Identification of the Adversary from a Single Adversarial Example

Minhao Cheng [1] Rui Min [1] Haochen Sun [1] Pin-Yu Chen [2]

Abstract

Deep neural networks have been shown vulnerable to adversarial examples. Even though many defense methods have been proposed to enhance the robustness, it is still a long way toward providing an attack-free method to build a trustworthy machine learning system. In this paper, instead of enhancing the robustness, we take the investigator's perspective and propose a new framework to trace the first compromised model copy in a forensic investigation manner. Specifically, we focus on the following setting: the machine learning service provider provides model copies for a set of customers. However, one of the customers conducted adversarial attacks to fool the system. Therefore, the investigator's objective is to identify the first compromised copy by collecting and analyzing evidence from only available adversarial examples. To make the tracing viable, we design a random mask watermarking mechanism to differentiate adversarial examples from different copies. First, we propose a tracing approach in the data-limited case where the original example is also available. Then, we design a data-free approach to identify the adversary without accessing the original example. Finally, the effectiveness of our proposed framework is evaluated by extensive experiments with different model architectures, adversarial attacks, and datasets. Our code is publicly available at `https://github.com/rmin2000/adv_tracing.git`.

1. Introduction

It has been shown recently that machine learning algorithms, especially deep neural networks, are vulnerable to adversarial attacks (Szegedy et al., 2014; Goodfellow et al., 2015). That is, given a victim neural network model and a correctly classified example, an adversarial attack aims to compute a small perturbation such that the original example will be misclassified with this perturbation added. To enhance the robustness against attacks, many defense strategies have been proposed (Madry et al., 2018; Zhang et al., 2019; Cheng et al., 2020a). However, they suffer from poor scalability and generalization on other attacks and trade-offs with test accuracy on clean data, making the robust models hard to deploy in real life. Therefore, in this paper, we turn our focus on the aftermath of adversarial attacks, where we take the forensic investigation to identify the first compromised model copy for generating the adversarial attack. In this paper, we show that given only a **single** adversarial example, we could trace the source copy that the adversary based for conducting the attack. As shown in Figure 1, we consider the following setting: a Machine Learning as a Service (MLaaS) provider will provide online model query access for a set of customers. The copy could also be distributed to every customer locally if deployed in time-sensitive applications such as auto-pilot systems. The model architecture and weight details are encrypted and hidden from the cus-

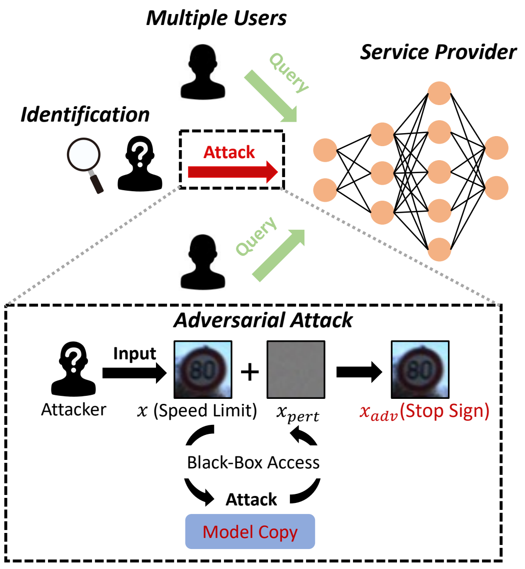

Figure 1. Illustration of the threat model where the malicious user conducts adversarial attacks through model query and uses the generated adversarial examples to attack other users' copies.

[1]Department of Computer Science & Engineering, The Hong Kong University of Science and Technology, Hong Kong [2]IBM Research, NY, USA. Correspondence to: Minhao Cheng <minhaocheng@ust.hk>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

tomers for the consideration of intellectual property (IP) protection and maintenance. In other words, every customer could only access the input and output of the provided copy but not the internal configurations. On the other side, the service provider has full access to every detail of all copies, including the training procedure, model architecture, and hyperparameters. However, there exists a malicious user who aims to fool the system by conducting adversarial attacks and gaining profit from the generated adversarial examples. Since all copies are based on the same model, adversarial examples generated by the adversary could be transferred to the other users' copies with a very high probability, 100% if all copies are the same. Thus it is critical for the interested party to conduct the investigation and trace the malicious user by identifying the compromised model copy. Taking the auto-pilot systems on the self-driving car as an example, the malicious user could conduct adversarial attacks on a road sign by querying his own vehicle model copy and then create an adversarial sticker to fool other vehicles using the same detection system. Also, online ML service provider such as ChatGPT could be compromised by malicious users conducting adversarial attack through model query and utilizing the generated examples to disseminate hate crimes and gain profits.

Only given adversarial examples as evidence, in order to make the tracing possible, adversarial examples generated by different copies have to be **unique** so that we are able to find the source copy and trace the malicious user in the end. To achieve this goal, we design a random mask watermarking strategy which embeds the watermark to the generated adversarial samples without sacrificing performance. At the same time, the proposed strategy is efficient and scalable that only needs a few iterations of fine-tuning. In the presence of the original example, a high-accuracy tracing method is proposed, which compares the adversarial perturbation with every model copy's masked pattern and the adversarial example's output distribution among different copies. Because it is not always practical to have the original example as a reference, in the second part, we further discuss the most challenging and practical attack setting where only the adversarial example is available to the investigator. Observing that the model copy's probability predictions on the same adversarial example would change significantly with a different watermark applied, we derive an effective rule to find the compromised copy. Specifically, based on the property that adversarial example is not robust against noise, we redesign the tracing metric based on the change in the predicted probabilities when applying different watermarks, which we expect the compromised model copy to minimize.

Comprehensive experiments are conducted on multiple adversarial attacks and datasets. When there is only a **single** adversarial example available, the results demonstrate that the two proposed methods could successfully trace the suspect model copy with over 74% accuracy on average with the data-limited case and data-free case. The tracing accuracy increases significantly to around 97% when there are two adversarial examples available.

Our contributions are summarized below:

- To the best of our knowledge, we are the first to propose a novel and scalable framework to make it possible to trace the compromised model copy by only using a single sample and its corresponding adversarial example.
- In the absence of samples used to generate adversarial examples, we further utilize the prediction difference of each copy with different watermarks to identify the adversary without any requirement on the original sample.
- Extensive experiments were conducted to demonstrate the effectiveness of the proposed framework to trace the compromised copy that the malicious users utilize to conduct different black-box adversarial attacks under different network architectures and datasets. We showed that the adversary could be traced with high accuracy in different scenarios, and the proposed framework has good scalability and efficiency.

2. Related Work

Adversarial Attack Since the discovery of adversarial example (Szegedy et al., 2014), many attack methods have been proposed. Roughly speaking, based on the different levels of information accessibility, adversarial attacks can be divided into white-box and black-box settings. In the white-box setting, the adversary has complete knowledge of the targeted model, including the model architecture and parameters. Thus, back-propagation could be conducted to solve the adversarial object by gradient computation (Goodfellow et al., 2015; Kurakin et al., 2017; Madry et al., 2018; Carlini & Wagner, 2017). On the other hand, the black-box setting has drawn much attention recently, where the attacker could only query the model but has no direct access to any internal information. Based on whether the model feedback would give the probability output, the attacks could be soft-label attacks or hard-label attacks. In the soft-label setting, ZOO attack (Chen et al., 2017) first proposed using a finite difference to estimate the gradient coordinate-wise and then conducted the gradient descent. It was then improved by selecting a better prior distribution (Ilyas et al., 2018b) and compressing the search space (Tu et al., 2019). To further increase the query efficiency, instead of calculating the full gradient, gradient-sign-based methods (Liu et al., 2018) have been proposed, and the attacks could still achieve a good success rate. Also, random-search-based methods such as Square attack (Andriushchenko et al., 2020) and

SimBA (Guo et al., 2019) have found that the attacks could be more successful in other domains such as frequency, and the square pattern could further improve the query efficiency. On the other hand, boundary attack (Brendel et al., 2017) first proposed the hard-label setting and used the random search method to find the adversarial attacks. It was then improved by (Chen et al., 2020) by finding a better sampling prior. OPT attack (Cheng et al., 2018) and Sign-OPT attack (Cheng et al., 2020b), on the other hand, formalized the hard-label attack into an optimization framework and used the zeroth-order method to solve it.

Watermarking Model watermarking is introduced in intellectual property protection on machine learning systems. It could be roughly divided into two categories: white-box and black-box watermarking, depending on the accessibility to the model and its parameters in order to extract the watermark. For the white-box watermarking, the first scheme for DNN (Uchida et al., 2017) tended to embed the watermark into the training process. Later, because of the extra capacity available in the state-of-art neural networks, DeepMarks (Chen et al., 2018) encoded the watermark to be a binary vector in the probability density function of trainable weights. However, it is not always convenient for the model owner to do the formal verification in the white-box watermarking, and the white-box watermark is also vulnerable to statistical attacks (Wang & Kerschbaum, 2019). Black-box methods are then proposed to solve the limitations mentioned above. DeepSigns (Darvish Rouhani et al., 2019) has done some improvement over (Uchida et al., 2017) and proposed a framework to incorporate the black-box case. Utilizing backdoor attacks to do watermarking is another big trend and has drawn much attention recently (Zhang et al., 2018; Adi et al., 2018). Adversarial examples are used to enable extraction of the watermark without requiring model parameters (Chen et al., 2019; Le Merrer et al., 2020). However, the forensic investigation of adversarial examples has a major difference from the model watermarking as the model watermarking must be shown in the generated adversarial examples. Also, it requires the tracking to be attack-agnostic. In other words, the tracing mark should be robust across different attacks simultaneously. However, the current watermarking methods in (Zhang et al., 2018; Adi et al., 2018) are either attack-dependent or have limited effect on the generated adversarial examples, which is not suitable for this task.

Traitor Tracing The proposed adversary tracing shares a lot of similarity with traitor tracing (Chor et al., 1994; Boneh & Franklin, 1999; Chor et al., 2000; Boneh & Naor, 2008) in cryptography where personal decryption keys are designed to trace the source of illegitimate keys. Although both methods use watermarking technology to distinguish distributed content and forensically trace compromised targets, the key difference is that in traitor tracing, the watermark embedded in the metadata is fixed and can be extracted by a predefined watermark extractor. However, in our proposed setting, the metadata would be the model copy, and we need to ensure that the generated adversarial examples display the watermark so that they can be traced.

Forensic investigation in Machine Learning: Although machine learning methods have already been used in forensic science (Carriquiry et al., 2019), there are a few studies on building trustworthy machine learning from a forensic perspective. Most papers focus on how to identify the model stealing attack by introducing the watermarking approaches to protect the intellectual property of the deep neural networks. That is to say, a unified and invisible watermark is hidden into models that can be extracted later as special task-agnostic evidence. However, to the best of our knowledge, we are the first paper to study the adversarial attack from a forensic investigation perspective.

3. Problem Setting

We formalize the identification of the compromised copy in the following setting. The machine learning service provider (owner) provides model query access to m customers (users) for the K-way classification task. Each user would be assigned a unique user ID i and we denote the model copy for user i to be f_i. As inference efficiency is critical in time-sensitive applications such as auto-pilot systems, these model copies could be also first encrypted for intellectual protection and security concerns and then distributed to the m customers (users). Therefore, in both cases, the customers only have black-box access to their own copies. In other words, the user i could only query his own copy f_i to get the prediction results without any access to the internal information about the model copy.

Unfortunately, a malicious user (adversary) exists who aims to fool the whole system, including other users' model copies, by conducting black-box adversarial attacks. Specifically, let the malicious user's copy be f_{att} (the *compromised copy*). As the attacker does not have access to query other users' copies, he/she then chooses to perform black-box attacks to his copy f_{att} to generate an adversarial example $\boldsymbol{x}_{adv}$. As all model copies are trained with the same dataset for the same classification task, the generated adversarial example could successfully lead to the misclassification of other users' copies. Our task is to find the compromised copy f_{att}.

To be noted, there could be multiple malicious users conducting colluded adversarial attack to fool the system. The investigator then aims to find all the attack participants. For the simplicity, we only focus on the single adversary setting and we will leave the multiple malicious users case in our future work.

4. Methodology

If the same model copy is distributed to every customer, the generated adversarial examples from different users' copies will be identical to each other, and it thus becomes impossible to trace the adversary by only giving adversarial examples. In the following section, we propose our framework which consists of two parts shown in Figure 2. First, we design a simple random mask watermarking method that would have a limited effect on the model copy's accuracy while embedding distinctive features in adversarial examples, distinguishing them from those generated from other copies. We further design a tail and head mechanism to make the training process scalable and efficient. Depending on the availability of the original example as a reference, we then propose two detection scenarios to identify the adversary from adversarial examples.

4.1. Random Mask Watermarking

Since we need to identify the compromised model from a large pool of customer copies, it requires us to assign a unique identification mark for every customer copy, and the mark should be reflected in the generated adversarial example.

In this section, we design a simple but effective method by applying a random watermark on each of the m model copies. As shown in Figure 2, for each copy $f_i (1 \leq i \leq m)$, we randomly select a set of pixels $\boldsymbol{w}^i$ as the *watermark* on the training samples. Formally, denote the input as $\boldsymbol{x} \in \mathbb{R}^{W \times H \times C}$. For every copy f_i, we randomly generate a binary matrix $\boldsymbol{w}^i \in \{0,1\}^{W \times H \times C}$ by sampling uniformly. We call the $\boldsymbol{w}^i$ *mask* for copy f_i, deciding the set of masked pixels. When $\boldsymbol{w}^i_{a,b,c} = 1$ for a specific pixel (a, b) at channel c, the value is set to be 0; otherwise, when $\boldsymbol{w}^i_{a,b,c} = 0$, the original pixel value is not modified. That is to say, for every input $\boldsymbol{x}$, the input after the mask $\tilde{\boldsymbol{x}}$ on copy f_i would be $\tilde{\boldsymbol{x}}^i_{a,b,c} := \boldsymbol{x}_{a,b,c} \cdot (1 - \boldsymbol{w}^i_{a,b,c})$ for each pixel (a, b) at each channel c. For simplicity, we use $\tilde{\boldsymbol{x}}^i = \boldsymbol{x} \odot (1 - \boldsymbol{w}^i)$ to denote the masked sample $\boldsymbol{x}_i$ in the whole paper, where $\odot$ represents the element-wise product.

Each input is first applied with the mask and then fed into the model copy in both the training and inference phases. To speed up the training process and make the pipeline scalable to thousands of users, we add each model copy with a few network layers as the head part h_i. The output of the head part will directly feed into a shared tail model t. In other words, we have each model copy as

$$f_i(\boldsymbol{x}) = t(h_i(\boldsymbol{x})), \tag{1}$$

Specifically, in the pretraining phase, we first train a model without the watermark from scratch as the base model. Then each model copy is assigned a unique copy head for the added specific watermark and shares a big common tail inherited from the base model. During the fine-tuning process, we freeze the parameters in the tail and embed the watermark to every customer copy by only fine-tuning the weights in the head part with a few epochs. Our following experiments will show it is sufficient to embed watermark to a few layers in DNNs without sacrificing accuracy. To be noted, the proposed method could be easily adapted to the new user case as we could just create a unique watermark and fine-tune the new head with a few epochs.

Since the distributed copy's detail is encrypted and not accessible in the black-box setting, users are unaware that their copies and the generated adversarial examples are watermarked. Moreover, as the watermarks are generated uniformly at random, each user's copy can be assigned a unique watermark even when the number of users is large. Furthermore, since deep neural networks have been shown to be robust against random noise (Fawzi et al., 2016; Cohen et al., 2019), the proposed watermarking would have a very limited effect on the performance, as shown in Table 1, when the masking ratio $\|\boldsymbol{w}^i\|_1$ is not too large.

In this paper, we only show a straightforward random mask watermarking scheme, which could already provide us with satisfactory tracing accuracy without hurting the performance. We leave the design of the better watermarking methods under this framework to future works.

4.2. Data-limited Adversary Identification

With the watermarking scheme described in Section 4.1, we can exploit the information embedded in the watermarked adversarial example (and the corresponding original example) to identify the compromised copy.

We first introduce the *data-limited* case where the corresponding original example $\boldsymbol{x}$, on which the given adversarial example $\boldsymbol{x}_{adv}$ is based, is available. A natural example of this setting is that the malicious user generates an adversarial sticker based on a road sign using his own self-driving car. As the self-driving system is built by the service provider and shared, the adversarial stickers would fool other passing vehicles. Then, as the investigator, we aim to trace the adversary by analyzing the road sign and the adversarial sticker. We now discuss the design of the detector in the data-limited case, which is inspired by the mechanism of adversarial attacks. Specifically, since the adversarial attack is formalized as an optimization problem, the adversary takes the gradient of the designed loss function $\mathcal{L}$ with respect to the input $\boldsymbol{x}$ to find the most effective perturbation.

Adversarial Perturbation: Formally, for the model f_i, the gradient of the designed loss function $\mathcal{L}$ with respect to

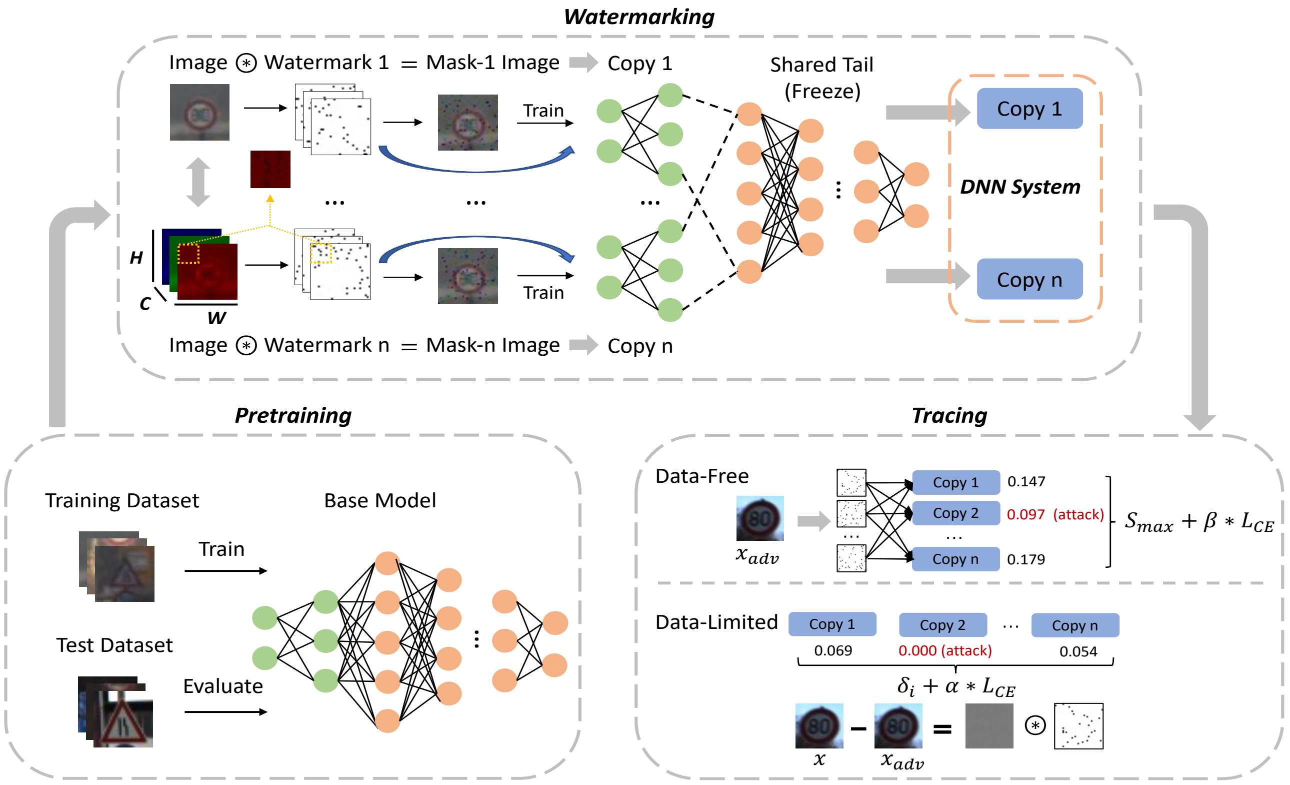

Figure 2. Proposed framework of identifying compromised model copy from adversarial examples.

the given sample $\boldsymbol{x}$ is

$$\frac{\partial \mathcal{L}(f_i(\tilde{\boldsymbol{x}}))}{\partial \boldsymbol{x}_{a,b,c}} = 0 \quad \text{if } \boldsymbol{w}^i_{a,b,c} = 1, \tag{2}$$

Since the black-box attacks are designed to approximate the gradients used in the white-box attacks, we could expect that the approximated gradients at the masked pixels would have a value close to 0 or be smaller in magnitude than the other pixels. The reason lies in the fact that the pixels within the watermark have zero contribution to the overall loss. Based on this observation, since we have access to the original example $\boldsymbol{x}$, we could calculate the adversarial perturbation $\boldsymbol{\delta} = \boldsymbol{x}_{adv} - \boldsymbol{x}$. If the adversarial example is generated by the compromised copy f_{att}, values in $\boldsymbol{\delta}$ should be much smaller in those coordinates in the corresponding watermark i.e coordinate that $\boldsymbol{w}^{att} = 1$. Notably, since different copies are embedded with individual watermarks, different adversarial examples generated by diverse copies would be distinguished in the region with low perturbation values. Therefore, given $\boldsymbol{x}_{adv}$ and $\boldsymbol{x}$, we thus calculate a score for each copy by summing up the absolute values of the adversarial perturbation overall masked pixels (of the corresponding model), i.e.,

$$\boldsymbol{\delta}^i = \sum_{a,b,c} \boldsymbol{w}^i_{a,b,c} \odot |\boldsymbol{x}_{adv} - \boldsymbol{x}|_{a,b,c} \tag{3}$$

Adversarial Stability: Moreover, we also observe that the cross-entropy loss between the prediction output of adversarial examples and the ground-truth label of clean examples differs among different copies. Since adversarial examples should be identical to original examples visually, the ground-truth label could be easily inferred. Specifically, with introduced masking watermark, some model copies prediction on the adversarial example will change even towards the label instead of deviating from it. That is, if the adversarial example $\boldsymbol{x}_{adv}$ is generated from copy f_i , the cross entropy loss $\mathcal{L}_{CE}(f_i(\boldsymbol{x}_{adv}), \boldsymbol{y})$ is smaller than $\mathcal{L}_{CE}(f_j(\boldsymbol{x}_{adv}), \boldsymbol{y})$ if $f_j(\boldsymbol{x}_{adv}) \neq \boldsymbol{y}$, $\forall j \neq i$, where $\boldsymbol{y}$ is the ground truth label of $\boldsymbol{x}$.

We then combine the two metrics and calculate the final score for each copy. Then, we take the copy with the smallest score as the compromised copy, i.e.,

$$att \leftarrow \arg\min_{1 \leq i \leq m}(\boldsymbol{\delta}^i + \alpha \mathcal{L}_{CE}(f_i(\boldsymbol{x}_{adv}), \boldsymbol{y})) \tag{4}$$

4.3. Data-free Adversary Identification

The previously introduced data-limited detector requires access to the original example as a reference, which is not realistic in many scenarios. Therefore, in the following section, we relax this constraint and discuss the tracing under the most challenging yet realistic setting where the only evidence available is the generated adversarial example. We propose a data-free detector based on the different copy outputs when applying different masks to the adversarial example.

Formally, for the given adversarial example $\boldsymbol{x}_{adv}$, we first apply every copy's watermark $\boldsymbol{w}^i, i \in [m]$ to create a set of masked adversarial examples $\{\tilde{\boldsymbol{x}}^i_{adv}\}_{i=1}^m$ where $\tilde{\boldsymbol{x}}^i_{adv} = \boldsymbol{x}_{adv} \odot (1 - \boldsymbol{w}^i)$. We then feed the masked adversarial examples set to each copy f_i to get its probability output. For every copy f_i, we get a probability output matrix $\boldsymbol{P}^i := [f_i(\tilde{\boldsymbol{x}}^1_{adv})^T, \dots, f_i(\tilde{\boldsymbol{x}}^m_{adv})^T] \in [0,1]^{m\times K}$, where each element in $\boldsymbol{P}^i$ is $\boldsymbol{P}^i_{a,b} = [f_i(\tilde{\boldsymbol{x}}^a_{adv})]_b$ and K is the number of classes.

Since adversarial examples are very close to the model's decision boundary (Brendel et al., 2017; Cheng et al., 2018), a slight perturbation to it would cause the model's prediction to change significantly. In other words, adversarial examples are sensitive to small perturbations, while ordinary examples are relatively more robust. It then inspires us to propose a metric based on this difference to detect the compromised model. Specifically, let us still assume the given adversarial example $\boldsymbol{x}_{adv}$ is from copy f_i. Then, when the corresponding watermark $\boldsymbol{w}^i$ is applied, the probability prediction will remain unchanged. However, when applying another watermark $\boldsymbol{w}^j, j \neq i$, it is likely that the watermarked adversarial example would be moved away from the decision boundary. Therefore, the maximal predicted class probability is generally larger after applying $\boldsymbol{w}_j$. At the same time, if the adversarial example is not generated from the copy, the extent of change would be limited. Therefore, we propose the max label score S_{max} based on the extent of change of prediction:

$$S^i_{max} = \frac{\max_{1\le k\le K} \boldsymbol{P}^i_{i,k}}{\sum_{1\le j\le m} \max_{1\le k\le K} \boldsymbol{P}^i_{j,k}} \tag{5}$$

We further combine the score of adversarial stability proposed in the data-limited case with the max label score to improve the detection accuracy:

$$att \leftarrow \arg\min_i (S^i_{max} + \beta \mathcal{L}_{CE}(f_i(\boldsymbol{x}_{adv}), \boldsymbol{y})) \tag{6}$$

5. Experimental Results

In this section, we extensively evaluate the proposed framework on a variety of adversarial attacks, datasets, and models. We first introduce the experiment setup and implementation details. In Section 5.1, we show the proposed random mask watermarking has a limited effect on the performance. We then test the tracing success rate on adversarial examples generated by different black-box attacks in the data-limited case and data-free case in Section 5.2.

Implementation Details We conduct our experiments on three popular image classification datasets GTSRB (Stallkamp et al., 2012), CIFAR-10 (Krizhevsky et al., 2009) and Tiny-ImageNet (Le & Yang, 2015). We use two widely used network architectures VGG16 (Simonyan & Zisserman, 2015) and ResNet18 (He et al., 2016). All our experiments were implemented in Pytorch and conducted using an RTX 3090 GPU. In the pretraining stage, the base model is trained with Adam optimizer with the learning rate of 10^{-3} and a batch size of 128 for 50 epochs. For every copy's watermark, we independently randomly sample 100 pixels to mask for both CIFAR-10 and GTSRB and increase the mask size to 400 pixels for Tiny-ImageNet which keep the masking rate around 3.26%. To embed different watermarks into individual copies efficiently, we separate the base model by taking the first several layers as the head and leaving the rest as the tail. We provide the time comparison of training a base model versus fine-tuning a model head in Appendix A. Specifically for ResNet18, we take the first residual block as the copy head, and we detach the first two convolution layers for VGG16. The parameter percentage of the head for ResNet18 and VGG16 is 1.34% and 0.26% respectively which is a very small portion of the whole network weights. We then fine-tune the weights in the head portion for 10 epochs while freezing the parameters of the model tail. More experiments with different watermark sizes and splits on the model head are conducted in Appendix B.1 and B.2.

Adversarial Attack Methods We perform the following five black-box adversarial attacks to generate the adversarial example:

- **Natural Evolutionary Strategy (NES):** Ilyas et al. (2018a) introduced a soft-label black-box adversarial attack that designed a loss on the output probability changes and used Neural evolution strategy (NES) to approximately estimate the gradient.
- **Bandits and Priors Attack (Bandit):** Ilyas et al. (2018b) introduced a soft-label black-box attack by using the bandit algorithm to find a better prior where the adversarial perturbation could be drawn with high probability.
- **Simple Black-box Attack (SimBA):** (Guo et al., 2019) introduced a soft-label black-box adversarial attack by sampling the perturbation direction from a predefined orthonormal basis. The sampled direction would be either added or subtracted from the target image to test its success.
- **Hop-Skip-Jump Attack (HSJ):** Chen et al. (2020) introduced a hard-label black-box attack which is applied the zeroth-order sign oracle to improve Boundary attack (Brendel et al., 2017).
- **Sign-OPT Attack (SignOPT):** Cheng et al. (2020b)

introduced hard-label black-box attack that uses a single query oracle to improve the query efficiency of OPT attack (Cheng et al., 2018).

For SimBA, HSJ and SignOPT attacks, we use Adversarial Robustness Toolbox (ART) (Nicolae et al., 2018)'s implementation. We use the default hyperparameters in the ART toolbox to conduct the attack. For the NES attack and Bandit attack, we re-implement them in Pytorch following the official implementation in `https://github.com/MadryLab/blackbox-bandits`. All the attacks are conducted in the ℓ_2 constraints and untargeted setting. The attack will be stopped when there is a successful adversarial example generated.

Evaluation Metric To evaluate the effectiveness of the proposed detection method, for each attack, we generate 10 **transferable** adversarial examples for every model copy. An adversarial example $\boldsymbol{x}_{adv}$ is defined as **transferable** if and only if the prediction of the compromised copy f_{att} is wrong and, at the same time, the prediction of at least one of the other $m-1$ copies is wrong. It is worth noting that it is rather difficult for the black-box attacks to have a good transferability over the proposed random mask watermarking so we consider 10 transferable examples are sufficient to test our detector's effectiveness. To sum up, for each attack, we have a total of 1000 adversarial examples under the setting of 100 copies. We then define the tracing accuracy to evaluate the detection rate defined as follows:

$$\text{Trace Acc} = \frac{N_{\text{correct}}}{N_{\text{total}}} \tag{7}$$

Where N_{correct} is the count of the correct identification of the compromised copy and N_{total} is the total number of the transferable adversarial example generated.

5.1. Model Performance with Random Mask Watermarking

In this section, we conduct experiments to verify whether all model copies could still maintain a good performance after applying the watermark. Specifically, we train 100 copies on three datasets CIFAR-10, GTSRB, and Tiny-ImageNet with two popular architectures VGG16 and ResNet18. We also add a baseline model without a watermark as a reference. In Table 1, it could be clearly observed that the accuracy of the watermarked copies has a similar performance compared with the baseline model. The worst accuracy drops are only around 1%, while both mean and median keep a very similar performance with the baseline. Concerning there exists randomness in the training procedure, the proposed watermarking method has a limited effect on the performance.

Table 1. The classification accuracies (%) of model copies with random mask watermarking on three datasets. We denote V- as the model trained with VGG16 and R- as the model trained with ResNet18.

Task	Baseline	Min	Mean	Median	Max
V-CIFAR10	90.70	89.30	89.71	89.72	90.20
R-CIFAR10	91.97	91.10	91.49	91.51	91.83
V-GTSRB	97.60	96.10	96.99	97.02	97.48
R-GTSRB	98.50	96.81	97.45	97.47	98.15
V-Tiny	45.99	45.08	45.52	45.55	45.82
R-Tiny	52.16	50.94	51.50	51.51	52.02

5.2. Identification Results

For identification in the data-limited setting, we conduct experiments on 100 copies applied with random masks. We set the hyperparameter α to 0.85 for CIFAR10, 0.5 for GTSRB, and 5 for Tiny-ImageNet and tested tracing accuracy on different attacks. We also show the results with different α in Section 5.4. The results in the top half of Table 2 illustrate that our detection method could identify the compromised copy successfully in all datasets and network architectures which achieves an average of 71.33% tracing accuracy with only one adversarial example. It is worth noting that a random guess would only result in a 1% tracing accuracy average with a total number of 100 copies. Specifically, our detection method is extremely effective to trace HSJ attack, where the average tracing accuracy is over 88.5%. However, it could also be seen that our tracing method has a poor performance on Bandit attack that achieves over 42% accuracy in the worst case, which may be affected by the noisy gradient estimation in the dimension reduction process. However, we believe it could be solved by introducing a better watermark design, which we leave as our future work.

As we further limit the accessibility, we trace the compromised model with only one adversarial example and show the tracing accuracy at the bottom half of Table 2. For the data-free case, we also set the hyperparameter β to 0.5 for three datasets. Although the original example is no longer available, we could still achieve a similar or even better tracing accuracy against some attacks. Specifically, the average tracing accuracy is 69.80% among different datasets and model architectures which is slightly lower than the data-limited case. However, we could see there is a significant tracing accuracy (23%) improvement in the SignOPT. It is because the data-free score is based on the property that the adversarial examples are more sensitive to a slight perturbation by different masks and more robust against noisy gradient estimation. Therefore, we could further combine those two scores in practice to have better tracing accuracy based on the available evidence.

Table 2. The tracing accuracies (%) in data-limited and data-free scenarios with only a single adversarial example available.

CASE	TASK	BANDIT	HSJ	NES	SIGNOPT	SIMBA	MEAN
DATA-LIMITED	V-CIFAR10	48.2	93.4	84.2	55.4	85.3	73.30
	R-CIFAR10	54.2	95.5	87.4	65.8	83.0	77.18
	V-GTSRB	42.1	98.7	86.3	56.9	91.0	75.00
	R-GTSRB	43.8	98.7	86.3	61.8	86.0	75.32
	V-TINY	47.5	64.1	51.4	54.2	78.7	59.18
	R-TINY	54.8	81.0	65.6	53.4	85.1	67.98
DATA-FREE	V-CIFAR10	65.8	83.9	71.6	85.7	59.2	73.24
	R-CIFAR10	69.3	89.4	77.8	90.5	56.4	76.68
	V-GTSRB	62.4	92.0	67.5	90.7	56.3	73.78
	R-GTSRB	61.8	92.8	73.2	91.5	52.7	74.40
	V-TINY	55.2	62.1	48.4	58.7	56.5	56.18
	R-TINY	58.5	72.3	59.2	70.1	62.4	64.50

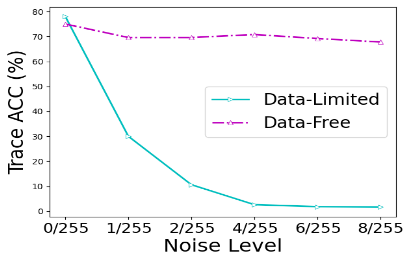

Figure 3. Average tracing accuracy on adaptive attack with different random noise levels.

Results on Adaptive Attacks To fully test the robustness of our proposed detectors, we also conducted an adaptive attack where the adversary has full access to the specific watermark embedded in each model. To be noted, it is not practical because users have only black-box access and it is not an easy task to directly infer which pixels are masked because of the noise estimation. Since our tracing method relies on the specific watermark embedded in individual model, we assume the adaptive attacker add Gaussian noise only to the pixels within the watermark without modifying the other region, which breaks our assumption proposed in Equation 2. We test the average tracing accuracy across different noise levels on CIFAR10 with ResNet18 structure. Our results are shown in Figure 3. We observe a significant accuracy drop in the data-limited case when adding random perturbation since we utilize the adversarial perturbation to identify the compromised model. However, we also notice that our data-free detector is not sensitive to random noise, which suggests that our tracing method can still be effective even if the adversary knows the predefined watermark.

5.3. Results on Multiple Adversarial Examples

In the previous experiments, we considered only one adversarial example, which is the most extreme case for forensic investigation. However, here comes a natural question: could the proposed method have a better detection rate if more adversarial examples are collected? In this section, we conduct experiments to answer this question. We use a simple strategy to combine multiple adversarial example scores. That is, we first calculate scores defined in Section 4.2 and Section 4.3 for each example, and then add up each score computed over all adversarial examples. Then we take the copy with the smallest sum as the compromised copy. We conduct the experiments on 100 copies of the random mask watermarked ResNet18 and VGG16 models for the CIFAR-10 dataset in both the data-limited and data-free settings. It could be seen in Figure 4 that the detection rate keeps increasing with the number of adversarial examples. We could get around 97% tracing accuracy on average when adding only 1 adversarial example to current accessibility. And the accuracy will reach 100% if given three or more adversarial examples. It shows our method is quite robust and has a great potential to be further improved.

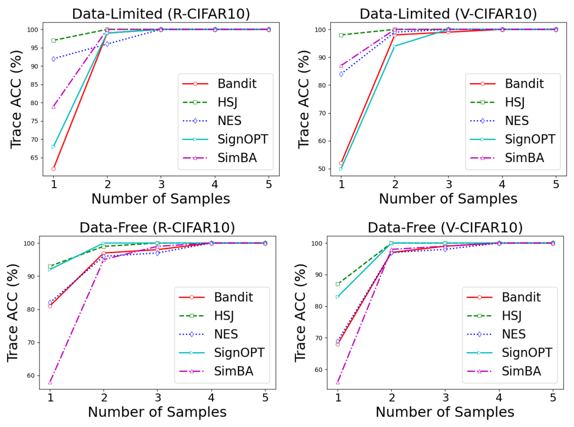

Figure 4. Tracing accuracy with multiple adversarial examples.

5.4. Results on Different α in Data-limited Identification

Since our objective in Eq. 6 is composed of two parts with the hyperparameter α, in this section, we conduct experiments to study the impact of α. We demonstrate the results obtained from ResNet18 and VGG16 on CIFAR-10 and GTSRB in Figure 5. From Figure 5, we observe that different attacks shows different sensitivity to the choice of α in terms of tracing accuracy. Although increasing the α would improve the tracing performance on Bandit and SignOPT, it will yield a slight performance drop against other types of attacks.

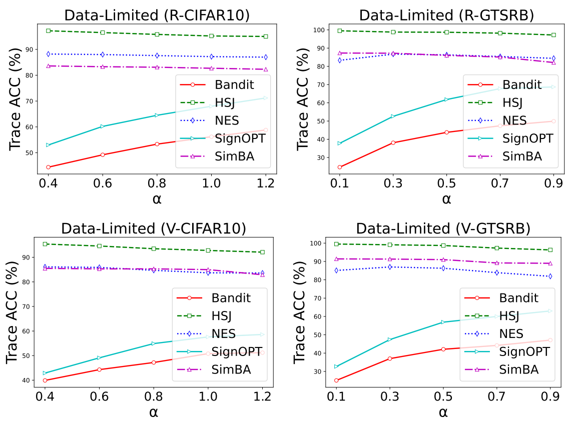

Figure 5. Tracing accuracy with different α values.

6. Conclusion and Limitations

In this paper, we develop the first framework for identifying the compromised model copy from a single adversarial example for forensic investigation. We first present a watermarking method to make the generated adversarial example unique. Depending on the accessibility of the original example, two identification methods are presented and compared. Our results demonstrate that the proposed framework has a limited effect on the model's performance and has a high success rate to find the compromised copy by only giving a single adversarial example. Our framework could further improve the detection rate to nearly 100% when two more adversarial examples are provided. As this study applied a simple yet effective random mask method to watermark model copies, future works could comprehensively study different watermarking methods in search of more efficient ways to distinguish the copies in the setting of this study. Moreover, more experiments could be conducted to design even more effective quantitative scores to identify the adversary in future works.

References

Adi, Y., Baum, C., Cisse, M., Pinkas, B., and Keshet, J. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1615–1631, 2018.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

Boneh, D. and Franklin, M. An efficient public key traitor tracing scheme. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, pp. 338–353. Springer, 1999.

Boneh, D. and Naor, M. Traitor tracing with constant size ciphertext. In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 501–510, 2008.

Brendel, W., Rauber, J., and Bethge, M. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pp. 39–57, 2017.

Carriquiry, A., Hofmann, H., Tai, X. H., and VanderPlas, S. Machine learning in forensic applications. *Significance*, 16(2):29–35, 2019.

Chen, H., Rohani, B. D., and Koushanfar, F. Deepmarks: A digital fingerprinting framework for deep neural networks. *arXiv preprint arXiv:1804.03648*, 2018.

Chen, H., Rouhani, B. D., and Koushanfar, F. Blackmarks: Blackbox multibit watermarking for deep neural networks. *arXiv preprint arXiv:1904.00344*, 2019.

Chen, J., Jordan, M. I., and Wainwright, M. J. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pp. 1277–1294. IEEE, 2020.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

Cheng, M., Le, T., Chen, P.-Y., Yi, J., Zhang, H., and Hsieh, C.-J. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

Cheng, M., Lei, Q., Chen, P.-Y., Dhillon, I., and Hsieh, C.-J. Cat: Customized adversarial training for improved robustness. *arXiv preprint arXiv:2002.06789*, 2020a.

Cheng, M., Singh, S., Chen, P., Chen, P.-Y., Liu, S., and Hsieh, C.-J. Sign-opt: A query-efficient hard-label adversarial attack. In *ICLR*, 2020b.

Chor, B., Fiat, A., and Naor, M. Tracing traitors. In *Crypto*, volume 94, pp. 257–270, 1994.

Chor, B., Fiat, A., Naor, M., and Pinkas, B. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. *International Conference on Machine Learning*, 2019.

Darvish Rouhani, B., Chen, H., and Koushanfar, F. Deepsigns: An end-to-end watermarking framework for ownership protection of deep neural networks. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 485–497, 2019.

Fawzi, A., Moosavi-Dezfooli, S.-M., and Frossard, P. Robustness of classifiers: from adversarial to random noise. *Advances in neural information processing systems*, 29, 2016.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

Guo, C., Gardner, J., You, Y., Wilson, A. G., and Weinberger, K. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pp. 2484–2493. PMLR, 2019.

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

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018a.

Ilyas, A., Engstrom, L., and Madry, A. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018b.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *International Conference on Learning Representations*, 2017.

Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Le Merrer, E., Perez, P., and Trédan, G. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.

Liu, S., Chen, P.-Y., Chen, X., and Hong, M. signsgd via zeroth-order oracle. In *International Conference on Learning Representations*, 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

Nicolae, M.-I., Sinn, M., Tran, M. N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32: 323–332, 2012.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014.

Tu, C.-C., Ting, P., Chen, P.-Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.-J., and Cheng, S.-M. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 742–749, 2019.

Uchida, Y., Nagai, Y., Sakazawa, S., and Satoh, S. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 269–277, 2017.

Wang, T. and Kerschbaum, F. Attacks on digital watermarks for deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2622–2626. IEEE, 2019.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. *International Conference on Machine Learning*, 2019.

Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., and Molloy, I. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 159–172, 2018.

A. Scalability

In Section 4.1, we propose an efficient training framework for model copies by splitting the entire model and only fine-tuning the head portion. To evaluate its effectiveness, we compare the computational time required to train a base model from scratch versus fine-tuning a model head with a frozen model tail in Table 3. The experimental results demonstrate that by fine-tuning the model head, our method could accelerate the overall training speed by over 4 times, which demonstrates our proposed framework are able to achieve a good Scalability.

Table 3. Comparison of computational time (seconds) between training a base model and fine-tuning a model head.

Dataset	Model	Training Base Model	Fine-tuning Model Head
CIFAR-10	ResNet18	596.79	108.90
	VGG16	841.64	155.25
GTSRB	ResNet18	1193.67	278.13
	VGG16	496.45	96.99
Tiny-ImageNet	ResNet18	4245.16	819.84
	VGG16	2307.29	454.49

B. Sensitivity Analysis

B.1. Results on different watermark sizes

We explore the potential impact of the watermark size by selecting various numbers of masked pixels. We conduct experiments with ResNet18 on CIFAR-10 and our results are demonstrated in Table 4. We find that as the watermark size increases, the tracing accuracy of NES and SimBA gets improved, while other attacks are not heavily affected by the watermark size. Although utilizing a larger watermark may help slightly enhance the tracing accuracy, they would mask off more pixels which degrade the model performance.

Table 4. Tracing accuracy (%) with different watermark sizes. The experimental results are conducted with ResNet18 on CIFAR-10.

Case	Watermark Size	Bandit	HSJ	NES	SignOPT	SimBA
Data-Limited	50	52	94	82	73	68
	100	51	97	91	68	79
	200	52	97	91	60	97
	400	51	99	90	58	96
	800	49	98	92	60	100
Data-Free	50	65	88	77	92	55
	100	71	85	76	82	59
	200	72	88	80	86	63
	400	70	84	79	89	66
	800	74	87	77	86	63

B.2. Results on different split strategy on the model head

We further analyze the split strategy on the model head and conduct our experiments with ResNet18 on CIFAR-10. Specifically, in our original setting, we split the model by taking the first several layers, including one ResNet block as the head and the rest as the tail. We then vary the split strategy by moving the ResNet block from the tail to the head each time. The tracing accuracies (%) on 100 model copies are shown in Table 5. We observe that utilizing a large model part as the head may slightly increase the overall tracing accuracy. However, a large head part would also increase the overall fine-tuning time and there exists a trade-off between the watermarking efficiency and the tracing performance. As shown in Table 5, we find that including only one ResNet block as the head is enough to achieve satisfactory tracing accuracy.

Table 5. Tracing accuracy (%) with different model heads. The experimental results are conducted with ResNet18 on CIFAR-10.

CASE	NUMBER OF BLOCKS	BANDIT	HSJ	NES	SIGNOPT	SIMBA
DATA-LIMITED	1	51	97	91	68	79
	2	56	94	86	58	77
	3	61	95	90	71	77
	4	59	96	90	75	90
DATA-FREE	1	71	85	76	82	59
	2	75	85	76	82	59
	3	71	89	81	90	56
	4	78	89	86	91	67