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.

Steinhardt, J.; Koh, P. W. W.; and Liang, P. S. 2017. Certified defenses for data poisoning attacks. In *Advances in neural information processing systems*, 3517–3529.

Suciu, O.; Marginean, R.; Kaya, Y.; III, H. D.; and Dumitras, T. 2018. When Does Machine Learning FAIL? Generalized Transferability for Evasion and Poisoning Attacks. In *Usenix Security Symposium*, 1299–1316.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=kklr_MTHMRQjG.

Tran, B.; Li, J.; and Madry, A. 2018. Spectral Signatures in Backdoor Attacks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 8011–8021.

Wang, B.; Cao, X.; Jia, J.; and Gong, N. Z. 2020. On Certifying Robustness against Backdoor Attacks via Randomized Smoothing. In *CVPR 2020 Workshop on Adversarial Machine Learning in Computer Vision*.

Wang, B.; and Gong, N. Z. 2019. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2023–2040.

Xiao, H.; Biggio, B.; Brown, G.; Fumera, G.; Eckert, C.; and Roli, F. 2015. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, 1689–1698. PMLR.

Yang, C.; Wu, Q.; Li, H.; and Chen, Y. 2017. Generative Poisoning Attack Method Against Neural Networks. *CoRR* abs/1703.01340. URL http://arxiv.org/abs/1703.01340.

Yang, G.; Gong, N. Z.; and Cai, Y. 2017. Fake Co-visitation Injection Attacks to Recommender Systems. In *24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017*. The Internet Society. URL https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/fake-co-visitation-injection-attacks-recommender-systems/.

Zhang, Z.; Jia, J.; Wang, B.; and Gong, N. Z. 2020. Backdoor attacks to graph neural networks. In *NeurIPS 2020 Workshop on Dataset Curation and Security*. URL https://arxiv.org/pdf/2006.11165.pdf.

Zhu, C.; Huang, W. R.; Li, H.; Taylor, G.; Studer, C.; and Goldstein, T. 2019. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *International Conference on Machine Learning*, 7614–7623.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2847–2856.