

# Certified Robustness of Nearest Neighbors against Data Poisoning and Backdoor Attacks

Jinyuan Jia, Yupei Liu, Xiaoyu Cao, Neil Zhenqiang Gong

Duke University  
{jinyuan.jia, yupei.liu, xiaoyu.cao, neil.gong}@duke.edu

## Abstract

Data poisoning attacks and backdoor attacks aim to corrupt a machine learning classifier via modifying, adding, and/or removing some carefully selected training examples, such that the corrupted classifier makes incorrect predictions as the attacker desires. The key idea of state-of-the-art certified defenses against data poisoning attacks and backdoor attacks is to create a *majority vote* mechanism to predict the label of a testing example. Moreover, each voter is a base classifier trained on a subset of the training dataset. Classical simple learning algorithms such as  $k$  nearest neighbors (kNN) and radius nearest neighbors (rNN) have intrinsic majority vote mechanisms. In this work, we show that the intrinsic majority vote mechanisms in kNN and rNN already provide certified robustness guarantees against data poisoning attacks and backdoor attacks. Moreover, our evaluation results on MNIST and CIFAR10 show that the intrinsic certified robustness guarantees of kNN and rNN outperform those provided by state-of-the-art certified defenses. Our results serve as standard baselines for future certified defenses against data poisoning attacks and backdoor attacks.

## Introduction

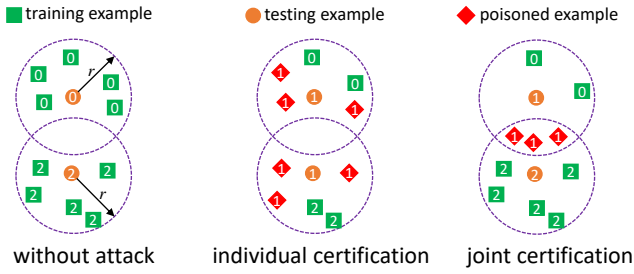
Data poisoning attacks and backdoor attacks (Barreno et al. 2006; Nelson et al. 2008; Biggio, Nelson, and Laskov 2012; Biggio, Fumera, and Roli 2013; Xiao et al. 2015b; Steinhart, Koh, and Liang 2017; Gu, Dolan-Gavitt, and Garg 2017; Chen et al. 2017; Liu et al. 2017; Shafahi et al. 2018) aim to corrupt the training phase of a machine learning system via carefully poisoning its training dataset including modifying, adding, and/or removing some training examples. Specifically, in data poisoning attacks, the corrupted downstream classifier makes incorrect predictions for clean testing inputs; and in backdoor attacks, the corrupted downstream classifier makes incorrect predictions for testing inputs embedded with a certain trigger. Data poisoning attacks and backdoor attacks pose severe security concerns to machine learning in critical application domains such as autonomous driving (Gu, Dolan-Gavitt, and Garg 2017), cybersecurity (Rubinstein et al. 2009; Suci et al. 2018; Chen et al. 2017), and healthcare analytics (Mozaffari-Kermani et al. 2014).

Multiple certifiably robust learning algorithms (Ma, Zhu, and Hsu 2019; Rosenfeld et al. 2020; Levine and Feizi 2021;

Jia, Cao, and Gong 2021) against data poisoning attacks and backdoor attacks were recently developed. A learning algorithm is certifiably robust against data poisoning attacks and backdoor attacks if it can learn a classifier on a training dataset that achieves a *certified accuracy* on a testing dataset when the number of poisoned training examples is no more than a threshold (called *poisoning size*). The certified accuracy of a learning algorithm is a lower bound of the accuracy of its learnt classifier no matter how an attacker poisons the training examples with the given poisoning size.

The key idea of state-of-the-art certifiably robust learning algorithms (Jia, Cao, and Gong 2021; Levine and Feizi 2021) is to create a *majority vote* mechanism to predict the label of a testing example. In particular, each voter votes a label for a testing example and the final predicted label is the majority vote among multiple voters. For instance, Bagging (Jia, Cao, and Gong 2021) learns multiple base classifiers (i.e., voters), where each of them is learnt on a random subsample of the training dataset. Deep Partition Aggregation (DPA) (Levine and Feizi 2021) divides the training dataset into disjoint partitions and learns a base classifier (i.e., a voter) on each partition. We denote by  $a$  and  $b$  the labels with the largest and second largest number of votes, respectively. Moreover,  $s_a$  and  $s_b$  respectively are the number of votes for labels  $a$  and  $b$  when there are no corrupted voters. The corrupted voters change their votes from  $a$  to  $b$  in the worst-case scenario. Therefore, the majority vote result (i.e., the predicted label for a testing example) remains to be  $a$  when the number of corrupted voters is no larger than  $\lceil \frac{s_a - s_b}{2} \rceil - 1$ . In other words, the number of corrupted voters that a majority vote mechanism can tolerate depends on the gap  $s_a - s_b$  between the largest and the second largest number of votes.

However, state-of-the-art certifiably robust learning algorithms achieve suboptimal certified accuracies due to two key limitations. First, *each poisoned training example leads to multiple corrupted voters* in the worst-case scenarios. In particular, modifying a training example corrupts the voters whose training subsamples include the modified training example in bagging (Jia, Cao, and Gong 2021) and corrupts two voters (i.e., two base classifiers) in DPA (Levine and Feizi 2021). Therefore, given the same gap  $s_a - s_b$  between the largest and the second largest number of votes, the majority vote result is robust against a small number of poisoned training examples. Second, *they can only certify robustness*



**Figure 1: An example to illustrate individual certification vs. joint certification.** Suppose rNN correctly classifies the two testing examples without attack. An attacker can poison 3 training examples. The attacker can make rNN misclassify each testing example individually. However, the attacker cannot make rNN misclassify both testing examples jointly.

for each testing example individually because it is hard to quantify how poisoned training examples corrupt the voters for different testing examples jointly. Suppose the classifier learnt by a learning algorithm can correctly classify testing inputs  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . An attacker can poison  $e$  training examples such that the learnt classifier misclassifies  $\mathbf{x}_1$  or  $\mathbf{x}_2$ , but the attacker cannot poison  $e$  training examples such that both  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are misclassified. When the poisoning size is  $e$ , existing certifiably robust learning algorithms would produce a certified accuracy of 0 for the two testing examples. However, the certified accuracy can be  $1/2$  if we consider them jointly. We note that Steinhardt, Koh, and Liang (2017) derives an approximate upper bound of the loss function under data poisoning attacks. However, their method cannot certify the learnt model predicts the same label for a testing example.

$k$  nearest neighbors (kNN) and radius nearest neighbors (rNN) (Fix and Hodges 1951; Cover and Hart 1967) are well-known classic learning algorithms. With good feature representation (e.g., those learnt via self-supervised learning), kNN and rNN can achieve classification accuracy comparable to those of complex learning algorithms such as neural networks (He et al. 2020). kNN and rNN have intrinsic majority vote mechanisms. Specifically, given a testing example, kNN (or rNN) predicts its label via taking a majority vote among the labels of its  $k$  nearest neighbors (or neighbors within radius  $r$ ) in the training dataset. Our major contribution in this work is that we show the intrinsic majority vote mechanisms in kNN and rNN make them certifiably robust against data poisoning attacks. Moreover, kNN and rNN address the limitations of state-of-the-art certifiably robust learning algorithms. Specifically, each poisoned training example leads to only one corrupted voter in the worst-case scenario in kNN and rNN. Thus, given the same gap  $s_a - s_b$ , the majority vote result (i.e., predicted label for a testing example) is robust against more poisoned training examples in kNN and rNN.

Furthermore, we show that rNN enables joint certification of multiple testing examples. Figure 1 illustrates an example of individual certification and joint certification with two testing examples in rNN. When we treat the two testing examples individually, an attacker can poison 3 training examples such that rNN misclassifies each of them. However, when we treat

them jointly, an attacker cannot poison 3 training examples to misclassify both of them. We propose such joint certification to derive a better certified accuracy for rNN. Specifically, we design methods to group testing examples in a testing dataset such that we can perform joint certification for each group of testing examples.

We evaluate our methods on MNIST and CIFAR10 datasets. We use  $\ell_1$  distance metric to calculate nearest neighbors. First, our methods substantially outperform state-of-the-art certifiably robust learning algorithms. For instance, in data poisoning attacks, when an attacker can arbitrarily poison 1,000 training examples on MNIST, the certified accuracy of rNN with  $r = 4$  is 22.9% and 40.8% higher than those of bagging (Jia, Cao, and Gong 2021) and DPA (Levine and Feizi 2021), respectively. Second, our joint certification improves certified accuracy. For instance, our joint certification improves the certified accuracy of rNN by 15.1% when an attacker can arbitrarily poison 1,000 training examples on MNIST for data poisoning attacks. Third, we show that self-supervised learning can improve the certified accuracy of kNN and rNN. For instance, when an attacker can arbitrarily poison 500 training examples on CIFAR10 in data poisoning attacks, the certified accuracy of kNN improves by 43.0% if we use CLIP (Radford et al. 2021), a feature extractor pre-trained via self-supervised learning, to extract features for each training or testing input.

In summary, we make the following contributions:

- We derive the intrinsic certified robustness guarantees of kNN and rNN against data poisoning attacks and backdoor attacks.
- We propose joint certification of multiple testing examples to derive a better certified robustness guarantee for rNN. rNN is the first method that supports joint certification of multiple testing examples.
- We evaluate our methods and compare them with state-of-the-art on MNIST and CIFAR10.

## Problem Setup

**Learning setting:** Assuming we have a training dataset  $D_{tr}$  with  $n$  training examples. We denote by  $\mathcal{M}$  a learning algorithm. Moreover, we denote by  $\mathcal{M}(D_{tr}, \mathbf{x})$  the label predicted for a testing input  $\mathbf{x}$  by a classifier learnt by  $\mathcal{M}$  on the training dataset  $D_{tr}$ . For instance, given a training dataset  $D_{tr}$  and a testing input  $\mathbf{x}$ , kNN finds the  $k$  training examples in  $D_{tr}$  that are the closest to  $\mathbf{x}$  as the nearest neighbors, while rNN finds the training examples in  $D_{tr}$  whose distances to  $\mathbf{x}$  are no larger than  $r$  as the nearest neighbors. The distance between a training input and a testing input can be measured by any distance metric. Then, kNN and rNN use majority vote among the nearest neighbors to predict the label of  $\mathbf{x}$ . Specifically, each nearest neighbor is a voter and votes its label for the testing input  $\mathbf{x}$ ; and the label with the largest number of votes is the final predicted label for  $\mathbf{x}$ .

**Data poisoning attacks:** We consider data poisoning attacks (Rubinstein et al. 2009; Biggio, Nelson, and Laskov 2012; Xiao et al. 2015a; Li et al. 2016; Muñoz-González et al. 2017; Jagielski et al. 2018) that aim to poison (i.e., modify, add, and/or remove) some carefully selected training

examples in  $D_{tr}$  such that the corrupted classifier has a low accuracy for testing inputs (either indiscriminate clean testing inputs or attacker-chosen ones).

**Backdoor attacks:** In backdoor attacks (Gu, Dolan-Gavitt, and Garg 2017; Liu et al. 2017; Chen et al. 2017), an attacker also poisons the training dataset, but the corrupted classifier makes incorrect, attacker-chosen predictions for testing inputs embedded with a certain trigger. For instance, the attacker can embed the trigger to some training inputs in  $D_{tr}$  and relabel them as the attacker-chosen label. The classifier built based on such poisoned training dataset predicts the attacker-chosen label for any testing input embedded with the same trigger. However, the predictions for clean testing inputs without the trigger are unaffected, i.e., the corrupted classifier and the clean classifier are highly likely to predict the same label for a clean testing input.

**Poisoned training dataset:** Both data poisoning attacks and backdoor attacks poison the training dataset to achieve their goals. For simplicity, we use  $D_{tr}^*$  to denote the *poisoned training dataset*. Note that  $D_{tr}^*$  could include duplicate training examples, e.g., when the attacker adds duplicate training examples. Moreover, we define the *poisoning size* of a poisoned training dataset  $D_{tr}^*$  (denoted as  $S(D_{tr}, D_{tr}^*)$ ) as the minimal number of modified/added/removed training examples that can turn  $D_{tr}$  into  $D_{tr}^*$ . Formally,  $S(D_{tr}, D_{tr}^*) = \max\{|D_{tr}^*|, |D_{tr}|\} - |D_{tr}^* \cap D_{tr}|$  is the poisoning size of  $D_{tr}^*$ .

**Certified accuracy:** Given a training dataset  $D_{tr}$  and a learning algorithm  $\mathcal{M}$ , we use *certified accuracy* on a testing dataset  $D_{te} = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$  to measure the algorithm’s performance. Specifically, we denote *certified accuracy at poisoning size  $e$*  as  $CA(e)$  and formally define it as follows:

$$CA(e) = \min_{D_{tr}^*, S(D_{tr}, D_{tr}^*) \leq e} \frac{\sum_{(\mathbf{x}_i, y_i) \in D_{te}} \mathbb{I}(\mathcal{M}(D_{tr}^*, \mathbf{x}_i) = y_i)}{|D_{te}|}, \quad (1)$$

where  $\mathbb{I}$  is the indicator function and  $\mathcal{M}(D_{tr}^*, \mathbf{x}_i)$  is the label predicted for a testing input  $\mathbf{x}_i$  by the classifier learnt by the algorithm  $\mathcal{M}$  on the poisoned training dataset  $D_{tr}^*$ .  $CA(e)$  is the least testing accuracy on  $D_{te}$  that the learning algorithm  $\mathcal{M}$  can achieve no matter how an attacker poisons the training examples when the poisoning size is at most  $e$ . For data poisoning attacks, the testing dataset  $D_{te}$  is a set of clean testing examples. For backdoor attacks,  $D_{te}$  is a set of testing examples embedded with a trigger. Our goal is to derive lower bounds of  $CA(e)$  for learning algorithms kNN and rNN.

## Certified Accuracy of kNN and rNN

We first derive a lower bound of the certified accuracy via *individual certification*, which treats testing examples in  $D_{te}$  individually. Then, we derive a better lower bound of the certified accuracy for rNN via *joint certification*, which treats testing examples jointly.

### Individual Certification

Given a poisoning size at most  $e$ , our idea is to certify whether the predicted label stays unchanged or not for each testing

input individually. If the predicted label of a testing input  $\mathbf{x}$  stays unchanged (i.e.,  $\mathcal{M}(D_{tr}, \mathbf{x}) = \mathcal{M}(D_{tr}^*, \mathbf{x})$ ) and it matches with the testing input’s true label, then kNN or rNN certifiably correctly classifies the testing input when the poisoning size is at most  $e$ . Therefore, we can obtain a lower bound of the certified accuracy at poisoning size  $e$  as the fraction of testing inputs in  $D_{te}$  which kNN or rNN certifiably correctly classifies. Next, we first discuss how to certify whether the predicted label stays unchanged or not for each testing input individually. Then, we show our lower bound of the certified accuracy at poisoning size  $e$ .

**Certifying the predicted label of a testing input:** Our goal is to certify that  $\mathcal{M}(D_{tr}, \mathbf{x}) = \mathcal{M}(D_{tr}^*, \mathbf{x})$  for a testing input  $\mathbf{x}$  when the poisoning size is no larger than a threshold. Given a training dataset  $D_{tr}$  (or a poisoned training dataset  $D_{tr}^*$ ) and a testing input  $\mathbf{x}$ , we use  $\mathcal{N}(D_{tr}, \mathbf{x})$  (or  $\mathcal{N}(D_{tr}^*, \mathbf{x})$ ) to denote the set of nearest neighbors of  $\mathbf{x}$  in  $D_{tr}$  (or  $D_{tr}^*$ ) for kNN or rNN. We note that there may exist ties when determining the nearest neighbors for kNN, i.e., multiple training examples may have the same distance to the testing input. Usually, kNN breaks such ties uniformly at random. However, such random ties breaking method introduces randomness, i.e., the difference of nearest neighbors before and after poisoned training examples (i.e.,  $\mathcal{N}(D_{tr}, \mathbf{x})$  vs.  $\mathcal{N}(D_{tr}^*, \mathbf{x})$ ) depends on the randomness in breaking ties. Such randomness makes it challenging to certify the robustness of the predicted label against poisoned training examples. To address the challenge, we propose to define a deterministic ranking of training examples and break ties via choosing the training examples with larger ranks. Moreover, such ranking between clean training examples does not depend on poisoned ones. For instance, we can use a cryptographic hash function (e.g., SHA-1) that is very unlikely to have collisions to hash each training example based on its input feature vector and label, and then we rank the training examples based on their hash values.

We use  $s_l$  to denote the number of votes in  $\mathcal{N}(D_{tr}, \mathbf{x})$  for label  $l$ , i.e., the number of nearest neighbors in  $\mathcal{N}(D_{tr}, \mathbf{x})$  whose labels are  $l$ . Formally, we have  $s_l = \sum_{(\mathbf{x}_j, y_j) \in \mathcal{N}(D_{tr}, \mathbf{x})} \mathbb{I}(y_j = l)$ , where  $l = 1, 2, \dots, c$  and  $\mathbb{I}$  is an indicator function. We note that  $s_l$  also depends on the testing input  $\mathbf{x}$ . However, we omit the explicit dependency on  $\mathbf{x}$  for simplicity. kNN or rNN essentially predicts the label of the testing input  $\mathbf{x}$  as the label with the largest number of votes, i.e.,  $\mathcal{M}(D_{tr}, \mathbf{x}) = \operatorname{argmax}_{l \in \{1, 2, \dots, c\}} s_l$ . Suppose  $a$  and  $b$  are the labels with the largest and second largest number of votes, i.e.,  $s_a$  and  $s_b$  are the largest and second largest ones among  $\{s_1, s_2, \dots, s_c\}$ , respectively. We note that there may exist ties when comparing the labels based on their votes. We define a deterministic ranking of labels in  $\{1, 2, \dots, c\}$  and take the label with the largest rank when such ties happen. For instance, when labels 2 and 3 have tied largest number of votes, we take label 3 as  $a$ . In the worse-case scenario, each poisoned training example leads to one corrupted voter in kNN or rNN, which changes its vote from label  $a$  to label  $b$ . Therefore, kNN or rNN still predicts label  $a$  for the testing input  $\mathbf{x}$  when the number of poisoned training examples is no more than  $\lceil \frac{s_a - s_b}{2} \rceil - 1$  (without considering the ties breaking). Formally, we have the following theorem:

**Theorem 1.** Assuming we have a training dataset  $D_{tr}$ , a testing input  $\mathbf{x}$ , and a nearest neighbor algorithm  $\mathcal{M}$  (i.e., kNN or rNN).  $a$  and  $b$  respectively are the two labels with the largest and second largest number of votes among the nearest neighbors  $\mathcal{N}(D_{tr}, \mathbf{x})$  of  $\mathbf{x}$  in  $D_{tr}$ . Moreover,  $s_a$  and  $s_b$  are the number of votes for  $a$  and  $b$ , respectively. Then, we have the following:

$$\mathcal{M}(D_{tr}^*, \mathbf{x}) = a, \\ \forall D_{tr}^*, S(D_{tr}, D_{tr}^*) \leq \lceil \frac{s_a - s_b + \mathbb{I}(a > b)}{2} \rceil - 1. \quad (2)$$

*Proof.* See Supplementary Material.  $\square$

**Deriving a lower bound of  $CA(e)$ :** kNN or rNN certifiably correctly classifies a testing input  $\mathbf{x}$  if it correctly predicts its label before attacks and the predicted label stays unchanged after an attacker poisons the training dataset. Therefore, the fraction of testing inputs that kNN or rNN certifiably correctly classifies is a lower bound of  $CA(e)$ . Formally, we have the following theorem:

**Theorem 2 (Individual Certification).** Assuming we have a training dataset  $D_{tr}$ , a testing dataset  $D_{te} = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$ , and a nearest neighbor algorithm  $\mathcal{M}$  (i.e., kNN or rNN).  $a_i$  and  $b_i$  respectively are the two labels with the largest and second largest number of votes among the nearest neighbors  $\mathcal{N}(D_{tr}, \mathbf{x}_i)$  of  $\mathbf{x}_i$  in  $D_{tr}$ . Moreover,  $s_{a_i}$  and  $s_{b_i}$  are the number of votes for  $a_i$  and  $b_i$ , respectively. Then, we have the following lower bound of  $CA(e)$ :

$$CA(e) \geq \frac{\sum_{(\mathbf{x}_i, y_i) \in D_{te}} \mathbb{I}(a_i = y_i) \cdot \mathbb{I}(e \leq e_i^*)}{|D_{te}|}, \quad (3)$$

$$\text{where } e_i^* = \lceil \frac{s_{a_i} - s_{b_i} + \mathbb{I}(a_i > b_i)}{2} \rceil - 1.$$

*Proof.* See Supplementary Material.  $\square$

## Joint Certification

We derive a better lower bound of the certified accuracy via jointly considering multiple testing examples. Our intuition is that, given a group of testing examples and a poisoning size  $e$ , an attacker may not be able to make a learning algorithm misclassify all the testing examples jointly even if it can make the learning algorithm misclassify each of them individually. In particular, rNN enables such joint certification. It is challenging to perform joint certification for kNN because of the complex interactions between the nearest neighbors of different testing examples (see our proof of Theorem 3 for specific reasons). Next, we first derive a lower bound of  $CA(e)$  on a group of testing examples for rNN. Then, we derive a lower bound of  $CA(e)$  on the testing dataset  $D_{te}$  via dividing it into groups. Finally, we discuss different strategies to divide the testing dataset into groups, which may lead to different lower bounds of  $CA(e)$ .

**Deriving a lower bound of  $CA(e)$  for a group of testing examples:** Suppose we have a group of testing examples which have different predicted labels in rNN. Our key intuition is that when an attacker can poison  $e$  training examples,

the attacker can only decrease the total votes for the testing examples' predicted labels by at most  $e$  in rNN, as the testing examples' predicted labels are different. We denote by  $\mathcal{U}$  a group of testing examples with different predicted labels and by  $m$  its size, i.e.,  $m = |\mathcal{U}|$ . The next theorem shows a lower bound of  $CA(e)$  on the testing examples in  $\mathcal{U}$  for rNN.

**Theorem 3.** Assuming we have a training dataset  $D_{tr}$ , the learning algorithm rNN, and a group of  $m$  testing examples  $\mathcal{U} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$  with different predicted labels.  $a_i$  and  $b_i$  respectively are the two labels with the largest and second largest number of votes among the nearest neighbors  $\mathcal{N}(D_{tr}, \mathbf{x}_i)$  of  $\mathbf{x}_i$  in  $D_{tr}$ . Moreover,  $s_{a_i}$  and  $s_{b_i}$  are the number of votes for  $a_i$  and  $b_i$ , respectively. Without loss of generality, we assume the following:

$$(s_{a_1} - s_{b_1}) \cdot \mathbb{I}(a_1 = y_1) \geq (s_{a_2} - s_{b_2}) \cdot \mathbb{I}(a_2 = y_2) \geq \dots \geq (s_{a_m} - s_{b_m}) \cdot \mathbb{I}(a_m = y_m). \quad (4)$$

Then, the certified accuracy at poisoning size  $e$  of rNN for  $\mathcal{U}$  has a lower bound  $CA(e) \geq \frac{w-1}{|\mathcal{U}|}$ , where  $w$  is the solution to the following optimization problem:

$$w = \underset{w', w' \geq 1}{\operatorname{argmin}} w' \\ \text{s.t. } \sum_{i=w'}^m \max(s_{a_i} - s_{b_i} - e + \mathbb{I}(a_i > b_i), 0) \\ \cdot \mathbb{I}(a_i = y_i) \leq e. \quad (5)$$

*Proof.* When an attacker can poison at most  $e$  training examples, the attacker can add at most  $e$  new nearest neighbors and remove  $e$  existing ones in  $\mathcal{N}(D_{tr}, \mathbf{x}_i)$  (equivalent to modifying  $e$  training examples) in the worst-case scenario. We denote by  $s_{a_i}^*$  and  $s_{b_i}^*$  respectively the number of votes for labels  $a_i$  and  $b_i$  among the nearest neighbors  $\mathcal{N}(D_{tr}^*, \mathbf{x}_i)$ . First, we have  $s_{b_i}^* \leq s_{b_i} + e$  for  $\forall i \in \{1, 2, \dots, m\}$  since at most  $e$  new nearest neighbors are added. Second, we have  $s_{a_i}^* \geq s_{a_i} - e_i$  in rNN, where  $e_i$  is the number of removed nearest neighbors in  $\mathcal{N}(D_{tr}, \mathbf{x}_i)$  whose true labels are  $a_i$ . Note that kNN does not support joint certification because  $s_{a_i}^* \geq s_{a_i} - e_i$  does not hold for kNN.

Next, we derive the minimal value of  $e_i$  such that rNN misclassifies  $\mathbf{x}_i$ . In particular, we consider two cases. If  $a_i \neq y_i$ , i.e.,  $\mathbf{x}_i$  is misclassified by rNN without attack, then we have  $e_i = 0$ . If  $a_i = y_i$ ,  $\mathbf{x}_i$  is misclassified by rNN when  $s_{a_i}^* \leq s_{b_i}^*$  if  $a_i < b_i$  and  $s_{a_i}^* < s_{b_i}^*$  if  $a_i > b_i$  after attack, which means  $e_i \geq s_{a_i} - s_{b_i} - e + \mathbb{I}(a_i > b_i)$ . Since  $e_i \geq 0$ , we have  $e_i \geq \max(s_{a_i} - s_{b_i} - e + \mathbb{I}(a_i > b_i), 0)$ . Combining the two cases, we have the following lower bound for  $e_i$  that makes rNN misclassify  $\mathbf{x}_i$ :  $e_i \geq \max(s_{a_i} - s_{b_i} - e + \mathbb{I}(a_i > b_i), 0) \cdot \mathbb{I}(a_i = y_i)$ . Moreover, since the attacker can remove at most  $e$  training examples and the group of testing examples have different predicted labels, i.e.,  $a_i \neq a_j \forall i, j \in \{1, 2, \dots, m\}$  and  $i \neq j$ , we have  $\sum_{i=1}^m e_i \leq e$ . We note that the lower bound of  $e_i$  is non-increasing as  $i$  increases based on Equation (4). Therefore, in the worst-case scenario, the attacker can make rNN misclassify the last  $m - w + 1$  testing inputs whose corresponding  $e_i$  sum to be at most  $e$ . Formally,  $w$  is the solution to the optimization problem in Equation (5). Therefore, the certified accuracy at poisoning size  $e$  is at least  $\frac{w-1}{|\mathcal{U}|}$ .  $\square$

**Deriving a lower bound of  $CA(e)$  for a testing dataset:** Based on Theorem 3, we can derive a lower bound of  $CA(e)$  for a testing dataset via dividing it into disjoint groups, each of which includes testing examples with different predicted labels in rNN. Formally, we have the following theorem:

**Theorem 4** (Joint Certification). *Given a testing dataset  $D_{te}$ , we divide it into  $\lambda$  disjoint groups, i.e.,  $\mathcal{U}_1, \mathcal{U}_2, \dots, \mathcal{U}_\lambda$ , where the testing examples in each group have different predicted labels in rNN. Then, we have the following lower bound of  $CA(e)$ :*

$$CA(e) \geq \frac{\sum_{j=1}^{\lambda} \mu_j \cdot |\mathcal{U}_j|}{\sum_{j=1}^{\lambda} |\mathcal{U}_j|}, \quad (6)$$

where  $\mu_j$  is the lower bound of the certified accuracy at poisoning size  $e$  on group  $\mathcal{U}_j$ , which we can obtain by invoking Theorem 3.

*Proof.* See Supplementary Material.  $\square$

**Strategies of grouping testing examples:** Our Theorem 4 is applicable to any way of dividing the testing examples in  $D_{te}$  to disjoint groups once the testing examples in each group have different predicted labels in rNN. Therefore, a natural question is how to group the testing examples in  $D_{te}$  to maximize our lower bound of certified accuracy. For instance, a naive method is to randomly divide the testing examples into disjoint groups, each of which includes at most  $c$  (the number of classes) testing examples with different predicted labels. We call such method *Random Division (RD)*. However, RD achieves suboptimal performance because it does not consider the certified robustness of each individual testing example. In particular, some testing examples can or cannot be certifiably correctly classified no matter which groups they belong to. However, if we group them with other testing examples, the certified accuracy may be degraded because each group can have at most  $c$  testing examples. For instance, if a testing example cannot be certifiably correctly classified no matter which group it belongs to, then adding it to a group would exclude another testing example from the group, which may degrade the certified accuracy for the group.

Therefore, we propose to isolate these testing examples and divide the remaining testing examples into disjoint groups. We call such method *Isolation and Division (ISLAND)*. Specifically, we first divide the testing dataset  $D_{te}$  into three disjoint parts which we denote as  $D_{te}^0$ ,  $D_{te}^1$ , and  $D_{te}^2$ .  $D_{te}^0$  contains the testing examples that cannot be certifiably correctly classified at poisoning size  $e$  no matter which group they belong to. Based on our proof of Theorem 3, a testing example  $(\mathbf{x}_i, y_i)$  that satisfies  $(s_{a_i} - s_{b_i} - e + \mathbb{I}(a_i > b_i)) \cdot \mathbb{I}(a_i = y_i) \leq 0$  cannot be certifiably correctly classified at poisoning size  $e$  no matter which group it belongs to. Therefore,  $D_{te}^0$  includes such testing examples. Moreover, based on Theorem 1, a testing example  $(\mathbf{x}_i, y_i)$  that satisfies  $e \leq \lceil \frac{s_{a_i} - s_{b_i} + \mathbb{I}(a_i > b_i)}{2} \rceil - 1$  can be certifiably correctly classified at poisoning size  $e$ . Therefore,  $D_{te}^1$  includes such testing examples. Each testing example in  $D_{te}^0$  or  $D_{te}^1$  forms a group

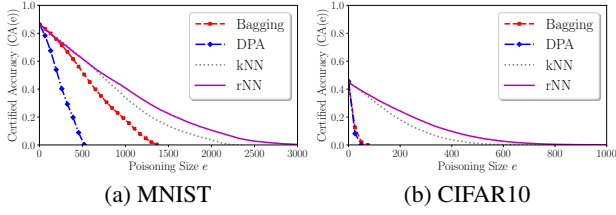
by itself.  $D_{te}^2$  includes the remaining testing examples, which we further divide into groups. Our method of dividing  $D_{te}^2$  into groups is inspired by the proof of Theorem 3. In particular, we form a group of testing examples as follows: for each label  $l \in \{1, 2, \dots, c\}$ , we find the testing example that has the largest value of  $(s_{a_i} - s_{b_i} - e + \mathbb{I}(a_i > b_i)) \cdot \mathbb{I}(a_i = l)$  and we skip the label if there is no remaining testing example whose predicted label is  $l$ . We apply the procedure to recursively group the testing examples in  $D_{te}^2$  until no testing examples are left.

## Evaluation

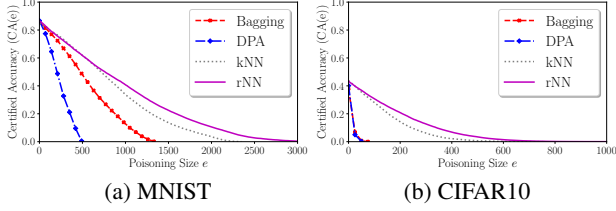
**Datasets:** We evaluate our methods on MNIST and CIFAR10. We use the popular histogram of oriented gradients (HOG) (Dalal and Triggs 2005) method (we adopt public implementation (hog 2021)) to extract features for each example, which we found improves certified accuracy. Note that previous work (Jia, Cao, and Gong 2021) used a pre-trained model to extract features via transfer learning. However, the pre-trained model may also be poisoned and thus we don't use it. For simplicity, we rank the training examples in a dataset using their indices and use them to break ties in determining nearest neighbors for kNN. Moreover, we rank the labels as  $\{1, 2, \dots, 10\}$  to break ties for labels. Following previous work (Wang et al. 2019a), we adopt a white square located at the bottom right corner of an image as the trigger in backdoor attacks for both MNIST and CIFAR10. The sizes of the triggers are  $5 \times 5$  and  $10 \times 10$  for MNIST and CIFAR10, considering different image sizes in those two datasets.

**Parameter settings:** While any distance metric is applicable, we use  $\ell_1$  in our experiments for both kNN and rNN. Unless otherwise mentioned, we adopt the following settings:  $k = 5, 000$  for both MNIST and CIFAR10 in kNN; and  $r = 4$  for MNIST and  $r = 20$  for CIFAR10 in rNN, considering the different feature dimensions of MNIST and CIFAR10. By default, we use the ISLAND grouping method in the joint certification for rNN.

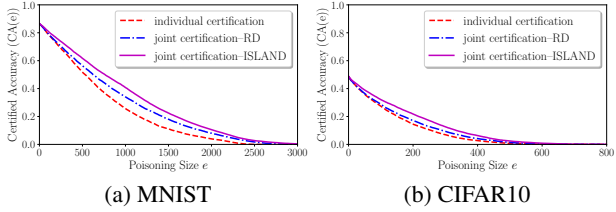
**Comparing with bagging (Jia, Cao, and Gong 2021) and DPA (Levine and Feizi 2021):** Figure 2 and 3 show the comparison results of bagging, DPA, kNN, and rNN for data poisoning attacks and backdoor attacks, respectively. Bagging learns  $N$  base classifiers, each of which is learnt on a random subsample with  $\xi$  training examples of the training dataset. Moreover, bagging's certified accuracy is correct with a confidence level  $1 - \alpha$ . DPA divides a training dataset into  $\zeta$  disjoint partitions and learns a base classifier on each of them. Then, DPA takes a majority vote among the base classifiers to predict the label of a testing example. All the compared methods have tradeoffs between accuracy under no attacks (i.e.,  $CA(0)$ ) and robustness against attacks. Therefore, we set their parameters such that they have similar accuracy under no attacks (i.e., similar  $CA(0)$ ). In particular, we use the default  $k$  for kNN, and we adjust  $r$  for rNN,  $\xi$  for bagging, and  $\zeta$  for DPA. The searched parameters are as follows:  $r = 4$ ,  $\xi = 20$ , and  $\zeta = 2, 500$  for MNIST; and  $r = 21$ ,  $\xi = 300$ , and  $\zeta = 500$  for CIFAR10. We find that the searched parameters are the same for data poisoning attacks and backdoor attacks. Note that we set  $N = 1, 000$  and  $\alpha = 0.001$  for bagging following Jia, Cao, and Gong (2021).



**Figure 2: Comparing kNN and rNN with state-of-the-art methods against data poisoning attacks.**



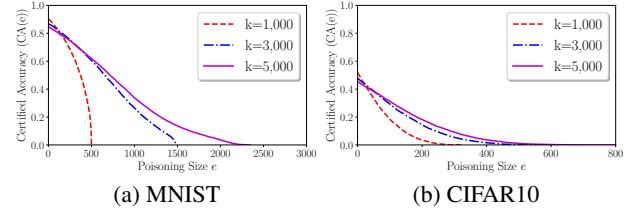
**Figure 3: Comparing kNN and rNN with state-of-the-art methods against backdoor attacks.**



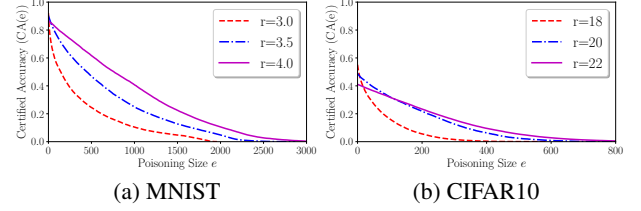
**Figure 4: Comparing individual certification with joint certification for rNN against data poisoning attacks.**

We have the following observations. First, both kNN and rNN outperform bagging and DPA. The superior performance of kNN and rNN stems from two reasons: 1) each poisoned training example corrupts multiple voters for bagging and DPA, while it only corrupts one voter for kNN and rNN, which means that, given the same gap between the largest and second largest number of votes, kNN and rNN can tolerate more poisoned training examples; and 2) rNN enables joint certification that improves the certified accuracy. Second, rNN achieves better certified accuracy than kNN when the poisoning size is large. The reason is that rNN supports joint certification. Third, kNN (or rNN) achieves similar certified accuracy against data poisoning attacks and backdoor attacks for a given poisoning size. This is because, in backdoor attacks, adding a trigger to a testing image does not affect its label predicted by a clean classifier.

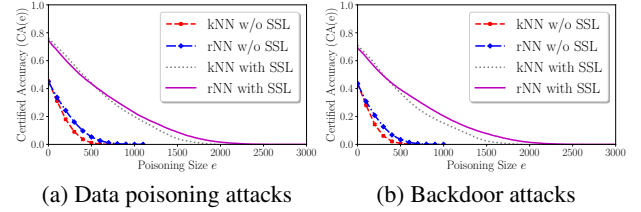
**Comparing individual certification with joint certification:** Figure 4 and 8 (in Supplementary Material) compare individual certification and joint certification (with the RD and ISLAND grouping methods) for rNN against data poisoning attacks and backdoor attacks. Our empirical results validate that joint certification improves the certified accuracy upon individual certification. Moreover, our ISLAND grouping method outperforms the RD method.



**Figure 5: Impact of  $k$  on the certified accuracy of kNN against data poisoning attacks.**



**Figure 6: Impact of  $r$  on the certified accuracy of rNN against data poisoning attacks.**



**Figure 7: Self-supervised learning improves the certified accuracy of kNN and rNN, where CIFAR10 is used.**

**Impact of  $k$  and  $r$ :** Figure 5, 6, 9 (in Supplementary Material), 10 (in Supplementary Material) show the impact of  $k$  and  $r$  on the certified accuracy of kNN and rNN against data poisoning attacks and backdoor attacks, respectively. As the results show,  $k$  and  $r$  achieve tradeoffs between accuracy under no attacks (i.e.,  $CA(0)$ ) and robustness. Specifically, when  $k$  or  $r$  is smaller, the accuracy under no attacks, i.e.,  $CA(0)$ , is larger, but the certified accuracy decreases more quickly as the poisoning size  $e$  increases.

**Self-supervised learning improves the certified accuracy:** *Self-supervised learning (SSL)* (Hadsell, Chopra, and LeCun 2006; He et al. 2020; Chen et al. 2020) aims to learn a feature extractor using a large amount of unlabeled data, such that the feature extractor can be used to extract features for a variety of downstream learning tasks. We adopt the pre-trained feature extractor called CLIP (Radford et al. 2021) to extract features. CLIP was pre-trained using 400 million (image, text) pairs and we use its public implementation (cli 2021). Note that we assume the pre-trained feature extractor is not poisoned. For each training or testing input in CIFAR10, we use CLIP to extract its features. Then, given the features, we use kNN or rNN to classify testing inputs. We use the default  $k$  (i.e.,  $k = 5,000$ ) for both kNN without SSL and kNN with SSL. We adjust  $r$  such that rNN without SSL (or rNN



with SSL) has similar certified accuracy under no attacks with kNN without SSL (or kNN with SSL). Figure 7 shows the comparison results. Our results show that self-supervised learning can significantly improve the certified accuracy of kNN and rNN.

## Related Work

**Data poisoning attacks and backdoor attacks:** Data poisoning attacks have been proposed against various learning algorithms such as Bayes classifier (Nelson et al. 2008), SVM (Biggio, Nelson, and Laskov 2012), neural networks (Muñoz-González et al. 2017; Shafahi et al. 2018; Suciu et al. 2018; Demontis et al. 2019; Zhu et al. 2019; Huang et al. 2020), recommender systems (Li et al. 2016; Yang, Gong, and Cai 2017; Fang et al. 2018; Fang, Gong, and Liu 2020; Huang et al. 2021), federated learning (Bhagoji et al. 2019; Fang et al. 2020; Bagdasaryan et al. 2020), and others (Rubinstein et al. 2009; Vuurens, de Vries, and Eickhoff 2011). Backdoor attacks (Gu, Dolan-Gavitt, and Garg 2017; Chen et al. 2017; Liu et al. 2017; Jia, Liu, and Gong 2022) corrupt both the training and testing phases of a machine learning system such that the corrupted classifier predicts an attacker-chosen label for any input embedded with a trigger. **Defenses against data poisoning attacks and backdoor attacks:** To mitigate data poisoning attacks and/or backdoor attacks, many empirical defenses (Cretu et al. 2008; Rubinstein et al. 2009; Barreno et al. 2010; Biggio et al. 2011; Feng et al. 2014; Jagielski et al. 2018; Tran, Li, and Madry 2018; Diakonikolas et al. 2019; Liu, Dolan-Gavitt, and Garg 2018; Wang et al. 2019a) have been proposed. Steinhardt, Koh, and Liang (2017) derived an upper bound of the loss function for data poisoning attacks when the model is learnt using examples in a feasible set. Diakonikolas et al. (2019) proposed a robust meta-algorithm which iteratively removes outliers such that the model parameters learnt on the poisoned dataset are close to those learnt on the clean dataset under certain assumptions. However, these defenses cannot guarantee that the predicted label for a testing input is certifiably unaffected and cannot provide (rigorous) certified accuracies.

Recently, several certified defenses (Ma, Zhu, and Hsu 2019; Wang et al. 2020; Rosenfeld et al. 2020; Weber et al. 2020; Levine and Feizi 2021; Jia, Cao, and Gong 2021) were proposed to defend against data poisoning attacks and/or backdoor attacks. These defenses provide certified accuracies for a testing dataset either probabilistically (Ma, Zhu, and Hsu 2019; Wang et al. 2020; Weber et al. 2020; Jia, Cao, and Gong 2021) or deterministically (Rosenfeld et al. 2020; Levine and Feizi 2021). All these defenses except Ma, Zhu, and Hsu (2019) leverage majority vote to predict the label of a testing example. In particular, a voter is a base classifier learnt on a perturbed version of the training dataset in randomized smoothing based defenses (Rosenfeld et al. 2020; Wang et al. 2020; Weber et al. 2020), while a voter is a base classifier learnt on a subset of the training dataset in bagging (Jia, Cao, and Gong 2021) and DPA (Levine and Feizi 2021). Ma, Zhu, and Hsu (2019) showed that a differentially private learning algorithm achieves certified accuracy against data poisoning attacks. They also train multiple differentially private classifiers, but they are not used to predict the label

of a testing example via majority vote. Instead, their average accuracy is used to estimate the certified accuracy.

kNN and rNN have intrinsic majority vote mechanisms and we show that they provide deterministic certified accuracies against data poisoning attacks and backdoor attacks. Moreover, rNN enables joint certification. We note that DPA (Levine and Feizi 2021) proposed to use a hash function to assign training examples into partitions, which is different from our use of hash function. In particular, we use a hash function to rank training examples. Moreover, both DPA and our work rank the labels to break ties.

**Nearest neighbors and robustness:** A line of works (Wilson 1972; Guyon et al. 1996; Peri et al. 2019; Bahri, Jiang, and Gupta 2020) leveraged nearest neighbors to clean a training dataset. For instance, Wilson (1972) proposed to remove a training example whose label is not the same as the majority vote among the labels of its 3 nearest neighbors. Peri et al. (2019) proposed to remove a training example whose label is not the mode amongst labels of its  $k$  nearest neighbors in the feature space. Bahri, Jiang, and Gupta (2020) combined kNN with an intermediate layer of a preliminary deep neural network model to filter suspiciously-labeled training examples. Another line of works (Gao, Niu, and Zhou 2018; Reeve and Kabán 2019) studied the resistance of nearest neighbors to random noisy labels. For instance, Gao, Niu, and Zhou (2018) analyzed the resistance of kNN to asymmetric label noise and introduced a Robust kNN to deal with noisy labels. Reeve and Kabán (2019) further analyzed the Robust kNN proposed by Gao, Niu, and Zhou (2018) in the setting with unknown asymmetric label noise.

kNN and its variants have also been used to defend against adversarial examples (Wang, Jha, and Chaudhuri 2018; Sitawarin and Wagner 2019a; Papernot and McDaniel 2018; Sitawarin and Wagner 2019b; Dubey et al. 2019; Yang et al. 2020; Cohen, Sapiro, and Giryes 2020). For instance, Wang, Jha, and Chaudhuri (2018) analyzed the robustness of nearest neighbors to adversarial examples and proposed a more robust 1-nearest neighbor. Several works (Amsaleg et al. 2017; Wang, Jha, and Chaudhuri 2018; Wang et al. 2019b; Yang et al. 2020) proposed adversarial examples to nearest neighbors, e.g., Wang et al. (2019b) proposed adversarial examples against 1-nearest neighbor. These works are orthogonal to ours as we focus on analyzing the certified robustness of kNN and rNN against data poisoning and backdoor attacks.

## Conclusion and Future Work

In this work, we derive the certified robustness of nearest neighbor algorithms, including kNN and rNN, against data poisoning attacks and backdoor attacks. Moreover, we derive a better lower bound of certified accuracy for rNN via jointly certifying multiple testing examples. Our evaluation results show that 1) both kNN and rNN outperform state-of-the-art certified defenses against data poisoning attacks and backdoor attacks, and 2) joint certification outperforms individual certification. Interesting future work includes 1) extending joint certification to other learning algorithms, 2) improving joint certification via new grouping methods, and 3) improving certified accuracy of kNN and rNN via new distance metrics.

## Acknowledgments

We thank the anonymous reviewers for insightful reviews. This work was supported by the National Science Foundation under Grants No. 1937786 and 2112562, as well as the Army Research Office under Grant No. W911NF2110182.

## References

2021. CLIP. <https://github.com/openai/CLIP>. Accessed: 2021-08.
2021. HOG. <https://scikit-image.org/docs/dev/api/skimage.feature.html#skimage.feature.hog>. Accessed: 2021-08.
- Amsaleg, L.; Bailey, J.; Barbe, D.; Erfani, S.; Houle, M. E.; Nguyen, V.; and Radovanović, M. 2017. The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *WIFS*.
- Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *AISTATS*.
- Bahri, D.; Jiang, H.; and Gupta, M. 2020. Deep k-NN for Noisy Labels. In *ICML*.
- Barreno, M.; Nelson, B.; Joseph, A. D.; and Tygar, J. D. 2010. The security of machine learning. *Machine Learning*.
- Barreno, M.; Nelson, B.; Sears, R.; Joseph, A. D.; and Tygar, J. D. 2006. Can machine learning be secure? In *AsiaCCS*.
- Bhagoji, A. N.; Chakraborty, S.; Mittal, P.; and Calo, S. 2019. Analyzing federated learning through an adversarial lens. In *ICML*.
- 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*.
- Biggio, B.; Fumera, G.; and Roli, F. 2013. Security evaluation of pattern classifiers under attack. *IEEE Transactions on Knowledge and Data Engineering*, 26(4): 984–996.
- Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning attacks against support vector machines. In *ICML*.
- Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.
- Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv*.
- Cohen, G.; Sapiro, G.; and Giryes, R. 2020. Detecting adversarial samples using influence functions and nearest neighbors. In *CVPR*.
- Cover, T.; and Hart, P. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1): 21–27.
- 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 S & P*.
- Dalal, N.; and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR*.
- Demontis, A.; Melis, M.; Pintor, M.; Jagielski, M.; Biggio, B.; Oprea, A.; Nita-Rotaru, C.; and Roli, F. 2019. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *USENIX Security Symposium*.
- Diakonikolas, I.; Kamath, G.; Kane, D.; Li, J.; Steinhardt, J.; and Stewart, A. 2019. Sever: A robust meta-algorithm for stochastic optimization. In *ICML*.
- Dubey, A.; Maaten, L. v. d.; Yalniz, Z.; Li, Y.; and Mahajan, D. 2019. Defense against adversarial images using web-scale nearest-neighbor search. In *CVPR*.
- Fang, M.; Cao, X.; Jia, J.; and Gong, N. 2020. Local model poisoning attacks to Byzantine-robust federated learning. In *USENIX Security Symposium*.
- Fang, M.; Gong, N. Z.; and Liu, J. 2020. Influence function based data poisoning attacks to top-n recommender systems. In *WWW*.
- Fang, M.; Yang, G.; Gong, N. Z.; and Liu, J. 2018. Poisoning attacks to graph-based recommender systems. In *ACSAC*.
- Feng, J.; Xu, H.; Mannor, S.; and Yan, S. 2014. Robust logistic regression and classification. In *NeurIPS*.
- Fix, E.; and Hodges, J. 1951. Discriminatory Analysis: Non-parametric Discrimination, Consistency Properties. *Report No. 4, USAF School of Aviation Medicine, Randolph Field, Texas, Feb*.
- Gao, W.; Niu, X.-Y.; and Zhou, Z.-H. 2018. On the consistency of exact and approximate nearest neighbor with noisy data. *Arxiv*.
- Gu, T.; Dolan-Gavitt, B.; and Garg, S. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv*.
- Guyon, I.; Matic, N.; Vapnik, V.; et al. 1996. Discovering Informative Patterns and Data Cleaning. In *KDD*.
- Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*.
- He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.
- Huang, H.; Mu, J.; Gong, N. Z.; Li, Q.; Liu, B.; and Xu, M. 2021. Data poisoning attacks to deep learning based recommender systems. In *NDSS*.
- Huang, W. R.; Geiping, J.; Fowl, L.; Taylor, G.; and Goldstein, T. 2020. MetaPoison: Practical General-purpose Clean-label Data Poisoning. In *NeurIPS*.
- 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 *IEEE S & P*.
- Jia, J.; Cao, X.; and Gong, N. Z. 2021. Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks. In *AAAI*.
- Jia, J.; Liu, Y.; and Gong, N. Z. 2022. BadEncoder: Backdoor Attacks to Pre-trained Encoders in Self-Supervised Learning. In *IEEE S & P*.
- Levine, A.; and Feizi, S. 2021. Deep Partition Aggregation: Provable Defense against General Poisoning Attacks. In *ICLR*.
- Li, B.; Wang, Y.; Singh, A.; and Vorobeychik, Y. 2016. Data poisoning attacks on factorization-based collaborative filtering. In *NeurIPS*.



- Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*.
- Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2017. Trojaning attack on neural networks. In *NDSS*.
- Ma, Y.; Zhu, X.; and Hsu, J. 2019. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *IJCAI*.
- 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*.
- 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.
- Papernot, N.; and McDaniel, P. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv*.
- Peri, N.; Gupta, N.; Ronny Huang, W.; Fowl, L.; Zhu, C.; Feizi, S.; Goldstein, T.; and Dickerson, J. P. 2019. Deep k-nn defense against clean-label data poisoning attacks. *arXiv*.
- Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*.
- Reeve, H. W.; and Kabán, A. 2019. Fast rates for a knn classifier robust to unknown asymmetric label noise. *arXiv*.
- Rosenfeld, E.; Winston, E.; Ravikumar, P.; and Kolter, J. Z. 2020. Certified Robustness to Label-Flipping Attacks via Randomized Smoothing. In *ICML*.
- 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 *SIGCOMM*.
- Shafahi, A.; Huang, W. R.; Najibi, M.; Suci, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*.
- Sitawarin, C.; and Wagner, D. 2019a. Defending against adversarial examples with k-nearest neighbor. *arXiv*.
- Sitawarin, C.; and Wagner, D. 2019b. On the robustness of deep k-nearest neighbors. In *IEEE Security and Privacy Workshops*.
- Steinhardt, J.; Koh, P. W. W.; and Liang, P. S. 2017. Certified defenses for data poisoning attacks. In *NeurIPS*.
- Suci, O.; Marginean, R.; Kaya, Y.; Daume III, H.; and Dumitras, T. 2018. When does machine learning FAIL? generalized transferability for evasion and poisoning attacks. In *USENIX Security Symposium*.
- Tran, B.; Li, J.; and Madry, A. 2018. Spectral signatures in backdoor attacks. In *NeurIPS*.
- Vuurens, J.; de Vries, A. P.; and Eickhoff, C. 2011. How much spam can you take? an analysis of crowdsourcing results to increase accuracy. In *Proc. ACM SIGIR Workshop on Crowdsourcing for Information Retrieval*.
- 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.; Yao, Y.; Shan, S.; Li, H.; Viswanath, B.; Zheng, H.; and Zhao, B. Y. 2019a. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE S & P*.
- Wang, L.; Liu, X.; Yi, J.; Zhou, Z.-H.; and Hsieh, C.-J. 2019b. Evaluating the Robustness of Nearest Neighbor Classifiers: A Primal-Dual Perspective. *arXiv*.
- Wang, Y.; Jha, S.; and Chaudhuri, K. 2018. Analyzing the robustness of nearest neighbors to adversarial examples. In *ICML*.
- Weber, M.; Xu, X.; Karlas, B.; Zhang, C.; and Li, B. 2020. RAB: Provable Robustness Against Backdoor Attacks. *arXiv*.
- Wilson, D. L. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3): 408–421.
- Xiao, H.; Biggio, B.; Brown, G.; Fumera, G.; Eckert, C.; and Roli, F. 2015a. Is feature selection secure against training data poisoning? In *ICML*.
- Xiao, H.; Biggio, B.; Nelson, B.; Xiao, H.; Eckert, C.; and Roli, F. 2015b. Support vector machines under adversarial label contamination. *Neurocomputing*, 160: 53–62.
- Yang, G.; Gong, N. Z.; and Cai, Y. 2017. Fake Co-visitation Injection Attacks to Recommender Systems. In *NDSS*.
- Yang, Y.-Y.; Rashtchian, C.; Wang, Y.; and Chaudhuri, K. 2020. Robustness for non-parametric classification: A generic attack and defense. In *AISTATS*.
- 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 *ICML*.