IPCert: Provably Robust Intellectual Property Protection for Machine Learning

Zhengyuan Jiang, Minghong Fang, Neil Zhenqiang Gong
Duke University
{zhengyuan.jiang,minghong.fang,neil.gong}@duke.edu

Abstract

Watermarking and fingerprinting are two popular methods to protect intellectual property (IP) of a model. In particular, a model owner can use them to detect whether a given model is a stolen version of its model. Robustness against perturbation added to a model is a key desired property for IP protection methods. In this work, we first show that existing IP protection methods are not robust against model perturbations in the worst-case scenarios as previously thought. Second, we propose a randomized smoothing based framework that can turn a watermarking/fingerprinting method to be provably robust against model perturbations. However, a straightforward application of randomized smoothing achieves suboptimal provable robustness. To address the challenge, we propose optimization strategies to enhance provable robustness. We evaluate our framework on multiple datasets to show its provable robustness.

1. Introduction

A machine learning model represents intellectual property (IP) of a model owner since training a model often requires a large amount of human, data, and computation resources. However, an attacker can steal a model provider's model (called *target model*) through different ways. For instance, via exploiting computer vulnerabilities, an attacker can compromise the computer that stores the target model and remotely steals it [31]. When the target model is deployed as a client-side application (e.g., a mobile app), an attacker can have access to the application and reverse engineer it to steal the target model [23]. When the target model is deployed as a cloud service, an attacker can steal it via repeatedly querying it [24, 26]. After stealing a target model, an attacker can further *post-process* (e.g., fine-tune, distill) it.

Therefore, various methods have been proposed to protect IP of a model. Given a model (called *suspect model*), a model owner can use these methods to detect whether it is a stolen version of its target model. Existing IP protection methods can be categorized into two groups: *watermarking* [3, 31, 25, 8] and *fingerprinting* [4, 29, 17, 20]. In these methods, a model owner picks some data points (called *IP data points*) and labels for them (called *IP labels*) based on its target model. Given a suspect model, the model owner uses it to predict the label of each IP data point and calculates a *matching rate (MR)* as the fraction of the IP data points, whose labels predicted by the suspect model match with the corresponding IP labels. The model owner detects the suspect model as a stolen version of its target model if MR is larger than a *threshold*. In watermarking, the model owner uses the IP data points and labels to augment the training data when training its target model, while in fingerprinting, the model owner extracts IP data points and labels near the classification boundary of the target model after it was trained using a standard training algorithm.

After an attacker steals a target model, the attacker can post-process it to evade detection of an IP protection method. In particular, when a post-processed version of the target model has a MR that is smaller than the detection threshold, the model owner would not detect it as a stolen version of its target model. Therefore, robustness against post-processing is a key desired property of an IP protection method, i.e., an IP protection method should still detect a post-processed version as a stolen one.

Our work: In this work, we perform a systematic study on the robustness of IP protection methods. First, we show that existing IP protection methods are not robust against post-processing in the worst-case scenarios as previously thought. Second, we propose a framework to build IP protection methods that are provably robust against post-processing even in the worst-case scenarios. Third, we propose optimization strategies to further enhance the provable robustness of our framework. Fourth, we evaluate our framework systematically on multiple datasets.

Measuring robustness of existing IP protection methods. Existing studies [2, 4] claimed that watermarking and fingerprinting are robust against post-processing. However, their claims are based on popular post-processing techniques such as fine-tuning and distillation. We find that such claim is not valid, especially in the worst-case scenarios where the IP protection method and its parameters are known to an

attacker. Specifically, we propose a post-processing method that adds a small carefully crafted perturbation to a target model, such that 1) MR of the perturbed target model is small, so existing IP protection methods cannot detect the perturbed target model as a stolen version, and 2) the perturbed target model has similar accuracy with the target model. Specifically, we formulate finding the perturbation as an optimization problem and leverage projected gradient descent to solve it. Our results on multiple datasets show that, compared to fine-tuning and distillation, existing IP protection methods are much less robust against our post-processing.

A majority vote based framework. After showing that existing IP protection methods are not robust against deliberated post-processing, we then propose a framework called *IPCert* to build IP protection methods that are provably robust against any post-processing even in the worst-case scenarios. Our IPCert is based on *majority vote randomized smoothing* [7] and can turn an existing IP protection method to be provably robust. Suppose a model owner has a target model and protects it using an existing IP protection method, which has a set of IP data points and labels. Given a suspect model, the model owner constructs multiple noisy suspect models via adding random Gaussian noises to it; for each IP data point, the model owner uses each noisy suspect model to predict its label and takes a majority vote among the predicted labels as the final predicted label; and MR is defined as the fraction of IP data points whose majority-vote predicted labels match with the corresponding IP labels. IPCert detects the suspect model as a stolen version if the MR is large enough.

Theoretically, we show that MR of IPCert for a suspect model is larger than a *threshold*, called *certified matching rate (CMR)*, when the ℓ_2 distance between the suspect model and the target model is bounded by R (called *perturbation bound*). In other words, when the target model is perturbed by post-processing, MR for the perturbed target model (i.e., suspect model) is at least CMR no matter what perturbation is added once its ℓ_2 norm is bounded by R.

Optimizing CMR. However, a straightforward application of majority vote randomized smoothing achieves suboptimal CMR as we will show in experiments. To address the challenge, we propose optimization strategies to further enhance CMR of IPCert. Specifically, for watermarking, we propose that the model owner adds Gaussian noise to the model parameters when training the target model; and for fingerprinting, we propose a method that considers robustness when selecting IP data points. Our strategies make the majority-vote label of an IP data point more likely to match with its IP label, leading to a larger CMR.

Extensive evaluation. We evaluate IPCert on multiple datasets. Moreover, we compare IPCert with *median smoothing (MS)* based provably robust IP protection framework [3], which is the only existing framework to build provably robust IP protection methods. First, our results show that, for watermarking, IPCert achieves much better CMR than MS, while for fingerprinting, IPCert achieves comparable CMR with MS. Second, for popular post-processing such as fine-tuning and distillation, IPCert achieves better MR than MS. Third, our results show that our optimization strategies substantially improve CMR of IPCert. Fourth, for suspect models that are non-stolen versions of a target model, both IPCert and MS achieve very small MR, which indicates that they can correctly distinguish between stolen and non-stolen models.

2. Related Work

Watermarking: Watermarking has been widely used to protect the copyrights of multimedia [28]. Many works [31, 27, 18, 14, 9, 3, 25, 21, 22, 16, 5, 11, 30] extended watermarking to protect IP of machine learning models in the past several years. Existing watermarking methods can be *white-box* [25] or *black-box* [3, 31, 2]. In white-box watermarking, a model owner embeds a watermark into the parameters of its target model when training it; and given a white-box access to a suspect model, the model owner detects it as a stolen version of the target model if the same watermark can be extracted from its parameters. In black-box watermarking, the model owner picks a set of IP data points and labels, and uses them to augment the training dataset when training the target model. Given a suspect model, the model owner just queries it to obtain its predicted labels for the IP data points and detects it as a stolen version if the MR between the predicted labels and the IP labels is large enough.

Black-box watermarking is more general than white-box watermarking since the former only needs a black-box access to a suspect model. Therefore, in this work, we focus on black-box watermarking and simply use the term watermarking unless otherwise mentioned.

Fingerprinting: The key idea of fingerprinting [4, 29, 17, 20] is that each machine learning model has a unique classification boundary. Therefore, fingerprinting extracts some IP data points near the classification boundary of the target model after it has been trained using the standard training algorithm. Moreover, the label predicted by the target model for an IP data point is treated as its IP label. Note that watermarking modifies the training process of the target model, while fingerprinting does not. Given a suspect model, fingerprinting detects it as a stolen version if it predicts the same IP labels for a large fraction of the IP data points, i.e., the suspect model has very similar classification boundary with the target model. Technically, fingerprinting finds an IP

data point x via the following optimization problem [4]:

$$\min_{x} \text{ReLU}(Z_i(x) - Z_j(x) + k) + \text{ReLU}(\max_{t \neq i,j} Z_t(x) - Z_i(x)), \quad (1)$$

where $\text{ReLU}(\cdot)$ is the ReLU activation function, i, j are randomly sampled labels, $Z_i(x)$ is the ith logit of the target model, and k is a hyperparameter. A model owner can find multiple IP data points via solving the optimization problem using different initialized x and label pairs i, j.

Certified robustness via randomized smoothing: *Randomized smoothing* [7] is a popular technique originally designed to build provably robust classifiers against adversarial examples. Specifically, given a classifier and a testing input, randomized smoothing adds random Gaussian noise to the testing input and uses the classifier to predict the noisy testing input. Randomized smoothing can be divided into *majority vote smoothing* [7] and *median smoothing* [6] depending on how to derive the predicted label of the original testing input based on the predicted label of the noisy one. For instance, majority vote smoothing constructs multiple noisy testing inputs via adding random Gaussian noise to the testing input and takes a majority vote among the predicted labels of them as the final predicted label of the original testing input.

MS framework[3] extended median smoothing to build provably robust watermarking methods for IP protection. Specifically, given a suspect model and a set of IP data points and labels of a watermarking method for a target model, they add random Gaussian noise to the suspect model to construct multiple noisy suspect models; calculate MR of the IP data points and labels for each noisy suspect model; and take the median MR among the noisy suspect models as the final MR for the original suspect model. They can show that their MR for a suspect model is larger than a threshold (i.e., CMR) when the ℓ_2 distance between the target model and the suspect model is bounded. However, their method suffers from two limitations: 1) median smoothing achieves suboptimal CMR, and 2) they only studied watermarking IP protection methods. The first limitation is because median smoothing is intrinsically less robust than majority vote smoothing since the labels of an IP data point predicted by the noisy suspect models are considered independently. In particular, only one suspect model's MR (i.e., the median MR) is considered as the final MR in median smoothing.

Ambiguity attacks to IP protection: Our work focuses on robustness of IP protection. Security against *ambiguity attacks* [8] is another desired property of IP protection, which is orthogonal to robustness. In particular, in an ambiguity attack, instead of post-processing a target model, an attacker constructs IP data points and labels that have a large MR for the target model. As a result, both the model owner and attacker have valid IP data points to verify the ownership of the target model, making it hard to verify the true ownership. A formal framework [1] proposed a cryptographic signature based watermarking method that is provably secure against some ambiguity attacks, and DeepIPR [8] proposed a white-box watermarking method for neural networks that is secure against ambiguity attacks.

3. Measuring Robustness of Existing Methods

We show that existing IP protection methods, including both watermarking and fingerprinting, are not robust against deliberated post-processing that adds small carefully crafted perturbations to the parameters of a target model.

3.1. Worst-case Post-processing

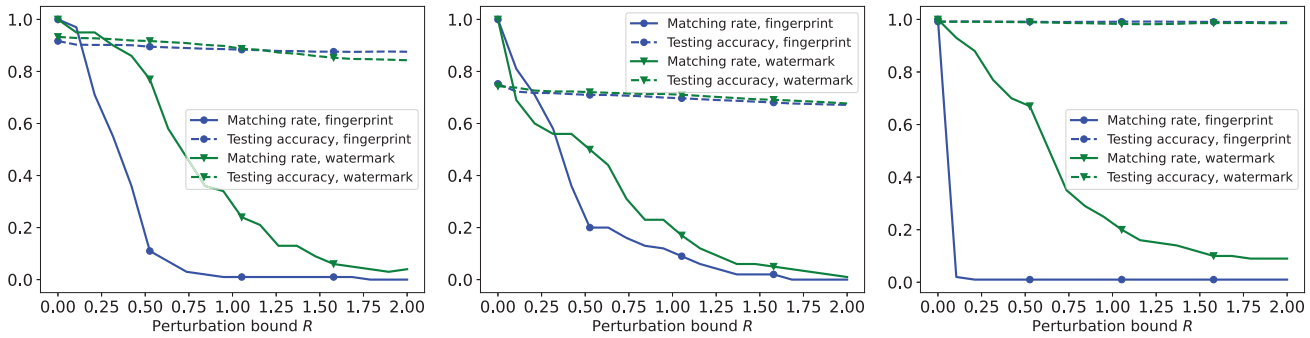
Formulating worst-case post-processing as an optimization problem: Suppose θ represents the parameters of a target model. An IP protection method has n IP data points and labels, which we call *IP dataset* and denote as $D_{IP} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$. We aim to measure the robustness of the IP protection method against post-processing in the worst-case scenarios, where an attacker knows the IP protection method and its IP dataset, e.g., an insider of the IP protection system sells such information to an attacker. In particular, our post-processing aims to perturb the target model to achieve two goals: 1) MR for the IP dataset is low, and 2) the perturbed model has similar accuracy with the target model. Towards the first goal, our post-processing method adds perturbation δ to the model parameters θ such that the perturbed model $\theta + \delta$ predicts labels for the IP data points that do not match with the corresponding IP labels. Towards the second goal, we assume a validation dataset D_v is available for post-processing and the perturbed model $\theta + \delta$ has a small loss (i.e., is accurate) for D_v. Formally, we formulate finding perturbation δ as the following optimization problem:

$$\min L(\delta) = -\lambda \cdot \frac{1}{|D_{IP}|} \sum\nolimits_{(x,y) \in D_{IP}} l(\theta + \delta, x, y) + \frac{1}{|D_v|} \sum\nolimits_{(x,y) \in D_v} l(\theta + \delta, x, y), \quad (2)$$

$$s.t. \ ||\delta||_2 \leq R, \quad (3)$$

where l is a loss function (e.g., cross-entropy loss in our experiments), the first term $-\frac{1}{|D_{IP}|} \sum_{(x,y) \in D_{IP}} l(\theta, x, y)$ quantifies the first goal, the second term quantifies the second goal, λ is a hyperparameter to balance them, and R is the perturbation bound.

Solving the optimization problem via PGD: We can obtain the perturbation δ via solving the above optimization problem. For instance, we can use the popular *projected gradient descent (PGD)* [19] to solve the optimization problem. Specifically, the perturbation δ is initialized to be 0. In each iteration, we compute the gradient of the objective function



(a) CIFAR-10 (b) CIFAR-100 (c) MNIST

Figure 1. Results of our post-processing method on the three datasets.

$L(\delta)$ with respect to δ and move δ towards the inverse of the gradient by a small step (called *learning rate*). If the ℓ_2 norm of the updated δ is larger than the perturbation bound R, we project it so its ℓ_2 norm is R. We repeat the process for *max_iter* iterations. Algorithm 1 shows our algorithmic details.

Algorithm 1 Our post-processing to existing IP protection methods

Input: Parameters θ of a target model, IP dataset D_{IP}, validation dataset D_v, learning rate lr, perturbation bound R, and maximum number of iterations *max_iter*.

Output: Perturbation δ

$\delta \leftarrow 0$
for $iter$ = 1 to *max_iter* **do**
  $g \leftarrow \nabla_\delta L(\delta)$
  $\delta \leftarrow \delta - lr \cdot g$
  if $\|\delta\|_2 > R$ **then**
    $\delta \leftarrow \delta \cdot \frac{R}{\|\delta\|_2}$
  end if
end for

3.2. Experiments

Experimental setup: We use three benchmark datasets: CIFAR-10 [13], CIFAR-100 [13], and MNIST [15]. The target model architecture is ResNet-18 [10]. We divide each training dataset into two halves. One half is used to train a target model, while the other half is used to post-process the target model. Moreover, we train another 50 models with different initialization using the half of training data for each dataset, and we treat them as *non-stolen models* since they are trained from scratch instead of post-processing of the target model. When training a model, we use a learning rate of 0.05, momentum of 0.9, weight decay of 1e-4, and batch size of 256. We train each model for 100 epochs.

We use a popular watermarking method [2] and a fingerprinting method [4] to protect a target model, though our results are also applicable to other methods. For the watermarking method, we pick 100 images and labels as IP dataset D_{IP} following [2]. The IP dataset is used to augment the training data when training a target model. For fingerprinting, we follow the settings in IPGuard [4]. Specifically, after a target model is trained, we extract 100 IP data points from it via solving the optimization problem in Eq. 1 using $k = 5$ as well as random initialization of x and label pairs i, j. For each IP data point, we use the target model to predict its label, which we treat as the corresponding IP label.

Table 1. Matching rates of the 50 non-stolen models.

Method	Dataset	Median	Maximum
Watermark	CIFAR-10	0.10	0.15
	CIFAR-100	0.01	0.04
	MNIST	0.11	0.14
Fingerprint	CIFAR-10	0.03	0.14
	CIFAR-100	0.00	0.03
	MNIST	0.02	0.11

Experimental results: In our post-processing, we set learning rate $lr = 0.001$, max_iter=1,000, and D_v includes 1,000 examples sampled from the testing dataset uniformly at random. Given a target model and a perturbation bound R, we use our post-processing method to generate a perturbation and construct a perturbed model. Then, we calculate 1) MR of the perturbed model for the IP dataset, and 2) testing accuracy of the perturbed model on the corresponding testing dataset excluding the validation data.

Figure 1 shows MR and testing accuracy of the perturbed model when the perturbation bound R increases, for both watermarking and fingerprinting. Our results show that our post-processing method can substantially decrease MR via adding a small perturbation to the target model. For instance, MR reduces to almost 0 when the perturbation bound is 2 for CIFAR-10 dataset. Table 1 shows the MRs of the 50 non-stolen models, where "Median" and "Maximum" are the median and maximum MRs of the 50 non-stolen models. Our results show that existing IP protection methods are not robust against small perturbations added to the target model

Table 2. Results of fine-tuned and distilled models.

Method	Dataset	Fine-tuning			Distillation		
		Accuracy	MR	Perturbation	Accuracy	MR	Perturbation
Watermark	CIFAR-10	0.92	0.61	15.76	0.93	0.80	15.08
	CIFAR-100	0.70	0.36	18.23	0.71	0.72	16.08
	MNIST	0.99	0.39	17.91	0.99	0.50	16.25
Fingerprint	CIFAR-10	0.92	0.65	16.67	0.92	0.65	15.20
	CIFAR-100	0.71	0.03	18.41	0.72	0.36	16.03
	MNIST	0.99	0.29	17.24	0.99	0.42	16.05

in the worst-case scenarios. In particular, when a target model is perturbed by our post-processing method, an IP protection method cannot distinguish between the perturbed model and a non-stolen model using MR, while the testing accuracy of the perturbed model is maintained.

Table 2 shows the testing accuracy, MR, and ℓ_2-norm perturbation of a perturbed model post-processed from a target model using fine-tuning and distillation. The ℓ_2-norm perturbation is the ℓ_2 distance between a perturbed model and a target model. In our experiments, for each dataset, we use half of the training data to fine-tune or distill a target model for 50 epochs (Section A.1 in Appendix shows more details on parameter settings). Our results show that, compared to our post-processing, these post-processing techniques introduce larger perturbations to a target model but the MR is still large. Take CIFAR-10 as an example, fine-tuning the target model introduces around 15.76 ℓ_2-norm perturbation but watermarking still achieves 0.61 MR. In other words, existing popular post-processing methods give us false sense of robustness in the worst-case scenarios.

4. Our IPCert

4.1. A Majority Vote Framework

Given a model and an IP dataset, IPCert computes a MR via adding Gaussian noise to the model. IPCert leverages majority vote smoothing [7], which was originally designed to build robust classifiers against adversarial examples. In particular, majority vote smoothing adds Gaussian noise to a testing input when building a robust classifier, while we add Gaussian noise to a model.

Predicting label for an IP data point: Given an IP dataset $D_{IP} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ and a model θ (e.g., target model, suspect model), where the IP dataset may be obtained via a watermarking or fingerprinting method. Our IPCert adds random Gaussian noise to the model before using it to predict the label of an IP data point. For simplicity, we denote by $F(\theta, x)$ the label for a data point x predicted by a model whose parameters are θ. Since we add random Gaussian noise ε to the model θ, the predicted label $F(\theta + \varepsilon, x)$ for an IP data point x is also random. We denote by p_c the probability that the predicted label $F(\theta + \varepsilon, x)$ is c. Our IPCert predicts the label with the largest probability p_c for IP data point x. Formally, we denote by $G(\theta, x)$ the label for x predicted by our IPCert when the model is θ, and $G(\theta, x)$ is defined as follows: $G(\theta, x) = \underset{c \in \mathcal{Y}}{\operatorname{argmax}} \Pr(F(\theta + \varepsilon, x) = c)$, where $\mathcal{Y}$ is the set of labels, $\varepsilon \sim \mathcal{N}\left(0, \sigma^2 I\right)$ is an isotropic Gaussian noise, and σ is the standard deviation of the Gaussian noise. $G(\theta, x)$ can be viewed as the majority-vote label among the labels predicted by multiple noisy suspect models for x, as we will further elaborate in Section 4.3.

Matching rate (MR): MR of IPCert for a model θ is the fraction of IP data points whose predicted labels $G(\theta, x)$ match with the corresponding IP labels. Formally, we have:

$$\text{MR} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(G(\theta, x_i) = y_i), \tag{4}$$

where $\mathbb{I}$ is an indicator function whose value is 1 if $G(\theta, x_i) = y_i$ and 0 otherwise.

4.2. Deriving Certified Matching Rate

Given a model and an IP dataset, we derive a lower bound of MR, called *certified matching rate (CMR)*, for IPCert when an arbitrary, bounded perturbation is added to the model. Next, we first review a key theoretical result from majority vote smoothing and then derive a CMR for IPCert.

Certified radius for an IP data point: Given a model θ and an IP data point x, we denote by A and B the predicted labels with the largest and second largest probabilities when adding random Gaussian noise to θ in our IPCert. Therefore, our IPCert predicts label A for x. Moreover, we denote by $\underline{p_A}$ a lower bound of the probability of label A and by $\overline{p_B}$ an upper bound of the probability of the label B in our IPCert. We will show how to estimate the probability lower/upper bounds $\underline{p_A}$ and $\overline{p_B}$ in Section 4.3. Suppose an attacker adds perturbation δ to the model θ. Based on majority vote smoothing [7], our IPCert still predicts label A for x when the perturbation δ is bounded. Formally, we have the following lemma based on [7]:

Lemma 4.1. *F is a function that takes a model θ and an IP data point x as input and outputs predicted label c. $\varepsilon \sim \mathcal{N}\left(0, \sigma^2 I\right)$ is Gaussian noise. Suppose $A \in \mathcal{Y}$ and $\underline{p_A}, \overline{p_B} \in [0, 1]$ satisfy: $Pr\left(F(\theta + \varepsilon, x) = A\right) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{c \neq A} Pr(F(\theta + \varepsilon, x) = c)$. Then, we have*

$G(\theta + \delta, x) = A$ *for all* $\|\delta\|_2 < r$*, where*

$$r = \frac{\sigma}{2}\left(\Phi^{-1}\left(\underline{p_A}\right) - \Phi^{-1}\left(\overline{p_B}\right)\right), \tag{5}$$

where Φ^{-1} *is the inverse cumulative distribution function of the standard Gaussian distribution.*

We call r *certified radius* for an IP data point x. Based on Lemma 4.1, we can obtain a certified radius r_i for each IP data point x_i, where $i = 1, 2, \cdots, n$. In other words, the label predicted by IPCert for x_i does not change when the perturbation added to the model is bounded by r_i.

Certified matching rate (CMR): Given a model θ, an IP dataset D_{IP}, and a perturbation bound R, we derive a lower bound of MR, i.e., CMR. Specifically, CMR is the least fraction of the IP data points, for which the labels predicted by IPCert match with the IP labels no matter what perturbation is added to the model once its ℓ_2-norm is bounded by R. Formally, we have CMR as follows:

$$\text{CMR} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(G(\theta, x_i) = y_i) \cdot \mathbb{I}(r_i > R), \tag{6}$$

where $\mathbb{I}$ is an indicator function, $G(\theta, x_i)$ is the label for x_i predicted by IPCert, and r_i is the certified radius for x_i. A model owner can calculate a CMR for its target model (i.e., the model θ is the target model). When an attacker steals the target model and post-processes it, the matching rate for the post-processed model is at least CMR no matter what post-processing (e.g., the worst-case post-processing in Section 3) is used once the ℓ_2-norm perturbation introduced by the post-processing is bounded by R.

4.3. Computing (Certified) Matching Rate

Given a model θ and an IP dataset D_{IP}, we compute MR of the model in our IPCert. Moreover, given a bound R of the perturbation that can be added to the model, we further compute the CMR of the model. The key to compute MR is to estimate the predicted label $G(\theta, x)$ for each IP data point x. A key challenge to compute CMR is to calculate the certified radius of each IP data point according to Lemma 4.1, and the key challenge to calculate the certified radius is how to estimate $\underline{p_A}$ and $\overline{p_B}$ for each IP data point. Next, we describe a Monte Carlo method to address the challenge.

Given a model θ and an IP data point x, we construct m ($m = 10,000$ in our experiments) noisy models via adding random Gaussian noises $\epsilon_1, \epsilon_2, \cdots, \epsilon_m$ to the model. We use each noisy model to predict the label of x, and we define the count m_c for label c as $m_c = \sum_{j=1}^{m} \mathbb{I}\left(F(\theta + \varepsilon_j, x) = c\right)$, where m_c is the number of noisy models that predict label c for x. $G(\theta, x)$ is the label (denoted as A) with the largest count. Given the predicted label $G(\theta, x)$ for each IP data point, we can calculate the MR of the model according to Eq. 4.

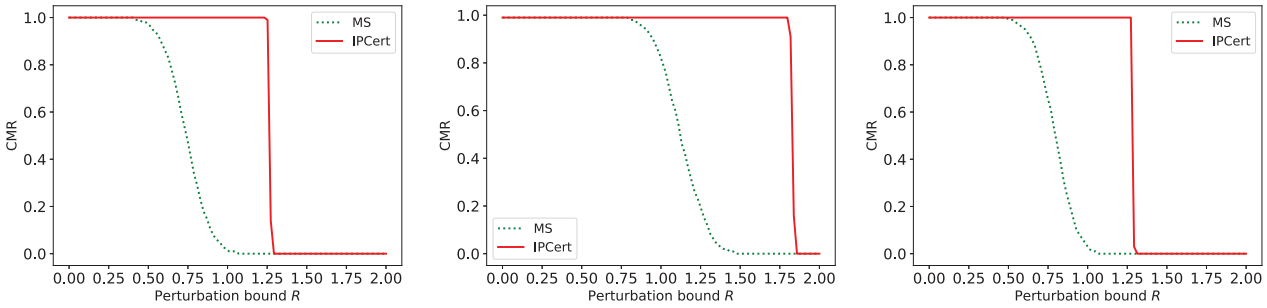
The count m_c follows a binomial distribution with parameters m and p_c, i.e., $m_c \sim \text{Binomial}(m, p_c)$. Therefore, we can use one-sided simultaneous confidence interval estimation to estimate $\underline{p_A}$ and $\overline{p_B}$. Specifically, according to *Certified_Topk* [12], we have $\underline{p_A} = B\left(\frac{\alpha}{|\mathcal{Y}|}; m_A, m - m_A + 1\right)$ and $\overline{p_c} = B\left(1 - \frac{\alpha}{|\mathcal{Y}|}; m_c + 1, m - m_c\right), \forall c \neq A$, where $|\mathcal{Y}|$ is the total number of classes, $B(\beta; a, b)$ is the βth quantile of the Beta distribution with parameters a and b, and the simultaneous confidence level for the above estimations is $1 - \alpha$. Then, we have $\overline{p_B} = \max_{c \neq A}\{\overline{p_c}\}$. After estimating $\underline{p_A}$ and $\overline{p_B}$ of an IP data point, we can calculate its certified radius according to Eq. 5. Given the certified radius of each IP data point, we can calculate the CMR of the model according to Eq. 6.

4.4. Optimizing Certified Matching Rate

Watermarking: In our IPCert, we add random Gaussian noise to a model before using it to classify an IP data point. When a noisy model is more likely to correctly classify an IP data point, the CMR may be larger. However, in standard training, a model, instead of a noisy one, is trained to learn the correlations between an IP data point and its label. As a result, the noisy model is likely to misclassify IP data points, leading to suboptimal CMR. To address the limitation, we propose to add Gaussian noise to the model during training. In particular, in each epoch, we first update the model following the standard training process. Then, we add multiple Gaussian noise to the model and calculate the average loss of the noisy models for the IP dataset, whose gradient is used to update the model. Algorithm 2 in Appendix shows the details of training with noise.

Fingerprinting: Fingerprinting extracts IP data points from a model that has already been trained. Therefore, instead of adding noise to the model during training, we propose a new method to extract IP data points after a model has been trained to maximize CMR. According to Eq. 6, CMR relies on two terms $\mathbb{I}(G(\theta, x_i) = y_i)$ and $\mathbb{I}(r_i > R)$. Therefore, we introduce *classification loss* and *robustness loss* to quantify them, respectively. The losses are smaller when the two terms are better satisfied across the IP dataset. Therefore, we select an IP dataset that has smaller classification and robustness losses.

Suppose we are given a model θ and we aim to extract an IP data point x with label j. We sample s (we use 16 in our experiments) Gaussian noise from $\mathcal{N}\left(0, \sigma^2 I\right)$, which we denote as $\epsilon_1, \epsilon_2, \cdots, \epsilon_s$. We use these noise to estimate $G(\theta, x)$, $\underline{p_A}$, and $\overline{p_B}$, based on which we define our classification and robustness losses. Define $\hat{g}_c(\theta, x) = \frac{1}{s}\sum_{t=1}^{s} g_c(\theta + \epsilon_t, x)$ for each label c, where g_c is the output probability of a model for label c. Then, we define classification loss L_C and robustness loss L_R as follows: $L_C = \text{ReLU}(\max_{c \neq j} \hat{g}_c(\theta, x) - \hat{g}_j(\theta, x))$ and $L_R =$



(a) CIFAR-10 (b) CIFAR-100 (c) MNIST

Figure 2. CMR of MS and IPCert for watermark on the three datasets.

$\frac{\sigma}{2}\text{ReLU}(\gamma - \xi(\theta, x))$, where $\xi(\theta, x) = \Phi^{-1}\left(\hat{g}_j\left(\theta, x\right)\right) - \Phi^{-1}\left(\max_{t \neq j} \hat{g}_t\left(\theta, x\right)\right)$ and γ is a hyperparameter (we set it to 8 in our experiments). In calculating $\xi(\theta, x)$, we use $\hat{g}_j\left(\theta, x\right)$ and $\max_{t \neq j} \hat{g}_t\left(\theta, x\right)$ to approximate $\underline{p_A}$ and $\overline{p_B}$, respectively. Finally, we define a loss function:

$$L(x) = L_{IP} + \eta(L_C + L_R), \tag{7}$$

where L_{IP} is the loss in Eq. 1 defined by IPGuard [4] and η is a hyperparameter. We find an IP data point x with label j via minimizing $L(x)$ using gradient descent. Specifically, we randomly initialize x and pick label i in L_{IP} (label j is already given) and iteratively update x until convergence. Algorithm 3 in Appendix shows the details.

5. Experiments

5.1. Experimental Setup

Datasets, models, and IP datasets: We use the same settings in Section 3.2. In particular, we use three datasets (CIFAR-10, CIFAR-100, and MNIST). For each dataset, we train a target model and 50 non-stolen models using the parameter settings in Section 3.2. One difference is that for watermarking, we add Gaussian noise when training a target model as described in Section 4.4. For watermarking, we obtain an IP dataset for each target model following Section 3.2. For fingerprinting, we obtain an IP dataset following our optimization strategy described in Section 4.4.

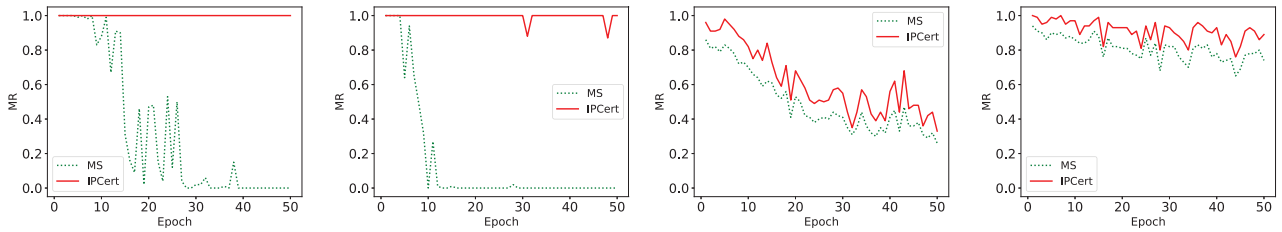
Evaluation metrics: We use CMR and MR as evaluation metrics. Given a target model, an IP dataset, and a perturbation bound R, we can calculate CMR of the target model. Such CMR is the least MR for a suspect model that is post-processed from the target model no matter what post-processing an attacker uses, once the ℓ_2-norm perturbation introduced by the post-processing is bounded by R. CMR does not depend on specific post-processing an attacker may use. MR is used to measure performance of an IP protection method when a specific post-processing method may be used by an attacker. In particular, given a post-processing method, we use it to post-process a target model and then calculate the MR for the post-processed model. Once the MR of a post-processed model is larger than those of the non-stolen models, the post-processed model and the non-stolen models can still be distinguished.

Compared method: We compare our IPCert with median smoothing (MS)-based IP protection framework [3]. Both IPCert and MS can turn a watermarking or fingerprinting method to be provably robust. Given a model and an IP dataset, MS uses the following way to compute MR: 1) the model owner constructs multiple noisy models via adding random Gaussian noise to the original model; 2) the model owner calculates MR for each noisy model by using it to predict labels for the IP data points; and 3) the model owner takes the median among all MRs as the final MR for the original model.

Parameter setting: Our IPCert has parameters: confidence level $1 - \alpha$, the number of noisy models m, and noise level σ. Unless otherwise mentioned, we set them as follows: $\alpha = 0.001$ and $m = 10,000$. Since fingerprinting does not interfere with the training process of the target model, while watermarking injects noise during training, fingerprinting can tolerate smaller perturbation. Therefore, we set $\sigma = 1$ for watermarking and $\sigma = 0.01$ for fingerprinting.

5.2. Experimental Results

IPCert achieves higher or comparable CMRs than/with MS: Figures 2 and 5 (in Appendix) respectively show the CMR of MS and IPCert for watermarking and fingerprinting on the three datasets. Our results show that, compared to MS, IPCert achieves higher or comparable CMR. The reason is that majority vote smoothing is intrinsically more robust than median smoothing since the labels of an IP data point predicted by the noisy models are considered independently in median smoothing. In other words, our IPCert can tolerate a larger or comparable perturbations added to a target model. For instance, on CIFAR-10 dataset, as shown in Figure 2(a), the labels of the IP data points predicted by our IPCert always match with the corresponding IP labels as long as the perturbation is bounded by 1.25.



(a) Watermark with fine-tuning (b) Watermark with distillation (c) Fingerprint with fine-tuning (d) Fingerprint with distillation

Figure 3. MR of MS and IPCert against fine-tuning and distillation on CIFAR-10. Figure 8 and 9 in Appendix show the results for the other two datasets.

Table 3. MRs of the 50 non-stolen models.

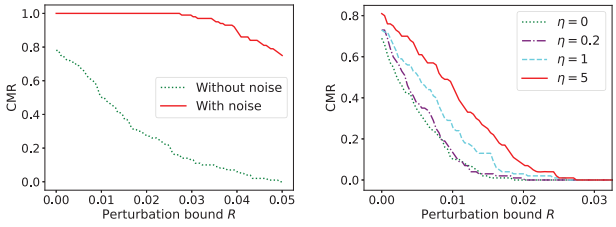
Method	Framework	Median	Maximum
Watermark	MS	0.12	0.16
	IPCert	0.12	0.15
Fingerprint	MS	0.08	0.14
	IPCert	0.06	0.15

However, for MS, a large fraction of IP data points may be misclassified if the perturbation added to a target model is larger than 0.75.

IPCert achieves higher MRs than MS against popular post-processing: Figure 3 shows the MRs of MS and IPCert for a post-processed model on CIFAR-10 dataset when an attacker uses fine-tuning and distillation to post-process a target model for 50 epochs. The post-processing settings are the same as those in Section 3.2. We also compare IPCert with MS on different datasets (CIFAR-100 and MNIST). The results in Figure 8 and 9 (in Appendix) show that for each dataset, IPCert performs better than MS for both watermark and fingerprint against fine-tuning and distillation. Our results show that IPCert achieves higher MRs than MS when an attacker post-processes a target model using fine-tuning or distillation. Note that our results for MS are different from those in median smoothing framework [3] due to different training settings.

IPCert and MS achieve similar MRs for non-stolen models: Table 3 summaries the MRs of the 50 non-stolen models for watermark and fingerprint on CIFAR-10 dataset, where "Median" and "Maximum" denote the median and maximum MRs of the 50 non-stolen models. We observe that IPCert and MS achieve comparable MRs for the non-stolen models, which means that IPCert improves provable robustness against perturbations added to a target model without increasing MRs for non-stolen models.

Optimization strategies to improve CMR: For watermark, we add isotropic Gaussian noise to a target model during training. For fingerprint, we propose a new method that considers robustness when selecting IP data points, where the hyperparameter η controls the selection. In our prior experiments, we set $\eta = 0$. Figure 4(a) shows the CMRs of IPCert for watermark on CIFAR-10 dataset, when the target model is trained without noise (i.e., the standard training algorithm) or with noise; while Figure 4(b) shows the CMRs of IPCert for fingerprint when η varies. Since the starting points of CMR curves are all 1 when $\sigma = 0.01$, we set $\sigma = 0.015$ in these experiments to show more details. Moreover, we show the results for small perturbation bounds to better contrast the differences. Our results show that our optimization strategies, i.e., training with noise for watermark and considering robustness to select IP data points for fingerprint, substantially increase CMRs.



(a) (b)

Figure 4. (a) Training with vs. w/o noise for watermark. (b) Varying η for selecting IP data points in fingerprint.

Impact of parameters: Figure 6 and 7 (in Appendix) show the impacts of confidence level $1-\alpha$, the number of noisy models m, and noise level σ on IPCert for CIFAR-10 dataset. We observe that 1) CMR increases slightly as α increases, 2) CMR also increases as m increases because we can estimate $\underline{p_A}$ and $\overline{p_B}$ more accurately using more noisy models, which in turn gives a larger certified radius for each IP data point and thus larger CMR, and 3) for watermark, when $\sigma = 0.5$, IPCert obtains the largest CMR, and we can optimize CMR via choosing appropriate value for σ.

6. Conclusion

We first find that existing IP protection methods are not robust against deliberated post-processing of a target model as previously thought in the worst-case scenarios. We then propose a novel majority vote smoothing framework called IPCert that can turn an existing IP protection method to be provably robust against post-processing. Moreover, training with noise for watermark and considering robustness to select IP data points for fingerprint can further enhance provable robustness of IPCert.

Acknowledgement

This work was supported by NSF grant No. 1937786, No. 1937787, No. 2125977, as well as ARO grant No. W911NF2110182.

References

[1] André Adelsbach, Stefan Katzenbeisser, and Helmut Veith. Watermarking schemes provably secure against copy and ambiguity attacks. In *Proceedings of the 3rd ACM workshop on Digital rights management*, pages 111–119, 2003.

[2] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, 2018.

[3] Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

[4] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 14–25, 2021.

[5] Laurent Charette, Lingyang Chu, Yizhou Chen, Jian Pei, Lanjun Wang, and Yong Zhang. Cosine model watermarking against ensemble distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9512–9520, 2022.

[6] Ping-yeh Chiang, Michael Curry, Ahmed Abdelkader, Aounon Kumar, John Dickerson, and Tom Goldstein. Detection as regression: Certified object detection with median smoothing. *Advances in Neural Information Processing Systems*, 33:1275–1286, 2020.

[7] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[8] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in neural information processing systems*, 32, 2019.

[9] Jia Guo and Miodrag Potkonjak. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[11] Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10758–10766, 2022.

[12] Jinyuan Jia, Xiaoyu Cao, Binghui Wang, and Neil Zhenqiang Gong. Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing. In *International Conference on Learning Representations*, 2019.

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[14] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

[15] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *Available: http://yann. lecun. com/exdb/mnist*, 1998.

[16] Hanwen Liu, Zhenyu Weng, and Yuesheng Zhu. Watermarking deep neural networks with greedy residuals. In *ICML*, pages 6978–6988, 2021.

[17] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. *arXiv preprint arXiv:1912.00888*, 2019.

[18] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. Distortion agnostic deep watermarking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13548–13557, 2020.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2017.

[20] Zirui Peng, Shaofeng Li, Guoxing Chen, Cheng Zhang, Haojin Zhu, and Minhui Xue. Fingerprinting deep neural networks globally via universal adversarial perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13430–13439, 2022.

[21] Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. Deepsigns: A generic watermarking framework for ip protection of deep learning models. *arXiv preprint arXiv:1804.00750*, 2018.

[22] Chen Sun and En-Hui Yang. A watermarking-based framework for protecting deep image classifiers against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3329–3338, 2021.

[23] Zhichuang Sun, Ruimin Sun, Long Lu, and Alan Mislove. Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1955–1972, 2021.

[24] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618, 2016.

[25] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.

[26] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE symposium on security and privacy (SP)*, pages 36–52, 2018.

[27] Peng Yang, Yingjie Lao, and Ping Li. Robust watermarking for deep neural networks via bi-level optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14841–14850, 2021.

[28] Innfarn Yoo, Huiwen Chang, Xiyang Luo, Ondrej Stava, Ce Liu, Peyman Milanfar, and Feng Yang. Deep 3d-to-2d watermarking: Embedding messages in 3d meshes and extracting them from 2d renderings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10031–10040, 2022.

[29] Ning Yu, Vladislav Skripniuk, Dingfan Chen, Larry Davis, and Mario Fritz. Responsible disclosure of generative models using scalable fingerprinting. *arXiv preprint arXiv:2012.08726*, 2020.

[30] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12805–12812, 2020.

[31] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 159–172, 2018.