Identification of the Adversary from a Single Adversarial Example

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks have been shown vulnerable to adversarial examples. Even though many defence 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 in a forensic investigation manner. Specifically, we focus on the following setting: the machine learning service provider provides models for a set of customers. However, one of the customers conducted adversarial attacks to fool the system. The investigator's objective is then to identify the first compromised model 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 models. 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.

1 Introduction

It has been shown recently that machine learning algorithms, especially deep neural networks, are vulnerable to adversarial attacks [1, 2]. To enhance the robustness against attacks, many defence strategies have been proposed [3, 4, 5]. 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 for generating the adversarial attack. In this paper, we show that given only a **single** adversarial example, we could trace the source model that adversaries based for conducting the attack. We consider the following setting: a Machine Learning as a Service (MLaaS) provider will provide models for a set of customers. For the consideration of time-sensitive applications such as auto-pilot systems, the models would be distributed to every customer locally. The model architecture and weight details are encrypted and hidden from the customers 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 model but not the internal configurations. On the other side, the service provider has full access to every detail of their models, 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 the models are trained for the same objective using the same dataset, adversarial examples generated by the adversary

Submitted to the ML Safety Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute

could be transferred to the other users' models with a very high probability, 100% if the models 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. To make the tracing possible, we design a random mask watermarking strategy which embeds the watermark to the generated adversarial samples without sacrificing model 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's masked pattern and the adversarial example's output distribution among different models. 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 for the investigator. Observing that the model'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 model. 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 to minimize. 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 by only using a single sample and its corresponding adversarial example.

2 Related Work

Adversarial Attack Since the discovery of adversarial example [1], 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 [2, 6, 3, 7]. 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. Some famous attacks in the soft-label settings are ZOO attack [8], NES [9], Bandit [10], SimBA [11]. In the hard-label setting, boundary attack [12] and HSJ [13] use random walk based method while OPT attack [14] and Sign-OPT attack [15] formalized the hard-label attack into an optimization framework and used the zeroth-order method to solve it.

Forensic investigation in Machine Learning Although machine learning methods have already been used in forensic science [16], 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 Methodology

We formalize the identification of the compromised model in the owner-customer distribution setting [17]. The machine learning service provider (owner) is assumed to own m copies of model $f_1, f_2, \ldots, f_m$ for the same K-way classification task trained using the same training dataset. As inference efficiency is critical in time-sensitive applications such as auto-pilot systems, these model copies are first encrypted for intellectual protection and security concerns and then distributed to the m customers (users). Therefore, the customers only have black-box access to their own distributed model. In other words, the user i could only query his own model f_i to get the prediction results without any access to the internal information about the model. Unfortunately, a malicious user (adversary) exists who aims to fool the whole system, including other users' models, by conducting black-box adversarial attacks. Specifically, let the malicious user's model copy to be f_{att} (the *compromised model*). As he does not have access to query other users' models, he 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' models. Our task is to find the compromised model f_{att} from the model pool.

We then propose our framework which consists of two parts shown in Figure 1 in Appendix. First, we design a simple random mask watermarking method that would have a limited effect on the models' accuracy while embedding distinctive features in adversarial examples, distinguishing them from those generated from other models. We then propose two detection scenarios to identify the adversary from adversarial examples.

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 1, for each model 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 model 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 model 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, 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 model 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 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 with a few network layers as 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 $f_i(\boldsymbol{x}) = t(h_i(\boldsymbol{x}))$. 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 with a unique model 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 the model by only fine-tuning the weights in the head part with a few epochs. Our experiments in Appendix will show it is sufficient to embed watermark to a few layers in DNNs without sacrificing model accuracy.

Data-limited adversary identification With the watermarking scheme described in Section 3, we can exploit the information embedded in the watermarked adversarial example (and the corresponding original example) to identify the compromised model.

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. 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.

Formally, for the model f_i, the gradient of the designed loss function $\mathcal{L}$ with respect to the given sample $\boldsymbol{x}$ is $\frac{\partial \mathcal{L}(f_i(\tilde{\boldsymbol{x}}))}{\partial \boldsymbol{x}_{a,b,c}} = 0$ if $\boldsymbol{w}^i_{a,b,c} = 1$. 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. 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 model copy f_{att}, values in $\boldsymbol{\delta}$ should be much smaller in those coordinates where $\boldsymbol{w}^{att} = 1$. Therefore, given $\boldsymbol{x}_{adv}$ and $\boldsymbol{x}$, we thus calculate a score for each model by summing up the absolute values of the adversarial perturbation overall masked pixels (of the corresponding model), i.e., $\boldsymbol{\delta}^i = \sum_{a,b,c} \boldsymbol{w}^i_{a,b,c} \odot |\boldsymbol{x}_{adv} - \boldsymbol{x}|_{a,b,c}$. 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 models. Since adversarial examples should be identical to original examples visually, the ground-truth label could be easily inferred. Specifically, if the adversarial example $\boldsymbol{x}_{adv}$ is generated from model 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 $\boldsymbol{x}$. Intuitively, model f_i would have the smallest confidence on the ground-truth label since some of the adversarial perturbation may be blocked by other models' watermarks. We then combine the two metrics and calculate the final score for each model. Then, we take the model with the smallest score as the compromised model, i.e.

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

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 model outputs when applying different masks to the adversarial example. Formally, for the given adversarial example $\boldsymbol{x}_{adv}$, we first apply every model's watermark $\boldsymbol{w}^i, i \in [m]$ to create a set of masked adversarial examples $\{\tilde{\boldsymbol{x}}^i_{adv}\}^m_{i=1}$ where $\tilde{\boldsymbol{x}}^i_{adv} = \boldsymbol{x}_{adv} \odot (1 - \boldsymbol{w}^i)$. We then feed the masked adversarial examples set to each model f_i to get its probability output. For every model 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 [12, 14], 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 model 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 model, the extent of change would be limited. Therefore, we propose the max label score S_{max} based on the extent of change of prediction:

$$S^i_{max} = \frac{\max_{1 \leq k \leq K} \boldsymbol{P}^i_{i,k}}{\sum_{1 \leq j \leq m} \max_{1 \leq k \leq K} \boldsymbol{P}^i_{j,k}} \tag{2}$$

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

$$att \leftarrow \underset{i}{\operatorname{argmin}}(S^i_{max} + \beta \mathcal{L}_{CE}(f_i(\boldsymbol{x}_{adv}), \boldsymbol{y})) \tag{3}$$

4 Experimental Results

Implementation Details: We conduct our experiments on two popular image classification datasets GTSRB [18] and CIFAR-10 [19]. We use two widely used network architectures VGG16 [20] and ResNet18 [21]. We perform the following five black-box adversarial attacks (NES [9], Bandit [10], SimBA [11], HSJ [13], SignOPT [15]) to generate the adversarial example. For all attacks, we use Adversarial Robustness Toolbox (ART) [22]'s implementation. We use the default hyperparameters in the ART toolbox to conduct the attack. 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 model f_{att} is wrong and, at the same time, the prediction of at least one of the other $m-1$ models is wrong.

To sum up, for each attack, we have a total of 1000 adversarial examples under the setting of 100 models. We then define the tracing accuracy to evaluate the detection rate defined as Trace Acc $= \frac{N_{\text{correct}}}{N_{\text{total}}}$ where N_{correct} is the count of the correct identification of the compromised model and N_{total} is the total number the transferable adversarial example generated.

Identification Results For identification in the data-limited setting, we conduct experiments on 100 copies of models applied with random masks. We set the hyperparameter α to 0.85 for CIFAR10 and 0.5 for GTSRB and test tracing accuracy on different attacks. The results in the top half of Table 1 illustrate that our detection method could identify the compromised model successfully in all datasets and network architectures which achieves an average of 75.2% tracing accuracy with only one adversarial example. 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 1. For the data-free case, we also set the hyperparameter β to 0.5 for both datasets. Although the original example is no longer available, we could still achieve a similar or even better tracing accuracy against some attacks.

Table 1: 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-Limit	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	75.32
Data-Free	V-CIFAR10	66.2	83.9	71.6	85.7	59.2	73.32
	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

We also apply the adaptive attack and multiple adversarial example experiments to further verify the proposed methods' effectiveness in Appendix.

5 Conclusion

In this paper, we develop the first framework for identifying the compromised model from a single adversarial example for the forensic investigation. We first present a watermarking method to make the generated adversarial example unique and differentiable. 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 model by only giving a single adversarial example. Our framework could further improve the detection rate to near 100% when two more adversarial examples are provided.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

[3] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[4] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. *International Conference on Machine Learning*, 2019.

[5] Minhao Cheng, Qi Lei, Pin-Yu Chen, Inderjit Dhillon, and Cho-Jui Hsieh. Cat: Customized adversarial training for improved robustness. *arXiv preprint arXiv:2002.06789*, 2020.

[6] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *International Conference on Learning Representations*, 2017.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, pages 39–57, 2017.

[8] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 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*, pages 15–26, 2017.

[9] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

[10] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. *arXiv preprint arXiv:1807.07978*, 2018.

[11] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019.

[12] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[13] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[14] Minhao Cheng, Thong Le, Pin-Yu Chen, Jinfeng Yi, Huan Zhang, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. *arXiv preprint arXiv:1807.04457*, 2018.

[15] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. In *ICLR*, 2020.

[16] Alicia Carriquiry, Heike Hofmann, Xiao Hui Tai, and Susan VanderPlas. Machine learning in forensic applications. *Significance*, 16(2):29–35, 2019.

[17] Jiyi Zhang, Wesley Joon-Wie Tann, and Ee-Chien Chang. Mitigating adversarial attacks by distributing different copies to different users. *arXiv preprint arXiv:2111.15160*, 2021.

[18] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

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

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[21] 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.

[22] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

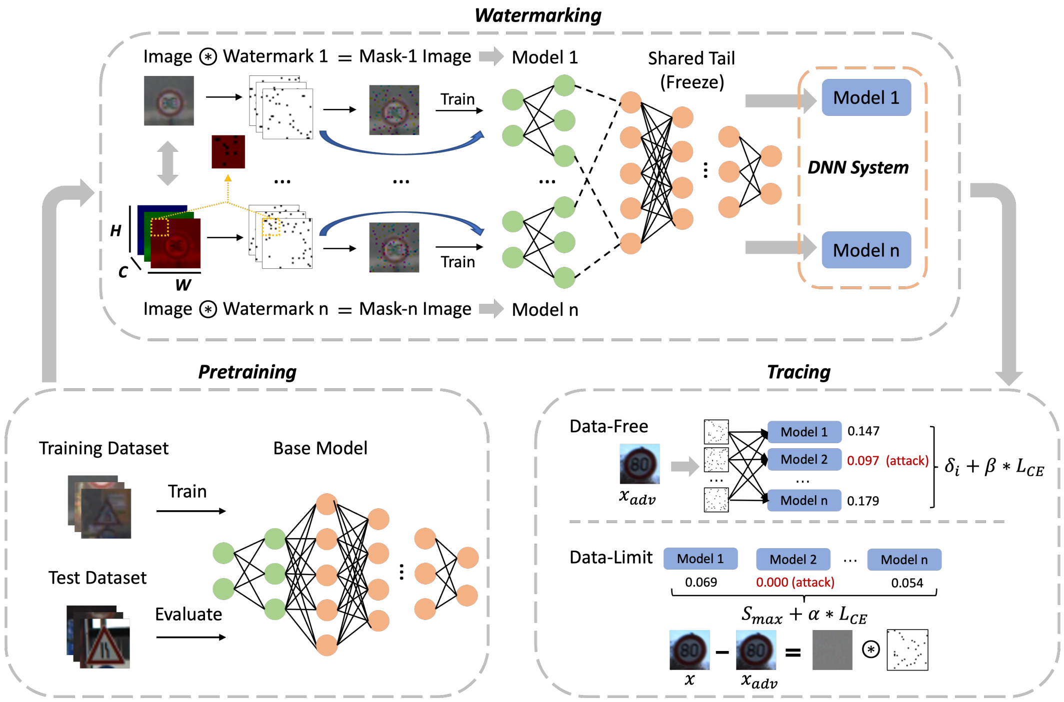

Figure 1: Proposed framework of identifying compromised model from adversarial examples.

A Appendix

A.1 Model performance with random mask watermarking

In this section, we conduct experiments to verify whether the model could still maintain a good performance after applying the watermark. Specifically, we train 100 models on two datasets CIFAR-10 and GTSRB with two popular architectures VGG16 and ResNet18. We also add a baseline model without watermark as a reference.

Table 2: The classification accuracies (%) of models with random mask watermarking. V-CIFAR10 represents the model trained with VGG16 using the CIFAR-10 dataset and R-GTSRB represents the ResNet18 model trained using the GTSRB dataset.

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

In Table 2, it could be clearly observed that the accuracy of the watermarked models 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 model performance.

A.2 Results on adaptive attack

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. The attacker then adds some Gaussian noise within the watermark to fool our tracing method. We test the average tracing accuracy across different noise levels on CIFAR10 with ResNet18 structure. Our results are shown in Table 2.

Not surprisingly, 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.

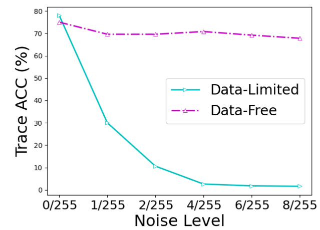

Figure 2: Average tracing accuracy on adaptive attack with different random noise levels.

A.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 3 and Section 3 for each example, and then add up each score computed over all adversarial examples. Then we take the model with the smallest sum as the compromised model. We then 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 3 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.

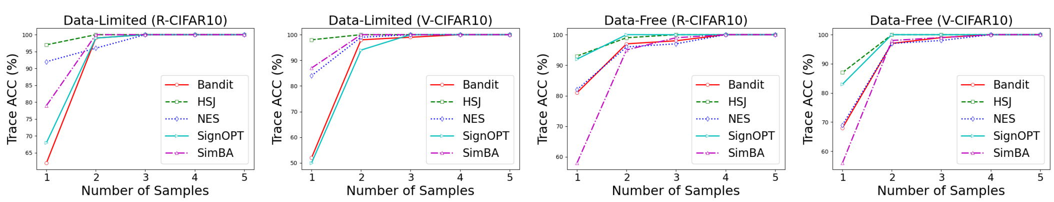

Figure 3: Tracing accuracy with different numbers of adversarial examples.

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.