MetaV: A Meta-Verifier Approach to Task-Agnostic Model Fingerprinting

Xudong Pan
xdpan18@fudan.edu.cn
School of Computer Science
Fudan University, China

Mi Zhang
mi_zhang@fudan.edu.cn
School of Computer Science
Fudan University, China

Yifan Yan
yanyf20@fudan.edu.cn
School of Computer Science
Fudan University, China

Min Yang
m_yang@fudan.edu.cn
School of Computer Science
Fudan University, China

ABSTRACT

Protecting the intellectual property (IP) of deep neural networks (DNN) becomes an urgent concern for IT corporations. For model piracy forensics, previous model fingerprinting schemes are commonly based on adversarial examples constructed for the owner’s model as the fingerprint, and verify whether a suspect model is indeed pirated from the original model by matching the behavioral pattern on the fingerprint examples between one another. However, these methods heavily rely on the characteristics of classification tasks which inhibits their application to more general scenarios. To address this issue, we present MetaV, the first task-agnostic model fingerprinting framework which enables fingerprinting on a much wider range of DNNs independent from the downstream learning task, and exhibits strong robustness against a variety of ownership obfuscation techniques. Specifically, we generalize previous schemes into two critical design components in MetaV: the adaptive fingerprint and the meta-verifier, which are jointly optimized such that the meta-verifier learns to determine whether a suspect model is stolen based on the concatenated outputs of the suspect model on the adaptive fingerprint. As a key of being task-agnostic, the full process makes no assumption on the model internals in the ensemble only if they have the same input and output dimensions. Spanning classification, regression and generative modeling, extensive experimental results validate the substantially improved performance of MetaV over the state-of-the-art fingerprinting schemes and demonstrate the enhanced generality of MetaV for providing task-agnostic fingerprinting. For example, on fingerprinting ResNet-18 trained for skin cancer diagnosis, MetaV achieves simultaneously 100% true positives and 100% true negatives on a diverse test set of 70 suspect models, achieving an about 220% relative improvement in ARUC over the optimal baseline.

CCS CONCEPTS

- Security and privacy → Domain-specific security and privacy architectures;
- Computing methodologies → Neural networks.

KEYWORDS

Fingerprinting, Intellectual Property, Deep Learning

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD ’22, August 14–18, 2022, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9385-0/22/08...
https://doi.org/10.1145/3534678.3539257

1 INTRODUCTION

In the past decades, deep learning finds a wide application in a variety of mission-critical scenarios in the real world, including autonomous driving [6], finance [15], intelligent healthcare [11], and many more. In the ever-evolving trend of applying deep learning in IT industry, increasingly more high-end computing power and massive amounts of well-annotated data are devoted to the construction of deep neural networks (DNN) [10, 29, 46], which are later deployed as prediction APIs, i.e., Machine-Learning-as-a-Service (MLaaS), to provide intelligent service for profiting. Considering the substantial training costs, many IT corporations as the model owners become aware of the importance of protecting the confidentiality of those well-trained DNN, as an inseparable part of their intelligent property (IP). Threateningly, even with careful access control, an attacker can still pirate the working DNN behind an online intelligent service by conducting system [19, 41] or algorithmic attacks [34, 43].

Orthogonal to the advances in protecting DNNs against model privacy [20], model watermarking and model fingerprinting are two fast-developing techniques for model piracy forensics. Applying model watermarking, the model owner embeds a secret into his/her owned model (i.e., the target model). Once the ownership of a DNN model is in doubt (i.e., the suspect model), a trusted third party verifies the existence of the exclusively known secret in the suspect model to determine the actual ownership. From Uchida et al. [36], previous works devise different types of secrets (e.g., a specific function or a specific parameter pattern) into various parts of
a DNN, which we briefly survey in Section 2. However, because model watermarking unavoidably modifies the original parameters of a well-trained DNN for secret embedding, the otherwise optimal accuracy would be slightly degraded, causing an unacceptable trade-off for mission-critical tasks in healthcare and traffic [5].

Complementary to model watermarking, model fingerprinting is a passive forensic technique against model piracy, which in general tests whether certain fingerprint of the target model are present in a suspect model, which would help collect essential evidence of model piracy in the wild before filing a lawsuit. As a key difference from model watermarking, the fingerprint is innate but not embedded to the target model. In other words, no modifications on the target model are conducted during fingerprinting, which provably preserves the normal utility of well-trained DNNs. Thanks to this desirable characteristic, model fingerprinting arises as a booming direction in model protection from the last year, which attracts increasing research efforts from different backgrounds [5, 24, 25, 37].

Following the fingerprinting framework in Cao et al. [5], previous schemes mostly focus on fingerprinting classifiers by constructing a special set of adversarial examples [33], i.e., normal examples added with human- imperceptible perturbations which cause misclassification of the target classifier, as the fingerprint, and verifying whether a suspect model is indeed stolen from the original model by matching the behavioral pattern, e.g., the predicted labels [25, 37] or probability vector similarity [24], on the fingerprint examples. Despite their pioneering contributions to model IP protection, existing schemes are however limited to fingerprinting DNNs for other important downstream tasks except for classification, mainly because they commonly rely on concepts like adversarial examples and classification boundary which have no direct counterparts in other typical learning tasks such as regression and generative modeling. With recent years witnessing the fast trend of distribution, deployment and redistribution of DNNs in nowadays deep learning ecosystem, how to conduct forensics on the improper reuse and illegal piracy for a more general set of DNNs poses an urgent open challenge to address.

1.1 Our Work

In this paper, we present a meta-verifier approach to task-agnostic model fingerprinting scheme (dubbed as MetaV), which for the first time enables fingerprinting on a much wider range of DNNs independent from the downstream learning task, and by design implements the robustness against a variety of ownership obfuscation techniques possibly adopted by the adversary.

To realize task-agnostic model fingerprinting, we generalize the idea of using adversarial examples and the corresponding classification results for fingerprinting respectively into two critical design components in MetaV, i.e., the adaptive fingerprint and the meta-verifier. Concisely, the adaptive fingerprint is a set of trainable inputs to the suspect model, the concatenated outputs of the suspect model on which are classified by the meta-verifier to be True or False, where True implies the suspect model is indeed stolen (i.e., positive suspect model), and False implies the suspect model is independent from the original model (i.e., negative suspect model).

To implement the design principles above, the adaptive fingerprint and the meta-verifier are jointly optimized on an ensemble composed of the target model, the positive and the negative suspect models which are virtually generated by MetaV during the fingerprinting construction phase. Specifically, the positive suspect models in the ensemble are crafted by post-processing the target model with a number of popular ownership obfuscation techniques, e.g., compression [13, 23], fine-tuning, partial retraining and distillation [16], while the irrelevant models are independently trained from scratch on similar learning tasks to the target model. As the full construction process of MetaV makes no assumption on the model internals or functions in the ensemble only if they have the same input and output dimensions, MetaV is therefore applicable independent of the downstream tasks for which the DNN is designed. Moreover, by permitting more types of obfuscation techniques in producing the stolen models for the model ensemble, our proposed MetaV is by construction robust against a diverse set of existing obfuscation techniques, with the potential to evolve along with future adversarial techniques.

In summary, we mainly make the following contributions:

- We present MetaV, the first task-agnostic fingerprinting framework with adaptive robustness against a variety of ownership obfuscation techniques, to substantially advance the cutting-edge model fingerprinting capability to a much broader set of DNNs for arbitrary downstream tasks.
- We generalize the existing fingerprinting schemes based on adversarial examples to a more universal fingerprinting framework based on the adaptive fingerprint and the meta-verifier, which are jointly optimized on an ensemble of the target model, the positive and negative suspect models crafted by the model owner to serve as a highly effective and robust fingerprint for the target model.
- We extensively evaluate the performance of MetaV on practical scenarios spanning classification, regression and generative modeling. Besides the unique contribution of MetaV in providing task-agnostic fingerprinting, MetaV brings noticeable improvement over all the state-of-the-art fingerprinting schemes on DNN classifiers. For example, on fingerprinting ResNet-18 [14] trained for skin cancer diagnosis [11], MetaV achieves simultaneously 100% true positives and 100% true negatives on a diverse test set of 70 suspect models, with an about 220% relative improvement in ARUC compared to the optimal baseline.

2 RELATED WORKS

2.1 Model Fingerprinting

Recently, a number of fingerprinting schemes, mainly based on constructing different types of adversarial examples as the model fingerprint, were proposed to protect the intellectual property of DNN classifiers. For example, Cao et al. [5] propose IPGuard, one of the earliest fingerprinting schemes, to find adversarial examples near the decision boundary of the target classifier. The key assumption of IPGuard is that the target DNN classifier can be uniquely represented by its decision boundary, which is more similar with the classification boundary of a positive suspect model than a negative one. Different from IPGuard, Lukas et al. [25] and Zhao et al. [45] independently propose to extract so-called conferrable adversarial examples from the ensemble of the target classifier and a set
of locally trained suspect classifiers. These conferrable adversarial examples, which transfer much better to the positive suspect models than to negative ones, can be regarded as a unique link between the target model and the positive suspect models. Besides, Wang and Chang [37] utilize the geometry characteristics inherited in the DeepFool algorithm to construct adversarial examples as the fingerprint [37], while Li et al. [24] leverage the similarity between models in terms of the probability vectors on test inputs for piracy detection [24]. More detailed surveys can be found in [4, 30]. In this work, our proposed MetaV generalizes the aforementioned fingerprinting techniques to abstract the usage of adversarial examples and the classification results into the adaptive fingerprint and the trainable meta-verifiers, which is applicable to arbitrary DNN models in a task-agnostic way.

### 2.2 Model Watermarking

Orthogonal to model fingerprinting, model watermarking embeds a watermark into the trained model before it is released, which potentially sacrifices the utility of the model. Previous works on model watermarking explore various types of watermarks such as secret bit strings [31, 36], generated serial numbers [40] and unrelated or slightly modified sample sets [2, 44]. These identifying codes are then encoded secretly into the least significant bit of the weight [36], the distribution of outputs at the intermediate [31] or the full layers [2, 40, 44].

### 3 SECURITY SETTINGS

#### 3.1 Backgrounds & Notions

Model fingerprinting is a multi-party security game among a model owner, a verifier and an attacker. Initially, a model owner devotes computing power and well-curated training data to build its own DNN $F : X \rightarrow Y$ for a certain downstream task, crowning the obtained model $F$ as an inseparable part of the model owner’s IP. Following the nomenclature in Cao et al. [5], we refer to the model $F$ as the target model. In nowadays deep learning ecosystems, the model owner can deploy the target model at a third-party platform like Amazon AWS as a prediction API to gain monetary profits. However, the profits may also serve as incentives on the potential attacker to conduct model piracy via, e.g., software/hardware vulnerabilities [19, 41], social engineering and algorithms against model ownership due to the transformed model architecture inherited by the student model, which exacerbates the obfuscation of the obfuscated model on the learning task [13] and inhibits heuristic-based fingerprinting from verifying the model ownership based on parameter comparison [5].

**Attacker’s Capability.** We assume the attacker would apply a variety of model post-processing techniques to obfuscate the ownership of the stolen model (detailed in the subsequent part) after he/she successfully steals the target model from an online prediction API. Such an obfuscated model is called a positive suspect model. Correspondingly, a suspect model independently trained by another honest model owner is called a negative suspect model. The attacker is assumed to answer any queries to his/her provided prediction API, as more queries served by the API bring more monetary profits.

**Verifier’s Capability.** Following, e.g., Cao et al. [5], we assume the verifier has a white-box access to the target model while a black-box access to the suspect models via their prediction APIs. To relax their assumptions, we do not assume the internal architecture or the type of downstream tasks of the target model.

#### 3.2 Threat Model

We mainly consider the following threat model in this paper.

- **Model Compression:** Compression-based obfuscation adopts weight (filter) pruning [23] to remove a certain ratio of small weights (filters) in a DNN, which would largely preserve the utility of the obfuscated model on the learning task [13] and inhibits heuristic-based fingerprinting from verifying the model ownership based on parameter comparison [5].

- **Fine-Tuning & Partial Retraining:** To obfuscate the behavioral pattern of a DNN, attackers may resume the training of the stolen model on public data collected from a similar domain of the training data. Specifically, to fine-tune the last $K$ layers of a trained DNN, the parameters of the last $K$ layers are further updated according to the learning objective with the other layers fixed. In comparison, during the partial retraining, the parameters of the last $K$ layers are first randomly initialized before the training is resumed. Due to the non-convexity of deep learning [8], both the fine-tuned and partially-retrained models may fall into a different local optimum, preserve the original utility, but exhibit divergent prediction behaviors [37].

- **Model Distillation:** Distillation-based obfuscation adopts the knowledge distillation strategies [12, 16] by viewing the stolen model or the corresponding prediction API as the teacher model, and a DNN of a different architecture as the student model [24]. Via distillation, the learned knowledge in the target model is inherited by the student model, which exacerbates the obfuscation of model ownership due to the transformed model architecture and the correspondingly altered predictive behaviors [25].

### 4 OUR METHODOLOGY FOR TASK-AGNOSTIC FINGERPRINTING

#### 4.1 Overview of MetaV

As a generalization of previous fingerprinting schemes, our proposed MetaV abstracts the usage of adversarial examples and the
corresponding prediction results as the fingerprint into two critical components respectively: (i) adaptive fingerprint, i.e., a set of trainable inputs to the target/suspect model(s), which we denote as $X_F := (x^F_1, \ldots, x^F_N)$ with $x^F_i \in \mathcal{X}$ and $N$ called the number of fingerprint examples, and (ii) meta-verifier, i.e., a binary classifier which takes the concatenated outputs of a suspect model on the adaptive fingerprint as its input and predicts whether the suspect model is positive or negative, denoted as $\mathcal{V} : \mathcal{Y}^n \rightarrow \mathbb{S}_2$, a 2-dimensional probability simplex $\{(p_0, 1-p_0)|0 \leq p_0, p_1 \leq 1\}$. As illustrated in Fig. 1, the general pipeline of MetaV mainly consists of the following three key stages:

- **Stage 1. (Model Ensemble Preparation)** As Fig. 1(a) shows, we first craft a number of positive and negative suspect models from the target model with the aid of public data from the same domain of the owner’s training data. At the end of this stage, we obtain the sets of positive and negative suspect models, i.e., $M_+$ and $M_-$. 

- **Stage 2. (Fingerprint Construction in MetaV)** As Fig. 1(b) shows, we then jointly optimize the adaptive fingerprint $X_F$ and the meta-verifier $\mathcal{V}$ to satisfy: For both the target model or any suspect model in $M_+$, the meta-verifier $\mathcal{V}$ is trained to predict True, i.e., $p_+ > p_-$, on the concatenated outputs of the model on examples in $X_F$, and vice versa for any suspect model in $M_-$. At the end of this stage, we obtain optimized adaptive fingerprint and the corresponding meta-verifier, i.e., $X^*_F$ and $\mathcal{V}^*$ respectively. We call $(X^*_F, \mathcal{V}^*)$ a fingerprinting pair.

- **Stage 3. (Fingerprint Verification in MetaV)** Finally, with the optimized fingerprinting pair $(X^*_F, \mathcal{V}^*)$, we verify whether and with what matching rate a suspect model $\hat{f}$ is a stolen version or an independently trained one by querying the prediction API with the fingerprint examples in $X_F$. The received prediction results are then concatenated and input to the meta-verifier to predict $(\hat{p}_- \hat{p}_+) = \mathcal{V}(\hat{f}(x^F_1) \oplus \ldots \oplus \hat{f}(x^F_N))$. When $\hat{p}_+$ is larger than a predefined threshold $\rho$, the verification process outputs True to claim possible model piracy behind the tested prediction API, or other the process outputs False to assert the fidelity of the suspect model.

In the following sections, we elaborate on the detailed methodology for the first two stages of MetaV.

### 4.2 Model Ensemble Preparation

To construct the adaptive fingerprint and the meta-verifier with simultaneously high robustness and uniqueness, MetaV is first required to collaborate with the model owner to prepare a diverse set of positive and negative suspect models. Intuitively, with a more representative set of positive suspect models, the obtained fingerprinting pair would stay robust against a wider range of ownership obfuscation techniques, resulting in higher true positives. Alternatively, more representative negative suspect models would lower the probability of the learned fingerprinting pair to be present in other irrelevant models, which therefore reduces the true negatives of MetaV. Specifically, the positive and negative suspect models are constructed as follows.

- **Prepare Positive Suspect Models.** We derive a representative set of positive suspect models by randomly applying one or more common ownership obfuscation techniques mentioned in Section 3 to the target model $F$. The applied obfuscation techniques are recommended to cover a wider range of hyperparameter configurations for better robustness. For example, we apply weight and filter pruning to the target model with different pruning ratios. As a notation, we denote the full set of obfuscation techniques for preparing the positive suspect models as $\mathcal{T}$. Therefore, we have $M_+ := \{T \circ f | T \in \mathcal{T}\}$. Section 5 presents the detailed composition of $\mathcal{T}$ in the evaluation settings part.

- **Prepare Negative Suspect Models.** We recommend three complementary sources to collect a diverse set of negative suspect models for the fingerprint construction stage of MetaV. First, MetaV may request the model owner to train a moderate number of relatively small-scale DNNs on the same training dataset of the target model. For IP protection of the target model, it would be reasonable for the model owner to devote additional computing power to collaborate with a trusted third-party verifier. Second, MetaV can also download a number of pretrained models

![Figure 1: The general pipeline of our proposed MetaV: (a) model ensemble preparation and (b) fingerprint construction.](image-url)
from online sources (e.g., PyTorch Hub) and fine-tune these models on the domain-relevant public data to serve as the negative suspect models. Moreover, MetaV may consider incorporate a proportion of irrelevant publicly available models into the set of negative suspect models to further enhance the uniqueness of the obtained fingerprinting pair. We denote the prepared negative suspect models as $M_-$.  

### 4.3 Fingerprint Construction in MetaV

In this part, we detail the learning objective and the optimization algorithm for training the fingerprint pair $(X_F, \mathcal{V})$ on the model ensemble $M_- \cup \{F\} \cup M_+$ prepared in the first stage. As Fig. 1(b) shows, the learning objective of MetaV is viewed as a binary classification problem. To formulate, we introduce an additional label $\ell$ for each model, which takes value in $\{+,-\}$ (literally, positive and negative respectively). Specifically, we label an arbitrary model $M_+ \in M_+ \cup \{F\}$ as $+$ and an arbitrary model $M_- \in M_- \cup \{F\}$ as $-$. To supervise the adaptive fingerprint and the meta-verifier with the labels, we solve the learning objective:

$$
\arg \max_{X_F, \mathcal{V}} \log p_+(F) + \frac{1}{|M_+|} \sum_{M_+ \in M_+} \log p_+(M_+) + \frac{1}{|M_-|} \sum_{M_- \in M_-} \log p_-(M_-),
$$

(1)

where $(p_+(M), p_-(M)) = \mathcal{V}(M(x_F^i) \oplus \ldots \oplus M(x_F^N))$, the prediction from the meta-verifier on the concatenated outputs of a model under test on the adaptive fingerprint. Intuitively, the learning objective above encourages the meta-verifier to output a $p_+$ higher than $p_-$ when the model under test is a positive suspect model or the target model, and vice versa for a negative suspect model.

As the learning objective above is fully derivative w.r.t. the adaptive fingerprint $X_F$ and the parameters of the meta-verifier, we leverage off-the-shelf non-convex optimizers (e.g., Adam [21]) for gradient-based optimization. However, we notice it is resource-consuming to conduct back-propagation over the whole model ensembles in each optimization step. As an alternative, we reformulate the batched learning objective in (1) as a stochastic objective with randomness in a tuple of $(M_-, F, M_+)$ uniformly sampled from $M_- \cup \{F\} \cup M_+$ in each iteration. Besides, we adopt the reparametrization trick in Carlini and Wagner [7] to constrain the adaptive fingerprint in the problem space $X$. Taking $X := [-1,1]^d$, a common case in computer vision for example, Algorithm 1 presents the details of the optimization algorithm.

### 5 EVALUATION SETTINGS

#### 5.1 Scenarios and Datasets

Table 2 provides an overview on the three scenarios, i.e., skin cancer diagnosis (Yang et al. [42], classification), warfarin dose prediction (Whirl-Carrillo et al. [38], regression), and fashion generation (Xiao et al. [39], generative modeling), covered in the evaluation sections. The information of the datasets are concisely introduced below.

- **Skin Cancer Diagnosis** (abbrev. Skin). The first scenario covers the usage of deep convolutional neural network (CNN) for skin cancer diagnosis. According to [42], we train a ResNet-18 [14] as the target model on DermAMNIST [42], which consists of 10005 multi-source dermatoscopic images of common pigmented skin lesions imaging dataset. The input size is originally $3 \times 28 \times 28$, which is upsampled to be $3 \times 224 \times 224$ to fit the input shape of a standard ResNet-18 architecture implemented in torchvision [1]. The task is a 7-class classification task.
- **Warfarin Dose Prediction** (abbrev. Warfarin). The second scenario covers the usage of FCN for warfarin dose prediction, which is a safety-critical regression task that helps predict the proper individualised warfarin dosing according to the demographic and physiological record of the patients (e.g., weight, age and genetics). We use the International Warfarin Pharmacogenetics Consortium (IWPC) dataset [38], which is a public dataset composed of 31-dimensional features of 6256 patients and is widely used for researches in automated warfarin dosing. According to Truda and Marais [35], we use a three-layer multi-layer perceptron (MLP) with ReLU as the target model, with its hidden layer composed of 100 neurons. As a notation, we denote the architecture as (31-100-1). The target model learns to predict the value of proper warfarin dosing, which is a non-negative real-valued scalar with its value in $(0,300.0]$.
- **Fashion Generation** (abbrev. Fashion). The final scenario covers the usage of FCN for generative modeling. We choose [39], which consists of 60000 images for articles of clothing of size $28 \times 28$. We train a DCGAN-like architecture [28] for generative modeling on this task. We solely view the generator as the target model, as a well-trained generator represents more the IP of the model owner because it can be directly used to generate realistic images without the aid of the discriminator. The detailed DCGAN architecture we use is demonstrated in Table 4.

#### 5.2 Fingerprinting Benchmarks

For each scenario, we construct a model benchmark composed of 140 positive/negative suspect models. We split the benchmark randomly by a ratio of 1 : 1 into two independent sets of suspect models for training and testing.

- **Constructing Positive Suspect Models**. Following Cao et al. [5] and Lukas et al. [25], we apply a number of popular ownership obfuscation techniques with a variety of hyperparameter configurations on the target model to derive the positive suspect models:

  (1) **Compression**: For weight pruning, we vary the ratio of pruned weights from 0.1 to 0.9 with a stride of 0.1. For filter pruning, we choose the ratio of pruned filters from 1/16 to 15/16 with a stride of 1/16.

  (2) **Fine-Tuning & Partial Retraining**: We consider 4 types of obfuscation in this category, i.e., fine-tuning/retraining the last layer and fine-tuning/retraining all layers. For both types of retraining, the last one layer is first reset as a randomly initialized layer, after which the model is finetuned according to the configuration. We set the number of epochs for fine-tuning and retraining both as 10.

  (3) **Distillation**: For each target model, we select 3-5 diverse models with different architectures as the student model. For the ResNet-18 classifier, we follow the classical distillation algorithm in Hinton et al. [16] to prepare the student model. We do not consider other model distillation algorithms because most of them require the access to the internals of the target
model (i.e., the teacher) for distillation, implausible for an attacker who pirates the model from the prediction API. For the multi-layer perception (MLP) as the regressor and the DCGAN [28] as the generator, we implement the distillation algorithms in Clark et al. [9] and Aguinaldo et al. [3] respectively. For fine-tuning, partial retraining and distillation, we mutate the random seeds to produce multiple suspect models belonging to the corresponding category.

- **Constructing Negative Suspect Models.** To construct the negative suspect models, we use different random seeds to initialize models of different architectures. We then train the models from scratch respectively on the original training data, on the public data from a similar domain of the training set, and on other irrelevant dataset to obtain a diverse benchmark of negative suspect models. Table 3 lists the composition of the suspect models for all the three scenarios. For convenience, we use the following abbreviation: fine-tuning the last layer (\(=\)FTLL), fine-tuning all layers (\(=\)FTAL), retraining the last layer (\(=\)RTLAL), retraining all layers (\(=\)RTLAL), weight-pruning (\(=\)WP), filter-pruning (\(=\)FP). For constructing distillation-based positive suspect models and independently trained negative suspect models, we implement 3-5 models of diverse architectures and incremental sizes for each of the three target models. For convenience, we index these models as \(S, M, L, XL, XLL\). Specifically, these models are:

- **Skin:** \(S\)=SqueezeNet-1-0 [18]; \(M\)=ResNet-18 (He et al. [14], the same as the target model); \(L\)=DenseNet-161 [17]; \(XL\)=AlexNet [22]; \(XXL\)=VGG-16 [32].
- **Warfarin:** \(S\)\=(31-100-1) (the same as the target model); \(M\)\=(31-100-100-1); \(L\)\=(31-100-100-100-1).
- **Fashion:** \(S\)=Architecture in Table 5 with \(s \leq 1\); \(M\)=with \(s = 2\); \(L\)=with \(s = 3\); \(XL\)=the same as the target model in Table 4.

5.3 Baselines

We cover 4 state-of-the-art fingerprinting schemes as the baselines for evaluating the effectiveness of MetaV under the classification settings. The baselines are respectively:

- **IPGuard** [5]: IPGuard is one of the earliest fingerprinting schemes on classification models, which searches for a set of adversarial examples with a specified label near the decision boundary of the target model as the fingerprinting examples.
- **ConferAE** [25]: ConferAE improves the design of IPGuard by further covering the distillation-based obfuscation techniques. Specifically, ConferAE constructs a set of adversarial examples on a prepared ensemble of positive/negative suspect models which may be implemented with different architectures from the target model. The generated adversarial examples are additionally required to be transferable from the target model to the positive suspect models but not to the negative ones.
- **DeepFoolFP** [37]: DeepFoolFP literally leverages the DeepFool algorithm [26] to generate adversarial examples as the model fingerprints, the motivation of which is to improve the efficiency of fingerprint construction. For verification, the above model fingerprinting schemes define the matching rate as the ratio of the fingerprinting examples which are correctly classified by the suspect model into the specified class.

- **ModelDiff** [24]: ModelDiff is a very recent technique which is originally proposed to quantify the behavioral similarity between a pair of models. Specifically, the similarity is measured as the cosine similarity of two models’ decision distance vector, each element of which is the distance of the logits between a clean test input and an adversarial example derived from the input. A suspect model is verified if its behavioral similarity with the target model is smaller than a threshold.

With no further specifications, the number of fingerprint examples, i.e., \(N\), is set as 100 for MetaV and the baselines by default. More details on the baselines are in Section 2.

5.4 Performance Metrics

Given a predefined threshold \(\epsilon \in (0, 1)\), MetaV and all the baseline methods recognize a suspect model as positive when the matching rate of fingerprint verification is higher than \(\epsilon\), or otherwise recognize the model as negative. In the evaluation, we follow the evaluation protocol in [5] which is composed of the metrics below:

1. **Robustness/Uniqueness** \((R(\rho)/U(\rho))\): The robustness metric measures the proportion of positive suspect models also recognized as positive by the fingerprinting scheme, i.e., true positives.
2. **Uniqueness** \((U(\rho))\): The uniqueness metric measures the proportion of negative suspect models also recognized as negative by the fingerprinting scheme, i.e., true negatives.
3. **Area under the Robustness-Uniqueness Curves** (ARUC): ARUC measures the area of the intersection region under the robustness and uniqueness when the threshold varies in \((0, 1)\), i.e., \(\int_{0}^{1} \min(\min(\{R(\rho), U(\rho)\}) \, d\rho\). A higher ARUC implies a more wider value range for the threshold to choose from to obtain simultaneously high robustness and uniqueness. ARUC is empirically calculated as the average \(\min(\{R(\rho), U(\rho)\})\) on \((0, 1/L, \ldots, (L-1)/L, 1)\) with \(L = 100\). For all the experiments, we run 5 repetitive experiments and report the average metric with the 95% confidence interval.

5.5 Other Implementation Details

5.5.1 Hyperparameter Setups. With no further specifications, we always set the number of fingerprint examples, i.e., \(N\), for MetaV and the baselines as 100 for fair comparisons. We set the learning rate in Algorithm 1 as 0.001 and the number of iteration as 1000. In all the three scenarios, we implement the meta-verifier \(V\) as a three-layer fully-connected neural network with the ReLU hidden layer size of 100.

5.5.2 Experimental Environment. All the defenses and experiments are implemented with PyTorch [27], an open-source software framework for numeric computation and deep learning. All our experiments are conducted on a Linux server running Ubuntu 16.04, one AMD Ryzen Threadripper 2990WX 32-core processor and 2 NVIDIA GTX RTX2080 GPUs.
6 RESULTS & ANALYSIS

6.1 Comparison with Baselines

First, we compare the performance of MetaV with 4 state-of-the-art model fingerprinting schemes specifically designed for classifiers. Fig. 3 reports the robustness (i.e., true positives) when the threshold $\rho$ is set to allow the uniqueness (i.e., true negatives) to reach 100% on the test set, along with the corresponding ARUC presented in Fig. 2(a). As Fig. 3 shows, our proposed MetaV is the only method which simultaneously achieves 100% robustness and uniqueness in fingerprinting a stolen and adversarially obfuscated ResNet-18 classifier for skin cancer diagnosis. Besides, as we can see from Fig.2(a), MetaV constructs the model fingerprint with the highest ARUC metric, i.e., $0.86 \pm 0.01$, among all the tested fingerprinting schemes, which improves the optimal baseline IPGuard by 0.59 absolutely, i.e., a roughly 220% relative improvement. As we construct a more diverse benchmark of suspect models compared with previous works, the ARUC of IPGuard is not as high as the results reported in Cao et al. [5]. Fig. 2(b)-(f) show the robustness and uniqueness curves of each fingerprint schemes.

6.2 Time Efficiency of MetaV

Next, we empirically study the learning behaviors and the time complexity of MetaV when constructing the fingerprint of a ResNet-18. As Fig. 4 shows, the ARUC and loss curves demonstrate the time efficiency of MetaV in fingerprint construction.

In less than 200 seconds, MetaV stably constructs a fingerprinting pair of ResNet-18 ($N = 100$), where the x-axis shows the wall-clock time.

Figure 2: (a) The ARUC of MetaV (Ours) and baselines on Skin. (b)-(f): Curves of robustness and uniqueness of MetaV on Warfarin and Fashion, where the ARUC is reported in the figure title.

Figure 3: The robustness when the threshold is set such that the uniqueness reaches 100%.

Figure 4: The learning curves of MetaV for constructing a fingerprinting pair of ResNet-18 ($N = 100$), where the x-axis shows the wall-clock time.

Results and Analysis

- **Comparison with Baselines:**
  - MetaV is the only method achieving 100% robustness and uniqueness.
  - MetaV constructs fingerprints with the highest ARUC ($0.86 \pm 0.01$).
  - Improves baseline IPGuard by 0.59 absolutely (220% relative).

- **Time Efficiency:**
  - Stably constructs fingerprinting pairs in under 200 seconds.
Figure 5: (a)-(b): Curves of robustness and uniqueness of MetaV on Warfarin and Fashion, with the ARUC reported in the figure title. The curves of ARUC (c) when the number of fingerprint examples increases and (d) when the model ensemble is enlarged on Warfarin and Fashion.

Table 1: Comparison of the time costs for fingerprint construction and verification (sec).

|          | Construction | Verification |
|----------|--------------|--------------|
| MetaV    | 202+11       | 7.2+0.8      |
| IPGuard  | 177+1        | 9.1+0.2      |
| ModelDiff| 123+1        | 13.6+0.5     |
| DeepFoolFP| 174+4      | 9.1+0.6      |
| ConferAE | >2860        | 10.0+0.6     |

6.3 MetaV for Task-Independent Fingerprinting

Besides the substantial improvements in fingerprinting classifiers, more importantly, MetaV presents the first task-agnostic fingerprinting scheme which can be applied to more general application scenarios. To validate, we apply MetaV to fingerprint an MLP for regression (i.e., the Warfarin case) and a DCGAN for generative modeling (i.e., the Fashion case), which requires no modification on Algorithm 1, as MetaV is by design independent from either the internals or the functions of the target model.

Fig. 5(a)-(b) plot the curves of robustness and uniqueness of MetaV on Warfarin and Fashion when the threshold $\rho$ increases from 0 to 1, where the area of the shaded region is by definition the ARUC. As we can see, the robustness and the uniqueness remain 1 unless the threshold is very close to 0 or 1. This results in an over 0.98 ARUC for both the two scenarios which existing fingerprinting schemes can hardly handle.

6.4 Number of Fingerprint Examples

We further study the influence of the number of fingerprint examples on the performance of MetaV and the baselines. Fig. 6&5(c) presents the ARUC curves when the number of fingerprint examples, i.e., $N$, increases, on the classification and non-classification tasks respectively. As is shown, in all the three scenarios, the performance of MetaV increases stably when $N$ increases from 10 to 100. For example, when fingerprinting ResNet-18 on Skin, the ARUC of MetaV is about 1.2x when $N$ is enlarged from 10 to 100. This is a desirable feature of MetaV as one would naturally expect a more accurate fingerprinting when more computing power is devoted to the construction of the model fingerprints. In comparison, the upward trend is unclear for all the baseline schemes. Similarly, enhanced performance is also observed on Warfarin and Fashion by about 3.2% and 17.0% respectively, which is noticeable considering the already high ARUC of MetaV when the number of fingerprint examples is 10.

6.5 Size of Prepared Model Ensemble

Finally, we provide quantitative results to analyze the impact of the model ensemble size on the performance of MetaV. We fix the number of fingerprint examples as 100 and randomly sample different ratios of positive/negative suspect models from the full model ensemble for training MetaV. Fig. 7&5(d) shows the ARUC curves on the three scenarios when the model ensemble size varies. As is shown, the ARUC of MetaV shows a steady upward trend when the model ensemble is enlarged, which conforms to our design principle that a more diverse set of crafted suspect models would help construct more unique and robust model fingerprints.

7 CONCLUSION

In this paper, we present MetaV, the first task-agnostic model fingerprinting framework which: (a) substantially improves existing fingerprinting schemes on classification models in terms of fingerprint robustness and uniqueness, and, (b) more importantly, advances the capability of model piracy forensics to more general application scenarios. As it is a common challenge for any novel fingerprinting methods to be evaluated on large-scale datasets (e.g., for evaluation on ImageNet, one has to train over 100 suspect models on the dataset to construct the benchmark, which would incur
over 50 days of computation on medium-end devices), we design our evaluation at the same scale of all our previous works by involving 32 × 32 images only. It would be meaningful for future works to cooperate with the industry to evaluate MetaV on larger datasets. Meanwhile, although MetaV by design has no assumptions on the input, the architecture, or the output of the target model, considering the impossibility of exhausting all possible task types, we mainly choose the three representative scenarios in our paper for evaluation. Future works may consider deploy and evaluate the effectiveness of MetaV on other typical learning tasks such as feature extraction, information retrieval and ranking.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful comments that helped improve the quality of the paper. This work was supported in part by the National Key Research and Development Program (2021YFB3101200), National Natural Science Foundation of China (61972099, U1736210, U1836213, 62172104, 62172105, 61902374, 62102093, 62102091), National Science Foundation of Shanghai (19ZR1404800). Min Yang is a faculty member of Shanghai Institute of Intelligent Electronics & Systems, and Engineering Research Center of Cyber Security Auditing and Monitoring, Ministry of Education, China. Mi Zhang and Min Yang are the corresponding authors.

REFERENCES

[1] [n. d.]. PyTorch Hub. https://pytorch.org/hub/. Accessed: 2021-02-01.
[2] Y. Adi, Carsten Baum, et al. 2018. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In USENIX Security Symposium.
[3] Angela Aguinaldo, Ping-Yeh Chiang, et al. 2019. Compressing GANs using Knowledge Distillation. ArXiv abs/1902.00159 (2019).
[4] Franziska Boenisch. 2020. A Survey on Model Watermarking Neural Networks. ArXiv (2020).
[5] Xiaoyu Cao, J. Jia, et al. 2021. IPGuard: Protecting the Intellectual Property of Deep Neural Networks via Fingerprinting the Classification Boundary. AsiaCS (2021).
[6] Yulong Cao, Chaowei Xiao, et al. 2019. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. CCS (2019).
[7] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. IEEE Symposium on Security and Privacy (2017).
[8] A. Choromanska, Mikhail Henaff, et al. 2015. The Loss Surfaces of Multilayer Networks. In IJISTAT.
[9] Kevin Clark, Minh-Thang Luong, et al. 2019. BART: Born-Again Multi-Task Networks for Natural Language Understanding. In ACL.
[10] J. Devlin, Ming-Wei Chang, et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL-HLT.
[11] Andre Esteve, B. Kuprel, et al. 2017. Dermatologist-level classification of skin cancer with deep neural networks. Nature (2017).
[12] Jianping Gou, B. Yu, et al. 2021. Knowledge Distillation: A Survey. Int. J. Comput. Vis. (2021).
[13] Song Han, Jeff Pool, et al. 2015. Learning both Weights and Connections for Efficient Neural Network. ArXiv (2015).
[14] Kaiming He, X. Zhang, et al. 2016. Deep Residual Learning for Image Recognition. CVPR (2016), 770–778.
[15] B. Heaton, Nicholas G. Polson, et al. 2016. Deep Learning for Finance: Deep Portfolios. Econometric Modeling: Capital Markets - Portfolio Theory eJournal (2016).
[16] Geoffrey E. Hinton et al. 2015. Distilling the Knowledge in a Neural Network. ArXiv (2015).
[17] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017), 2261–2269.
[18] Forrest N. Iandola, M. Moskewicz, Khalid Ashraf, Song Han, W. Dally, and K. Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. ArXiv abs/1602.07360 (2016).
[19] Hoyoung Jeong, Dohyun Ryu, et al. 2021. Neural Network Stealing via Meltdown. KON (2021), 36–38.
[20] Mika Juuti, Sebastian Szyller, et al. 2019. PRADA: Protecting Against DNN Model Stealing Attacks. EuroS&P (2019).
[21] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. CoRR abs/1412.6980 (2015).
[22] A. Krizhevsky. 2014. One weird trick for parallelizing convolutional neural networks. ArXiv abs/1404.5997 (2014).
[23] Hao Li, Amin Kadav, et al. 2017. Pruning Filters for Efficient ConvNets. ArXiv (2017).
[24] Yuanchun Li, Ziqi Zhang, et al. 2021. ModelDiff: testing-based DNN similarity comparison for model reuse detection. IJISTAT (2021).
[25] Nils Lukas, Yuxuan Zhang, et al. 2021. Deep Neural Network Fingerprinting by Conferable Adversarial Examples. ICPR (2021).
[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, et al. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. CVPR (2016).
[27] Adam Paszke, S. Gross, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In NeurIPS.
[28] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR abs/1511.06434 (2016).
[29] Esteban Real, A. Aggarwal, et al. 2019. Regularized Evolution for Image Classifier Architecture Search. In AAAI.
[30] F. Regazzoni, P. Palmieri, et al. 2021. Protecting artificial intelligence IP’s: a survey of watermarking and fingerprinting for machine learning. CAAI Transactions on Intelligence Technology (2021).
[31] B. Rouhani, Hulii Chen, et al. 2018. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. ArXiv (2018).
[32] K. Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv (2015).
[33] Christian Szegedy, W. Zaremba, Ilya Sutskever, Joan Bruna, et al. 2014. Intriguing properties of neural networks. ArXiv (2014).
[34] Florian Tramèr, F. Zhang, et al. 2016. Stealing Machine Learning Models via Prediction APIs. In USENIX Security.
[35] G. Truda and P. Marais. 2019. Warfarin dose estimation on multiple datasets with automated hyperparameter optimisation and a novel software framework. ArXiv (2019).
[36] Y. Uchida, Yuki Nagai, et al. 2017. Embedding Watermarks into Deep Neural Networks. ICIRM (2017).
[37] Si Wang and Chip-Hong Chang. 2021. Fingerprinting Deep Neural Networks - a DeepFool Approach. ISCAS (2021).
[38] M. Whirl-Carrillo, E. McDonagh, et al. 2012. Pharmacogenomics Knowledge for Personalized Medicine. Clinical Pharmacology & Therapeutics (2012).
[39] H. Xiao, K. Rasul, et al. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. ArXiv (2017).
[40] Xiangrui Xu, Y. Li, et al. 2020. “Identity Bracelets” for Deep Neural Networks. IEEE Access (2020).
[41] Mengjia Yan, Christopher W. Fletcher, and J. Torrellas. 2020. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. USENIX Security (2020).
[42] Jiancheng Yang, R. Shi, et al. 2020. MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis. ArXiv (2020).
[43] Honggang Yu, Kaichen Yang, et al. 2020. CloudLeak: Large-Scale Deep Learning Models Stealing Through Adversarial Examples. In NDSS.
[44] Jialong Zhang, Zhongdu Gu, et al. 2018. Protecting intellectual property of deep neural networks with watermarking. AsiaCS (2018).
[45] Jingting Zhao, Qingyu Hu, et al. 2020. AFA: Adversarial fingerprinting authentication for deep neural networks. Comput. Commun. (2020).
[46] Haoyi Zhou, Shanghang Zhang, et al. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In AAAI.
A MORE TECHNICAL DETAILS

Table 2: Learning tasks and datasets in the evaluation.

| Identifier | Type       | Dataset       | Target Model |
|------------|------------|---------------|--------------|
| Skin       | Classification | DermaMNIST   | ResNet-18    |
| Warfarin   | Regression  | IWPC Dataset  | MLP          |
| Fashion    | Generative Modeling | FashionMNIST | DCGAN        |

Table 3: Composition of suspect models for each scenario.

| Positive Suspect Models | Skin | Warfarin | Fashion |
|-------------------------|------|----------|---------|
| Fine-tuning             | FTLL | 2        | 2       |
|                         | FTAL | 2        | 2       |
| Partial Retraining      | RTLL | 2        | 2       |
|                         | RTAL | 2        | 2       |
| WP                      | 0,1,2,...,8,9 | 9 × 2     | 9 × 4   |
| FP                      | 1/16,1/8,...,15/16 | 15 × 2   | N/A     |

Table 4: The detailed architecture of DCGAN on Fashion, which is described by convention of PyTorch.

| Generator | Discriminator |
|-----------|---------------|
| nn.ConvTranspose2d(100, 128, 4, 1, 0, bias=False) | nn.Conv2d(128, 128, 4, 1, 0, bias=False) |
| nn.BatchNorm2d(128) | nn.LeakyReLU(0.2) |
| nn.ReLU() | nn.ReLU(0.2) |
| nn.Conv2d(128, 64, 4, 2, 1, bias=False) | nn.Conv2d(64, 128, 4, 2, 1, bias=False) |
| nn.LeakyReLU(0.2) | nn.LeakyReLU(0.2) |
| nn.Conv2d(64, 32, 4, 2, 1, bias=False) | nn.Conv2d(32, 64, 4, 2, 1, bias=False) |
| nn.BatchNorm2d(64) | nn.BatchNorm2d(32) |
| nn.LeakyReLU(0.2) | nn.LeakyReLU(0.2) |
| nn.Conv2d(32, 1, 4, 1, 0, bias=False) | nn.Conv2d(1, 32, 4, 2, 1, bias=False) |
| nn.LeakyReLU(0.2) | nn.ReLU() |

Algorithm 1 The algorithmic details of MetaV’s fingerprint construction stage.

1: **Input:** A prepared model ensemble $M \cup \{F\} \cup M_\ell$, the number of adaptive fingerprints $N$, the input/output dimension of the models $d_{in}, d_{out}$, the number of iterations $L$ and the learning rate $\lambda$.
2: **Output:** The optimal fingerprinting pair $(X^*_F, X^*_\ell)$.
3: Initialize real-valued variables $W_0 = (w_i)_{i=1}^N$.
4: Initialize a meta-verifier $V(\cdot ; \Theta_0) : \mathbb{Y}^{N \times d_{out}} \rightarrow \mathbb{S}^2$ with parameters $\Theta_0$ ($\Theta_0$)
5: We implement $V$ as a fully-connected neural network with a softmax output.
6: Initialize Adam optimizers $\text{Opt}_W$, $\text{Opt}_\Theta$ of a learning rate $\lambda$.
7: for $i \in \{0, \ldots, L - 1\}$ do
8:   Sample a tuple of models $(M_\ell, F, M_\ell)$ from $M_\ell \cup \{F\} \cup M_\ell$ respectively.
9:   For each $i \in [N]$, calculate $x^i_F = \tanh(w_i)$.
10: $\ell(W_i, \Theta_i) = \log p_e(M_\ell) + \log p_e(F) + \log p_e(M_\ell) \rightarrow (p_e(M), p_e(M)) = V(M(x^i_F) \oplus \ldots \oplus M(x^N_F))$
11: $W_{i+1} \leftarrow \text{Opt}_W (\ell, W_i)$,
12: $\Theta_{i+1} \leftarrow \text{Opt}_\Theta (\ell, \Theta_i)$.
13: end for
14: For each $i \in [N]$, $x^i_{F*} = \tanh(w^i_F)$.
15: $\Theta^* \leftarrow \Theta^L$.
16: **Return:** $(X^*_F, V(\cdot ; \Theta^*))$