Defending against Model Stealing via Verifying Embedded External Features

Yiming Li[1,*], **Linghui Zhu**[1,2,*], **Xiaojun Jia**[3], **Yong Jiang**[1,2], **Shu-Tao Xia**[1,2,†], **Xiaochun Cao**[3]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2]Peng Cheng Laboratory, Shenzhen, China
[3]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
li-ym18@mails.tsinghua.edu.cn; xiast@sz.tsinghua.edu.cn

Abstract

Obtaining a well-trained model involves expensive data collection and training procedures, therefore the model is a valuable intellectual property. Recent studies revealed that adversaries can 'steal' deployed models even when they have no training samples and can not get access to the model parameters or structures. Currently, there were some defense methods to alleviate this threat, mostly by increasing the cost of model stealing. In this paper, we explore the defense from another angle by verifying whether a suspicious model contains the knowledge of defender-specified *external features*. Specifically, we embed the external features by tempering a few training samples with style transfer. We then train a meta-classifier to determine whether a model is stolen from the victim. This approach is inspired by the understanding that the stolen models should contain the knowledge of features learned by the victim model. We examine our method on both CIFAR-10 and ImageNet datasets. Experimental results demonstrate that our method is effective in detecting different types of model stealing simultaneously, even if the stolen model is obtained via a multi-stage stealing process. The codes for reproducing main results are available at Github (https://github.com/zlh-thu/StealingVerification).

1 Introduction

Deep learning, especially deep neural networks (DNNs), has demonstrated its great power in many applications (Guo et al. 2020; Stokes et al. 2020; Minaee et al. 2021). In general, training a well-performed model requires a large number of training samples and a massive amount of computational resources. Both data collection and training process are expensive and time-consuming, which makes the trained model a valuable intellectual property to its owner.

Recently, researchers found that adversaries can 'steal' the (deployed) *victim model* even when they have no training samples and can not get access to the model parameters or structures (Tramèr et al. 2016; Orekondy, Schiele, and Fritz 2019; Chandrasekaran et al. 2020). For example, the adversaries may use the victim model to label an unlabeled dataset, based on which to train the *stolen model*. This threat is called *model stealing*. Since the model stealing can obtain a function-similar copy of the victim model stealthily, it poses a huge threat to model owners.

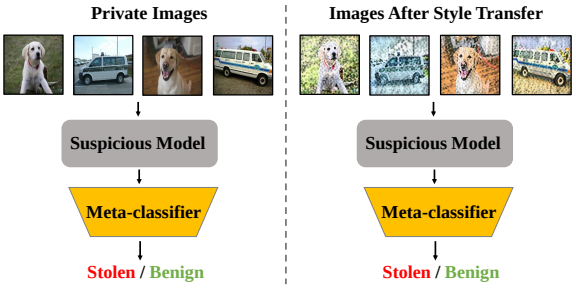


Figure 1: The verification stage of dataset inference and our method. Specifically, dataset inference adopts inherent features contained in benign private images while our method utilizes external features embedded in stylized images.

To alleviate the threat of model stealing, there were also some defense methods, mostly by introducing randomness or perturbation in the victim models to increase the costs of model stealing (Tramèr et al. 2016; Lee et al. 2019; Kariyappa and Qureshi 2020). For instance, defenders may perturb the prediction by rounding or adding noise to the posterior probabilities. However, these defenses may significantly reduce the performance of legitimate users and could even be bypassed by following adaptive attacks (Jia et al. 2021; Maini, Yaghini, and Papernot 2021).

In this paper, we explore the defense of model stealing from another perspective by *verifying whether a suspicious model has defender-specified behaviors*. If the model has such behaviors, we treat it as stolen from the victim. This approach is inspired by the understanding that the stolen models should contain the knowledge of features learned by the victim model and therefore they have similar behaviors. To the best of our knowledge, there is only one work, *i.e.*, *dataset inference* (Maini, Yaghini, and Papernot 2021), focusing on this perspective, where they adopted *inherent features* of the training set to verify model ownership. However, we reveal that this approach is easy to make misjudgments, especially when the training set of suspicious models have a similar distribution to that of the victim model. The misjudgment is most probably because different models may learn similar inherent features once their training sets have certain

*The first two authors contributed equally to this work.
†Corresponding Author: Shu-Tao Xia

similarities. Based on this understanding, we propose to embed defender-specified *external features* into victim models for ownership verification. These features are different from those contained in the original training set. Specifically, we embed external features by tempering the images of some training samples with *style transfer*. Since we only poison a few samples and do not change their labels, the embedded features will not hinder the functionality of the victim model. Besides, we also train a *benign model* based on the original training set. It is used only for training a *meta classifier* to determine whether a suspicious model is stolen from the victim model. Only the model containing the knowledge of external features will be deployed.

The main contribution of this work is four-fold: **(1)** We revisit the defense of model stealing from the aspect of ownership verification. **(2)** We reveal the limitations of existing verification-based methods. Based on these understandings, we propose a simple yet effective defense approach. **(3)** We verify the effectiveness of our method on benchmark datasets under various types of model stealing simultaneously. **(4)** Our work could provide a new angle about how to adopt 'data poisoning' for positive purposes.

2 Background and Related Work

In this paper, we focus on model stealing and its defenses towards image classification. Other tasks are out the scope of this paper. We will discuss them in our future work.

2.1 Model Stealing

Model stealing aims to steal the intellectual property from a victim by obtaining a function-similar copy of the deployed model. In general, existing methods can be divided into three categories based on adversary's permission level, as follows:

Dataset-Accessible Attacks ($\mathcal{A}_D$): In this setting, the adversary can get access to the training dataset whereas can only query the model. In this case, the adversary can train a substitute model based on knowledge distillation (Hinton, Vinyals, and Dean 2014) for model stealing.

Model-Accessible Attacks ($\mathcal{A}_M$): In this setting, the adversary has complete access to the victim model. This type of attack could happen when the victim model is open-sourced or via insider access. In this case, the adversary can obtain a substitute model by using data-free knowledge distillation based on zero-shot learning (Fang et al. 2019) or simply by fine-tuning the victim model with local training samples.

Query-Only Attacks ($\mathcal{A}_Q$): In this setting, the adversary can only query the model. Query-only attacks can also be divided into two subclasses, including the *label-query attacks* (Papernot et al. 2017; Jagielski et al. 2020; Chandrasekaran et al. 2020) and *logit-query attacks* (Tramèr et al. 2016; Orekondy, Schiele, and Fritz 2019), based on model's feedback. In general, label-query attacks adopted the victim model to annotate some substitute (unlabeled) samples, based on which to train their substitute model. In the logit-query attacks, the adversary usually obtains the substitute model by minimizing the distance between its predicted logits and those generated by the victim model.

2.2 Defenses against Model Stealing

Non-verification based Defenses. Currently, most of the existing methods alleviated the stealing threat by increasing the cost of model stealing through perturbing model results. For instance, defenders could round the probabilities (Tramèr et al. 2016), added noise to the prediction that results in a high loss (Lee et al. 2019), only returning the most confident label (Orekondy, Schiele, and Fritz 2019). However, these defenses may significantly reduce the performance of legitimate users and could even be bypassed by adaptive attacks (Jia et al. 2021; Maini, Yaghini, and Papernot 2021). Other works (Kesarwani et al. 2018; Juuti et al. 2019; Yan et al. 2021) detected model stealing by identifying malicious queries. However, these methods relied on some assumptions of malicious query patterns, which may not be adopted by the adversaries in practice.

Dataset Inference. To the best of our knowledge, this is the first and currently the only verification-based defense against model stealing. Its key idea is to identify whether a suspicious model contains the knowledge of the inherent features that the victim model V learned from the private training set. Specifically, let we consider a K-classification problem. For each sample $(\boldsymbol{x}, y)$, dataset inference first generated its minimum distance $\boldsymbol{\delta}_t$ to each class t by

$$\min_{\boldsymbol{\delta}_t} d(\boldsymbol{x}, \boldsymbol{x}+\boldsymbol{\delta}_t), s.t., V(\boldsymbol{x}+\boldsymbol{\delta}_t) = t, \tag{1}$$

where $d(\cdot)$ is a distance metric ($e.g.$, ℓ^∞ norm). The distance to each class $\boldsymbol{\delta} = (\boldsymbol{\delta}_1, \cdots, \boldsymbol{\delta}_K)$ is the feature embedding of sample $(\boldsymbol{x}, y)$ $w.r.t.$ the victim model V. After that, the defender will randomly select some samples inside (labeled as '+1') or out-side (labeled as '-1') their private dataset and use the feature embedding $\boldsymbol{\delta}$ to train a binary meta-classifier C, where $C(\boldsymbol{\delta}) \in [0, 1]$ indicates the probability that the sample $(\boldsymbol{x}, y)$ is from the private set. To determine whether a suspicious model is stolen from the victim, the defender creates equal-sized sample vectors from private and public samples and conduct the *hypothesis test* based on the trained C. If the confidence scores of private samples are significantly greater than those of public samples, the suspicious model is treated as stolen from the victim. However, as shown in following experiments, this method is easy to make misjudgments, especially when the training set of suspicious models have a similar distribution to that of the victim model. This limitation hinders its utility in practice.

Model Watermarking. The main purpose of model watermarking is detecting theft ($i.e.$, directly copy the model) instead of preventing model stealing. However, we notice that the dataset inference enjoys certain similarities to the *misclassification-based model watermarking* (Adi et al. 2018; Jia et al. 2021; Szyller et al. 2021), especially the backdoor-based ones (Adi et al. 2018; Zhang et al. 2018; Li et al. 2020b). As such, these approaches could be potential defenses against model stealing. Specifically, these methods performed ownership verification by making the protected model misclassifying defender-specified samples. For example, defenders may first adopt *backdoor attacks* (Gu et al. 2019; Li et al. 2020a; Nguyen and Tran 2020;

Li et al. 2021b; Bagdasaryan and Shmatikov 2021; Li et al. 2021a) to watermark the model during the training process and then conduct the ownership verification. In general, a backdoor attack can be characterized by three components, including the trigger pattern $\boldsymbol{t}$, target class y_t, and adversary-predefined poisoned image generator $G(\cdot)$. Given the benign training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, the backdoor adversary will randomly select $\gamma\%$ samples (*i.e.*, $\mathcal{D}_s$) from $\mathcal{D}$ to generate their poisoned version $\mathcal{D}_p = \{(\boldsymbol{x}', y_t)|\boldsymbol{x}' = G(\boldsymbol{x}; \boldsymbol{t}), (\boldsymbol{x}, y) \in \mathcal{D}_s\}$. Different backdoor attacks may assign different generator $G(\cdot)$. For example, $G(\boldsymbol{x}; \boldsymbol{t}) = (\boldsymbol{1} - \boldsymbol{\lambda}) \otimes \boldsymbol{x} + \boldsymbol{\lambda} \otimes \boldsymbol{t}$ where $\boldsymbol{\lambda} \in \{0, 1\}^{C \times W \times H}$ and $\otimes$ indicates the element-wise product in the BadNets (Gu et al. 2019). After $\mathcal{D}_p$ was generated, $\mathcal{D}_p$ and remaining benign samples $\mathcal{D}_b \triangleq \mathcal{D} \backslash \mathcal{D}_s$ will be used to train the model f_θ via

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_p \cup \mathcal{D}_b} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y). \tag{2}$$

In the verification stage, the defender will examine suspicious models in predicting y_t. If the confidence scores of poisoned samples are significantly greater than those of benign samples, the suspicious model is treated as watermarked and therefore it is stolen from the victim. However, as shown in the following experiments, these methods have far less effective in defending against model stealing.

3 Revisiting Verification-based Defenses

3.1 The Limitation of Dataset Inference

As illustrated in Section 2.2, dataset inference relied on a latent assumption that a model will not learn the features contained in the private dataset if it is not stolen from the victim. However, since different models may learn similar features even they are trained on different datasets, this assumption does not hold and therefore the method may misjudge. In this section, we verify this limitation.

Settings. In this section, we conduct the experiments on CIFAR-10 (Krizhevsky, Hinton et al. 2009) dataset with VGG (Simonyan and Zisserman 2015) and ResNet (He et al. 2016). Specifically, we randomly separate the original training set $\mathcal{D}$ into two disjoint subsets $\mathcal{D}_l$ and $\mathcal{D}_r$. We train the VGG on $\mathcal{D}_l$ (dubbed VGG-$\mathcal{D}_l$) and the ResNet on $\mathcal{D}_r$ (dubbed ResNet-$\mathcal{D}_r$), respectively. We also train the VGG on a noisy dataset $\mathcal{D}_l' \triangleq \{(\boldsymbol{x}', y)|\boldsymbol{x}' = \boldsymbol{x} + \mathcal{N}(0, 16), (\boldsymbol{x}, y) \in \mathcal{D}_l\}$ (dubbed VGG-$\mathcal{D}_l'$) for reference. In the verification process, we verify whether the VGG-$\mathcal{D}_l$ and VGG-$\mathcal{D}_l'$ is stolen from ResNet-$\mathcal{D}_r$ and whether the ResNet-$\mathcal{D}_r$ is stolen from VGG-$\mathcal{D}_l$ based on the settings proposed in dataset inference (Maini, Yaghini, and Papernot 2021). Besides, we also adopt the p-value as the evaluation metric. The p-value is calculated based on the approach described in Section 2.2. Note that *the smaller the p-value, the more confident that dataset inference believes the model stealing happened*. More detailed settings are in **Appendix**.

Results. As shown in Table 1, all models achieve decent performance even when the training samples are limited. However, the p-value is significantly smaller than 0.01 in all cases. In other words, the dataset inference believes that these models are stolen from the victim in a high confidence. However, in each case, since the suspicious and the victim model are trained on completely different training samples and with different model structures, the suspicious model should not be considered as stolen from the victim. These results reveal that *the dataset inference could make misjudgments and therefore its results are questionable*. In particular, the p-value of VGG-$\mathcal{D}_l$ is smaller than that of the VGG-$\mathcal{D}_l'$. This is probably because the latent distribution of $\mathcal{D}_l'$ is more different from that of $\mathcal{D}_r$ (compared with that of $\mathcal{D}_l$) and therefore models learn more different features.

	ResNet-$\mathcal{D}_r$	VGG-$\mathcal{D}_l$	VGG-$\mathcal{D}_l'$
Accuracy	88.0%	87.7%	85.0%
p-value	$\mathbf{10^{-7}}$	$\mathbf{10^{-5}}$	$\mathbf{10^{-4}}$

Table 1: The accuracy of victim models and p-value of verification processes. Dataset inference misjudged in all cases.

Model Type $\rightarrow$	Benign	Watermarked	Stolen
BA	91.99	90.03	70.17
ASR	0.01	98.02	**3.84**

Table 2: The performance (%) of different models.

3.2 The Limitation of Model Watermarking

Intuitively, the inference process of backdoor attacks is similar to unlocking a door with the corresponding key. As such, the success of backdoor-based model watermarking relied on an assumption that the trigger pattern matches hidden backdoors contained in the stolen model. This assumption holds in its originally discussed scenarios where the stolen model is the same as the victim model. However, it may not hold in the model stealing, since the backdoors contained in the stolen models may be changed or even removed during the stealing process. Accordingly, backdoor-based model watermarking may fail in defending against model stealing. In this section, we verify this limitation.

Settings. In this part, we adopt the most representative and effective backdoor attack, the BadNets (Gu et al. 2019), as an example for the discussion. The watermarked model will then be stolen by the data-free distillation-based model stealing (Fang et al. 2019). We adopt the *benign accuracy (BA)* and *attack success rate (ASR)* (Li et al. 2020a) to evaluate the performance of the stolen model. The larger the ASR, the more likely the stealing will be detected. More detailed settings can be found in **Appendix**.

Results. As shown in Table 2, the ASR of the stolen model is only 3.84%, which is significantly lower than that of the watermarked model. In other words, *the defender-specified trigger no longer matches the hidden backdoors contained in the stolen model*. As such, backdoor-based model watermarking will fail to detect model stealing.

To further understand the reason of this failure, we synthesize the potential trigger pattern of each model based on the targeted universal adversarial attack (Moosavi-Dezfooli et al. 2017). As shown in Figure 3, the pattern recovered from the watermarked model is similar to the ground-truth

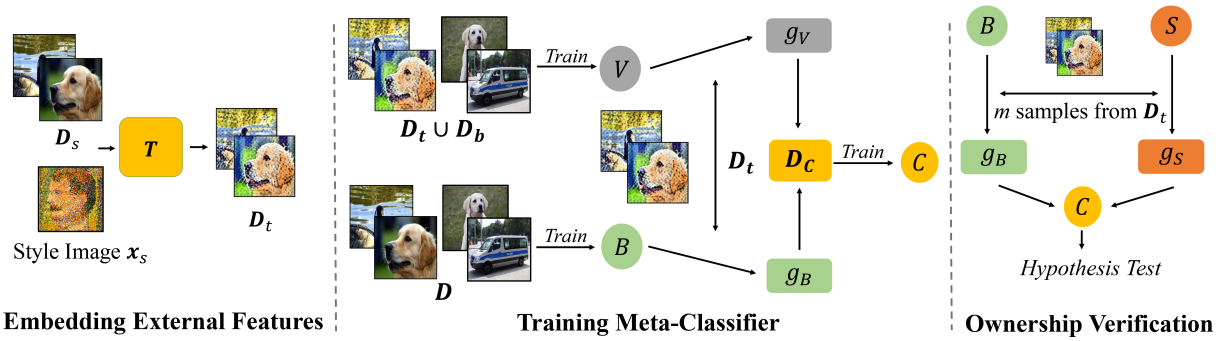
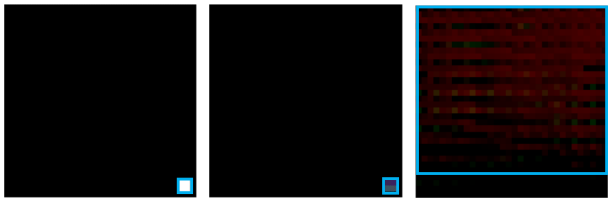


Figure 2: The main pipeline of our method. In the first stage, defenders will modify some images via style transfer for embedding external features. In the second stage, defenders will train a meta-classifier to determine whether a suspicious model is stolen from the victim based on gradients. In the last stage, defenders will conduct ownership verification via hypothesis test.



(a) (b) (c)

Figure 3: The adopted trigger pattern and synthesized ones obtained from the watermarked and stolen model. The trigger areas are indicated in the blue box. **(a)** ground-truth trigger pattern; **(b)** pattern obtained from the watermarked model; **(c)** pattern obtained from the stolen model.

one, whereas the one recovered from the stolen model is completely different from the ground-truth pattern. These results explain why backdoor-based model watermarking has minor effects in defending against model stealing.

Moreover, *backdoor-based model watermarking will introduce new security threats*, since it builds a stealthy latent connection between trigger pattern and target label. The adversary may use it to maliciously manipulate the predictions of deployed models. This problem will also hinder the utility of backdoor-based model watermarking in practice.

4 The Proposed Method

Based on the understandings in Section 3, in this paper, we propose to embed *external features* instead of inherent features for ownership verification. Specifically, as shown in Figure 2, our method consists of three main stages, including **(1)** embedding external features, **(2)** training an ownership meta-classifier, and **(3)** conducting ownership verification. Their technical details are in the following subsections.

4.1 Threat Model

Following the setting of existing works (Zhang et al. 2020; Wang and Kerschbaum 2021; Liu, Weng, and Zhu 2021), we conduct the defense in a *white-box* setting, where the defender has complete access to the suspicious model. However, the defender has no information about the stealing process. The goal of defenders is to accurately identify whether the suspicious model is stolen from a victim model, based on behaviors of the suspicious and victim model.

One may argue that only black-box defenses are practical since the adversary may refuse to provide the suspicious model. However, white-box defenses are also practical. In our understanding, the real-world adoption of verification-based defenses (in a legal system) requires an official institute for the arbitration. Specifically, all commercial models should be registered here, through the unique identification ($e.g.$, MD5 code) of their model's weights file. When this official institute is established, its staff should take responsibility for the verification process. For example, the staff can require the company to provide the model file with the same registered identification and then use our method (under the white-box setting) for the ownership verification.

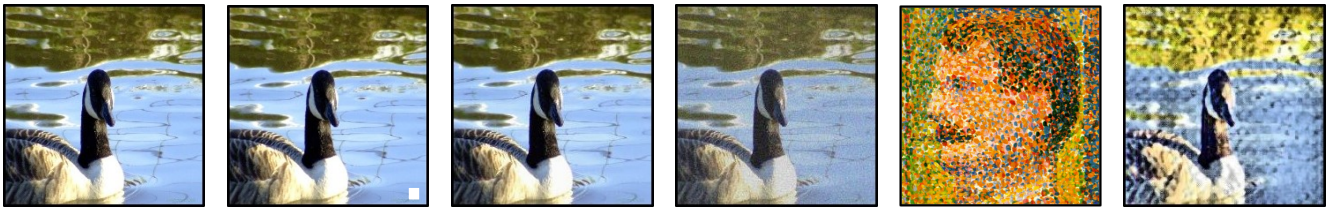
4.2 Embedding External Features

In this section, we describe how to embed external features into the victim model. Before we reach technical details, we first present the definition of inherent and external features.

Definition 1. *A feature f is called the inherent feature (of dataset $\mathcal{D}$) if and only if $\forall(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}, (\boldsymbol{x}, y) \in \mathcal{D} \Rightarrow (\boldsymbol{x}, y)$ contains feature f. Similarly, f is called the external feature (of dataset $\mathcal{D}$) if and only if $\forall(\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}, (\boldsymbol{x}, y)$ contains feature $f \Rightarrow (\boldsymbol{x}, y) \notin \mathcal{D}$.*

Although external features are well defined, how to construct them is still difficult since the learning dynamic of DNNs remains unclear and the concept of features itself is complicated. However, at least we know that the *image style* can serve as a feature for the learning of DNNs in image-related tasks, based on some recent studies (Geirhos et al. 2019; Duan et al. 2020; Cheng et al. 2021). As such, we can use *style transfer* (Johnson, Alahi, and Fei-Fei 2016; Huang and Belongie 2017; Chen et al. 2020) for embedding external features. People may also adopt other methods for the embedding. It will be discussed in our future work.

Specifically, let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ denotes the benign training set, $\boldsymbol{x}_s$ is a defender-specified *style image*, and $T : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ is a (trained) style transformer. In this stage, the defender first randomly selects $\gamma\%$ (dubbed *transformation rate*) samples ($i.e.$, $\mathcal{D}_s$) from $\mathcal{D}$ to generate their transformed version $\mathcal{D}_t = \{(\boldsymbol{x}', y) | \boldsymbol{x}' = T(\boldsymbol{x}, \boldsymbol{x}_s), (\boldsymbol{x}, y) \in \mathcal{D}_s\}$. The



(a) (b) (c) (d) (e) (f)

Figure 4: Images involved in different defenses. **(a)** benign image; **(b)** poisoned image in BadNets; **(c)** poisoned image in Gradient Matching; **(d)** poisoned image in Entangled Watermarks; **(e)** style image; **(f)** transformed image.

external features (contained in $\boldsymbol{x}_s$) will be learned by the victim model V_θ during the training process via

$$\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},y)\in\mathcal{D}_b\cup\mathcal{D}_t} \mathcal{L}(V_{\boldsymbol{\theta}}(\boldsymbol{x}), y), \tag{3}$$

where $\mathcal{D}_b \triangleq \mathcal{D}\backslash\mathcal{D}_s$ and $\mathcal{L}(\cdot)$ is the loss function.

In this stage, how to select the style image is an important question. Intuitively, it should be significantly different from those contained in the original training set. In practice, defenders can simply adopt oil or sketch paintings as the style image since most of the images that need to be protected are natural images. We will further discuss it in Section 5.3.

In particular, since we only poison a few samples and do not change their labels, the embedding of external features will not hinder the functionality of victim models or introduce new security risks ($e.g.$, hidden backdoors).

4.3 Training Ownership Meta-Classifier

Since there is no explicit expression of the embedded external features and those features also have minor influences on the prediction, we need to train an additional binary meta-classifier to determine whether the suspicious model contains the knowledge of external features.

In this paper, we adopt the gradients of model weights as the input to train the meta-classifier $C_{\boldsymbol{w}} : \mathbb{R}^{|\boldsymbol{\theta}|} \to \{-1, +1\}$. Specifically, we assume that the victim model V and the suspicious model S have the same model structure. This assumption can be easily satisfied since the defender can retain a copy of the suspicious model on the training set of the deployed model as the victim model. Once the suspicious model is obtained, the defender will train its benign version ($i.e.$, the B) on the original training set $\mathcal{D}$. After that, we can obtain the training set $\mathcal{D}_c$ of meta-classifier C via

$$\begin{aligned}\mathcal{D}_c = &\{(g_V(\boldsymbol{x}'), +1) \,|(\boldsymbol{x}', y) \in \mathcal{D}_t\} \cup \\ &\{(g_B(\boldsymbol{x}'), -1) \,|(\boldsymbol{x}', y) \in \mathcal{D}_t\},\end{aligned} \tag{4}$$

where $\mathrm{sgn}(\cdot)$ is sign function (Sachs 2012), $g_V(\boldsymbol{x}') = \mathrm{sgn}(\nabla_{\boldsymbol{\theta}}\mathcal{L}(V(\boldsymbol{x}'), y))$, and $g_B(\boldsymbol{x}') = \mathrm{sgn}(\nabla_{\boldsymbol{\theta}}\mathcal{L}(B(\boldsymbol{x}'), y))$.

At the end, the meta-classifier $C_{\boldsymbol{w}}$ is trained by

$$\min_{\boldsymbol{w}} \sum_{(\boldsymbol{s},t)\in\mathcal{D}_c} \mathcal{L}(C_{\boldsymbol{w}}(\boldsymbol{s}), t). \tag{5}$$

In particular, we adopt its sign vector instead of the gradient itself to highlight the influence of its direction. We verify its effectiveness in **Appendix**.

4.4 Ownership Verification with Hypothesis Test

When the meta-classifier is trained, given a transformed image $\boldsymbol{x}'$ and its label y, the defender can examine the suspicious model simply by the result of $C(g_S(\boldsymbol{x}'))$, where $g_S(\boldsymbol{x}') = \mathrm{sgn}(\nabla_{\boldsymbol{\theta}}\mathcal{L}(S(\boldsymbol{x}'), y))$. If $C(g_S(\boldsymbol{x}')) = 1$, the suspicious model is considered as stolen from the victim. However, it may be sharply affected by the randomness of selecting $\boldsymbol{x}'$. In this paper, we design a hypothesis test based method to increase the verification confidence, as follows:

Definition 2. *Let $\boldsymbol{X}'$ denotes the variable of transformed images, while μ_S and μ_B indicates the posterior probability of the event $C(g_S(\boldsymbol{X}')) = 1$ and $C(g_B(\boldsymbol{X}')) = 1$, respectively. Given a null hypothesis $H_0 : \mu_S \leq \mu_B$ ($H_1 : \mu_S > \mu_B$), we claim that the suspicious model S is stolen from the victim if and only if the H_0 is rejected.*

In practice, we randomly sample m different transformed images from $\mathcal{D}_t$ to conduct the pair-wise T-test (Hogg, McKean, and Craig 2005) and calculate its p-value. When the p-value is smaller than the significance level α, H_0 is rejected. Besides, we also calculate the *confidence score* $\Delta\mu = \mu_S - \mu_B$ to represent the verification confidence. The larger the $\Delta\mu$, the more confident the verification.

5 Experiments

5.1 Settings

Dataset and Model Selection. We evaluate our defense on CIFAR-10 (Krizhevsky, Hinton et al. 2009) and (a subset of) ImageNet (Deng et al. 2009) dataset. Following the settings of (Maini, Yaghini, and Papernot 2021), we use the WideResNet (Zagoruyko and Komodakis 2016) and ResNet (He et al. 2016) as the victim model on CIFAR-10 and ImageNet, respectively. More detailed settings are in **Appendix**.

Settings for Model Stealing. Following the settings in (Maini, Yaghini, and Papernot 2021), we conduct model stealing methods illustrated in Section 2.1 to evaluate the effectiveness of defenses. Besides, we also provide the results of directly copying the victim model (dubbed 'Source') and examining a suspicious model which is not stolen from the victim (dubbed 'Independent') for reference. More detailed settings can be found in **Appendix**.

Defense Setup. We compare our defense with dataset inference (Maini, Yaghini, and Papernot 2021) and model watermarking (Adi et al. 2018) with BadNets (Gu et al. 2019),

Model Stealing		BadNets		Gradient Matching		Entangled Watermarks		Dataset Inference		Ours	
		$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
Victim	Source	0.91	10^{-12}	0.88	10^{-12}	**0.99**	$\mathbf{10^{-35}}$	-	10^{-4}	0.97	10^{-7}
$\mathcal{A}_D$	Distillation	-10^{-3}	0.32	10^{-7}	0.20	0.01	0.33	-	10^{-4}	**0.53**	$\mathbf{10^{-7}}$
$\mathcal{A}_M$	Zero-shot	10^{-25}	0.22	10^{-24}	0.22	10^{-3}	10^{-3}	-	10^{-2}	**0.52**	$\mathbf{10^{-5}}$
	Fine-tuning	10^{-23}	0.28	10^{-27}	0.28	0.35	0.01	-	10^{-5}	**0.50**	$\mathbf{10^{-6}}$
$\mathcal{A}_Q$	Label-query	10^{-27}	0.20	10^{-30}	0.34	10^{-5}	0.62	-	10^{-3}	**0.52**	$\mathbf{10^{-4}}$
	Logit-query	10^{-27}	0.23	10^{-23}	0.33	10^{-6}	0.64	-	10^{-3}	**0.54**	$\mathbf{10^{-4}}$
Benign	Independent	10^{-20}	0.33	10^{-12}	0.99	10^{-22}	0.68	-	**1.00**	**0.00**	**1.00**

Table 3: Results on CIFAR-10 dataset.

Model Stealing		BadNets		Gradient Matching		Entangled Watermarks		Dataset Inference		Ours	
		$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
Victim	Source	0.87	10^{-10}	0.77	10^{-10}	**0.99**	$\mathbf{10^{-25}}$	-	10^{-6}	0.90	10^{-5}
$\mathcal{A}_D$	Distillation	10^{-4}	0.43	10^{-12}	0.43	10^{-6}	0.19	-	10^{-3}	**0.61**	$\mathbf{10^{-5}}$
$\mathcal{A}_M$	Zero-shot	10^{-12}	0.33	10^{-18}	0.43	10^{-3}	0.46	-	10^{-3}	**0.53**	$\mathbf{10^{-4}}$
	Fine-tuning	10^{-20}	0.20	10^{-12}	0.47	0.46	0.01	-	10^{-4}	**0.60**	$\mathbf{10^{-5}}$
$\mathcal{A}_Q$	Label-query	10^{-23}	0.29	10^{-22}	0.50	10^{-7}	0.45	-	$\mathbf{10^{-3}}$	**0.55**	$\mathbf{10^{-3}}$
	Logit-query	10^{-23}	0.38	10^{-12}	0.22	10^{-6}	0.36	-	10^{-3}	**0.55**	$\mathbf{10^{-4}}$
Benign	Independent	10^{-24}	0.38	10^{-23}	0.78	$\mathbf{10^{-30}}$	0.55	-	0.98	10^{-5}	**0.99**

Table 4: Results on ImageNet dataset.

gradient matching (Geiping et al. 2021), and entangled watermarks (Jia et al. 2021). We poison 10% training samples for all defenses. Besides, we adopt a white-square in the lower right corner as the trigger pattern for BadNets and adopt a oil paint as the style image for our defense. Other settings are the same as those used in their original paper. An example of images (*e.g.*, poisoned images and the style image) involved in different defenses is shown in Figure 4.

Evaluation Metric. We use the confidence score $\Delta\mu$ and p-value for the evaluation metric. Following the settings adopted in (Maini, Yaghini, and Papernot 2021), both $\Delta\mu$ and p-value are calculated based on the hypothesis test with 10 sampled images. In particular, except for the independent sources (which should not be regarded as stolen), *the smaller the p-value and the larger the $\Delta\mu$, the better the defense*. Among all defenses, the best result is indicated in boldface.

5.2 Main Results

As shown in Table 3-4, our defense reaches the best performance in almost all cases. For example, the p-value of our method is three orders of magnitude smaller than that of the dataset inference and six orders of magnitude smaller than that of the model watermarking in defending against the distillation-based model stealing on CIFAR-10 dataset. The only exceptions appear when there is no model stealing. In these cases, entangled watermarks based model watermarking has some advantages. Nevertheless, our method can still easily make correct predictions in these cases. In particular, our defense method has minor adverse effects on the performance of victim models. For example, the accuracy of the model trained on benign CIFAR-10 and its transformed version is 91.99% and 91.79%, respectively. This is mainly because we do not change the label of transformed images and therefore the transformation can be treated as data augmentation, which is mostly harmless.

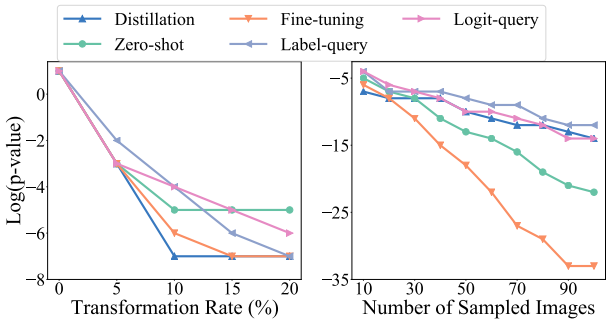


Figure 5: Effects of the transformation rate (%) and the number of sampled images.

5.3 Discussion

In this section, we discuss the effects of hyper-parameters and components involved in our method. Unless otherwise specified, all settings are the same as those in Section 5.2.

Effects of Transformation Rate. The larger the transformation rate γ, the more training samples are transformed during the training process of the victim model. As we expected, the p-value decrease with the increase of γ in defending all stealing methods (as shown in Figure 5). Note that the increase of γ may also lead to the accuracy decrease of victim models. Defenders should specify this hyper-parameter based on their specific requirements in practice.

Effects of the Number of Sampled Images. Recall that our method needs to specify the number of sampled (transformed) images (*i.e.*, the m) adopted in the hypothesis-based ownership verification. In general, the larger the m, the less the adverse effects of the randomness involved in this process and therefore the more confident the verification. This is probably the reason why the p-value also decreases with the increase of m, as shown in Figure 5.

Model Stealing		Pattern (a)		Pattern (b)		Pattern (c)		Pattern (d)	
		$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value	$\Delta\mu$	p-value
Victim	Source	0.98	10^{-7}	0.97	10^{-7}	0.98	10^{-10}	0.98	10^{-12}
$\mathcal{A}_D$	Distillation	0.68	10^{-7}	0.53	10^{-7}	0.72	10^{-8}	0.63	10^{-7}
$\mathcal{A}_M$	Zero-shot	0.61	10^{-5}	0.52	10^{-5}	0.74	10^{-8}	0.67	10^{-7}
	Fine-tuning	0.46	10^{-5}	0.50	10^{-6}	0.21	10^{-7}	0.50	10^{-9}
$\mathcal{A}_Q$	Label-query	0.64	10^{-5}	0.52	10^{-4}	0.68	10^{-8}	0.68	10^{-7}
	Logit-query	0.65	10^{-4}	0.54	10^{-4}	0.62	10^{-6}	0.73	10^{-7}
Benign	Independent	0.00	1.00	0.00	1.00	0.00	1.00	10^{-9}	0.99

Table 5: The effectiveness of our defense with different style images on CIFAR-10 dataset.

	Style Transfer		Meta-classifier	
	Patch-based Variant	Ours	BadNets	BadNets + Meta-classifier
Distillation	0.17	$\mathbf{10^{-7}}$	0.32	$\mathbf{10^{-3}}$
Zero-shot	0.01	$\mathbf{10^{-5}}$	0.22	$\mathbf{10^{-61}}$
Fine-tuning	10^{-3}	$\mathbf{10^{-6}}$	0.28	$\mathbf{10^{-5}}$
Label-query	10^{-3}	$\mathbf{10^{-4}}$	0.20	$\mathbf{10^{-50}}$
Logit-query	10^{-3}	$\mathbf{10^{-4}}$	0.23	$\mathbf{10^{-3}}$

Table 6: The effectiveness (p-value) of style transfer and meta-classifier on CIFAR-10 dataset.

(a) (b) (c) (d)

Figure 6: Style images adopted for the evaluation.

Effects of Style Image. In this part, we examine whether the proposed defense is still effective if we adopt other style images (as shown in Figure 6). As shown in Table 5, the p-value is significantly smaller than 0.01 in all cases. In other words, our method remains effective in defending against different stealing methods when different style images are used, although there will be some fluctuations in the results. We will further explore how to optimize the selection of style images in our future work.

Effectiveness of Style Transfer. To verify that the style watermark transfers better during the stealing process, we compare our method with its variant which uses the white-square patch (adopted in BadNets) to generate transformed images. As shown in Table 6, our method is significantly better than its patch-based variant. It is probably because DNNs are easier to learn the texture information (Geirhos et al. 2019) and the style watermark is bigger than the patch one. This phenomenon partly explains why our method works well.

Effectiveness of Meta-Classifier. To verify that the meta-classifier is also useful, we compare the BadNets-based model watermarking with its extension which also uses the meta-classifier (adopted in our method) for ownership verification. In this case, the victim model is the backdoored one and the transformed image is the one containing backdoor triggers. As shown in Table 6, adopting meta-classier significantly decrease the p-value in all cases, which verifies the effectiveness of the meta-classifier. These results also partly explains the effectiveness of our method.

Attack Stage $\rightarrow$	Source	First Stage	Second Stage
Attack Method $\rightarrow$	None	Zero-shot	Zero-shot
Model Structure $\rightarrow$	WRN-28-10	WRN-28-10	WRN-28-10
p-value $\rightarrow$	10^{-7}	10^{-5}	10^{-4}
Attack Method $\rightarrow$	None	Logit-query	Zero-shot
Model Structure $\rightarrow$	WRN-28-10	WRN-16-1	VGG-19
p-value $\rightarrow$	10^{-7}	10^{-4}	0.01

Table 7: Results in defending against multi-stage stealing.

5.4 Defending against Multi-Stage Model Stealing

In previous experiments, the stolen model is obtained by a single stealing attack. In this section, we explore whether our method is still effective if there are multiple stealing stages.

Settings. We discuss two types of multi-stage stealing on the CIFAR-10 dataset, including **(1)** stealing with the same attack and model structure and **(2)** stealing with different attacks and model structures. In general, the first one is the easiest multi-stage attack while the second one is the hardest. Other settings are the same as those used in Section 5.2.

Results. As shown in Table 7, the p-value ≤ 0.01 in all cases, *i.e.*, our method can successfully identify the existence of model stealing, even after multiple stealing stages. As we expected, the p-value in defending the second multi-stage attack is significantly larger than that of the first one indicating that the second task is harder. We will discuss how to better defend the second type of attack in our future work.

6 Conclusion

In this paper, we formulated the defense of model stealing as verifying whether a suspicious model contains the knowledge of defender-specified external features. Specifically, we embedded external features by modifying a few training samples with style transfer. This approach was inspired by the understanding that the stolen models should contain the knowledge of features learned by the victim model. We evaluated our defense on both CIFAR-10 and ImageNet datasets, which verified that our method can defend against various types of model stealing simultaneously while preserving high accuracy in predicting benign samples.

Acknowledgments

This work is supported in part by the National Key R&D Program of China under Grant 2019YFB1406500, the National Natural Science Foundation of China (62171248, U1736219, U1936210), the Guangdong Province Key Area R&D Program under Grant 2018B010113001, the R&D Program of Shenzhen (JCYJ20180508152204044), and the PCNL KEY project (PCL2021A07).

References

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *USENIX Security*.

Bagdasaryan, E.; and Shmatikov, V. 2021. Blind Backdoors in Deep Learning Models. In *USENIX Security*.

Chandrasekaran, V.; Chaudhuri, K.; Giacomelli, I.; Jha, S.; and Yan, S. 2020. Exploring connections between active learning and model extraction. In *USENIX Security*.

Chen, X.; Zhang, Y.; Wang, Y.; Shu, H.; Xu, C.; and Xu, C. 2020. Optical flow distillation: Towards efficient and stable video style transfer. In *ECCV*.

Cheng, S.; Liu, Y.; Ma, S.; and Zhang, X. 2021. Deep Feature Space Trojan Attack of Neural Networks by Controlled Detoxification. In *AAAI*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Duan, R.; Ma, X.; Wang, Y.; Bailey, J.; Qin, A. K.; and Yang, Y. 2020. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *CVPR*.

Fang, G.; Song, J.; Shen, C.; Wang, X.; Chen, D.; and Song, M. 2019. Data-free adversarial distillation. *arXiv preprint arXiv:1912.11006*.

Geiping, J.; Fowl, L.; Huang, W. R.; Czaja, W.; Taylor, G.; Moeller, M.; and Goldstein, T. 2021. Witches' brew: Industrial scale data poisoning via gradient matching. In *ICLR*.

Geirhos, R.; Rubisch, P.; Michaelis, C.; Bethge, M.; Wichmann, F. A.; and Brendel, W. 2019. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*.

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7: 47230–47244.

Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; and Bennamoun, M. 2020. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hinton, G.; Vinyals, O.; and Dean, J. 2014. Distilling the knowledge in a neural network. In *NeurIPS Workshop*.

Hogg, R. V.; McKean, J.; and Craig, A. T. 2005. *Introduction to mathematical statistics*. Pearson Education.

Huang, X.; and Belongie, S. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*.

Jagielski, M.; Carlini, N.; Berthelot, D.; Kurakin, A.; and Papernot, N. 2020. High accuracy and high fidelity extraction of neural networks. In *USENIX Security*.

Jia, H.; Choquette-Choo, C. A.; Chandrasekaran, V.; and Papernot, N. 2021. Entangled watermarks as a defense against model extraction. In *USENIX Security*.

Johnson, J.; Alahi, A.; and Fei-Fei, L. 2016. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*.

Juuti, M.; Szyller, S.; Marchal, S.; and Asokan, N. 2019. PRADA: protecting against DNN model stealing attacks. In *EuroS&P*.

Kariyappa, S.; and Qureshi, M. K. 2020. Defending against model stealing attacks with adaptive misinformation. In *CVPR*.

Kesarwani, M.; Mukhoty, B.; Arya, V.; and Mehta, S. 2018. Model extraction warning in mlaas paradigm. In *ACSAC*.

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

Lee, T.; Edwards, B.; Molloy, I.; and Su, D. 2019. Defending against neural network model stealing attacks using deceptive perturbations. In *IEEE S&P Workshop*.

Li, Y.; Li, Y.; Wu, B.; Li, L.; He, R.; and Lyu, S. 2021a. Invisible Backdoor Attack with Sample-Specific Triggers. In *ICCV*.

Li, Y.; Wu, B.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2020a. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745*.

Li, Y.; Zhai, T.; Jiang, Y.; Li, Z.; and Xia, S.-T. 2021b. Backdoor Attack in the Physical World. In *ICLR Workshop*.

Li, Y.; Zhang, Z.; Bai, J.; Wu, B.; Jiang, Y.; and Xia, S.-T. 2020b. Open-sourced Dataset Protection via Backdoor Watermarking. In *NeurIPS Workshop*.

Liu, H.; Weng, Z.; and Zhu, Y. 2021. Watermarking Deep Neural Networks with Greedy Residuals. In *ICML*.

Maini, P.; Yaghini, M.; and Papernot, N. 2021. Dataset inference: Ownership resolution in machine learning. In *ICLR*.

Minaee, S.; Boykov, Y. Y.; Porikli, F.; Plaza, A. J.; Kehtarnavaz, N.; and Terzopoulos, D. 2021. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *CVPR*.

Nguyen, A.; and Tran, A. 2020. Input-aware dynamic backdoor attack. In *NeurIPS*.

Orekondy, T.; Schiele, B.; and Fritz, M. 2019. Knockoff nets: Stealing functionality of black-box models. In *CVPR*.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *AsiaCCS*.

Sachs, L. 2012. *Applied statistics: a handbook of techniques*. Springer Science & Business Media.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Stokes, J. M.; Yang, K.; Swanson, K.; Jin, W.; Cubillos-Ruiz, A.; Donghia, N. M.; MacNair, C. R.; French, S.; Carfrae, L. A.; Bloom-Ackermann, Z.; et al. 2020. A deep learning approach to antibiotic discovery. *Cell*, 180(4): 688–702.

Szyller, S.; Atli, B. G.; Marchal, S.; and Asokan, N. 2021. Dawn: Dynamic adversarial watermarking of neural networks. In *ACM MM*.

Tramèr, F.; Zhang, F.; Juels, A.; Reiter, M. K.; and Ristenpart, T. 2016. Stealing machine learning models via prediction apis. In *USENIX Security*.

Wang, T.; and Kerschbaum, F. 2021. RIGA: Covert and Robust White-Box Watermarking of Deep Neural Networks. In *WWW*.

Yan, H.; Li, X.; Li, H.; Li, J.; Sun, W.; and Li, F. 2021. Monitoring-based Differential Privacy Mechanism Against Query Flooding-based Model Extraction Attack. *IEEE Transactions on Dependable and Secure Computing*.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *BMVC*.

Zhang, J.; Chen, D.; Liao, J.; Zhang, W.; Hua, G.; and Yu, N. 2020. Passport-aware Normalization for Deep Model Protection. *NeurIPS*.

Zhang, J.; Gu, Z.; Jang, J.; Wu, H.; Stoecklin, M. P.; Huang, H.; and Molloy, I. 2018. Protecting intellectual property of deep neural networks with watermarking. In *AsiaCCS*.