Fingerprinting Deep Neural Networks Globally via Universal Adversarial Perturbations

Zirui Peng[1]* Shaofeng Li[1]* Guoxing Chen[1,✉] Cheng Zhang[2] Haojin Zhu[1,✉] Minhui Xue[3,4]

[1] Shanghai Jiao Tong University [2] The Ohio State University [3] CSIRO's Data61 [4] The University of Adelaide

{pengzirui,shaofengli,guoxingchen,zhu-hj}@sjtu.edu.cn, zhang.7804@osu.edu, jason.xue@adelaide.edu.au

Abstract

In this paper, we propose a novel and practical mechanism to enable the service provider to verify whether a suspect model is stolen from the victim model via model extraction attacks. Our key insight is that the profile of a DNN model's decision boundary can be uniquely characterized by its Universal Adversarial Perturbations (UAPs). *UAPs belong to a low-dimensional subspace and piracy models' subspaces are more consistent with victim model's subspace compared with non-piracy model. Based on this, we propose a UAP fingerprinting method for DNN models and train an encoder via contrastive learning that takes fingerprints as inputs, outputs a similarity score. Extensive studies show that our framework can detect model Intellectual Property (IP) breaches with confidence $>$ 99.99 % within only 20 fingerprints of the suspect model. It also has good generalizability across different model architectures and is robust against post-modifications on stolen models.*

1. Introduction

In the past few years, deep learning has emerged as a promising approach and the foundation for a wide range of real-world applications. As network architecture becomes more and more sophisticated and training costs rise, well-trained models become lucrative targets for the adversary looking to "steal" them. By querying the publicly available APIs of these models, an adversary can collect the outputs to train a piracy model, dubbed *model extraction* attacks [4, 6, 7, 16, 31, 37, 42].

Existing works on mitigating model extraction attacks and protecting the Intellectual Property (IP) of trained models fall into two groups [8, 18]. The first group is based on *watermarking* techniques [2, 14, 17, 26, 29, 32, 35]. The idea is that the model owner introduces into her IP model a backdoor (*i.e.*, a watermark), which would persist during the model extraction. By checking whether a suspect model contains the injected watermark, a defender can determine whether this model is pirated.

The other category is based on *fingerprinting* techniques that leverage inherent information (*i.e.*, *decision boundary*) of the trained models. Observing that a DNN model can be uniquely profiled by its decision boundary which is also likely to be inherited by piracy models, a model extraction attack can be identified by examining whether a suspect model has (almost) the same decision boundary as the victim model. A line of research [5, 13, 24] adopts adversarial examples to represent the decision boundaries.

However, the effectiveness of existing mitigation schemes were challenged. Watermarking based solutions suffer from utility drop caused by watermarks. Another concern is that an attacker can illegally inject a backdoor to argue the ownership which violates the non-forgeable demand [17]. Adversarial examples can only capture the *local geometry*, particularly, orientations of the decision boundary in local regions surrounding the adversarial examples, which may fail to be transferred to the suspect model due to decision boundary variation during the extraction [21, 30].

In this paper, we explore methods to capture the global characteristics of the decision boundary. As demonstrated in Figure 1, we propose a more effective model extraction detection scheme based on *Universal Adversarial Perturbations* (UAPs) [27]. A carefully selected UAP vector $\mathbf{v}$ can fool the model on almost all datapoints. We find that UAPs are drawn from a low-dimensional subspace that contains most of the normal vectors of decision boundary. Due to decision boundary dependency, UAP subspaces of piracy models are more consistent with that of the victim model, which enables us to give a similarity score.

There are two challenges in applying UAPs for detecting model extraction. First, since the calculation of UAPs usually requires the knowledge of model parameters (*i.e.*, white-box access) which model owners are not willing to provide, it is intractable for the defender to obtain UAPs of suspect models via black-box access. The second challenge

*Equal contribution.

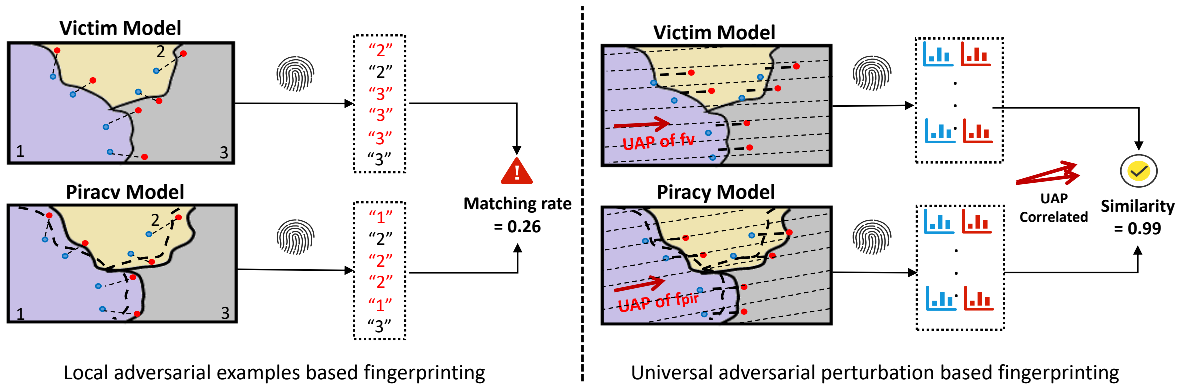

Victim Model
Piracy Model
Matching rate = 0.26
Local adversarial examples based fingerprinting
UAP of fv
UAP of fpir
UAP Correlated
Similarity = 0.99
Universal adversarial perturbation based fingerprinting

Figure 1. **Illustrations of local and universal adversarial perturbations.** Left: local adversarial perturbations are less robust to point-to-point decision boundary modification due to extraction. Right: our framework relies on the stable correlation of decision boundaries profiled by universal adversarial perturbations (UAPs).

is how to reliably distinguish between a piracy model and a homologous model (*i.e.*, model trained on the same training data rather than the victim model's outputs and should be not considered "stolen").

To tackle the first challenge, we propose a fingerprinting function which is obtained by querying the suspect model with a number of datapoints, added by victim's UAPs. A more informative fingerprints need to capture as many parts of decision boundaries as possible. We therefore adopt K-means clustering on the last layer of the victim model to ensure that the datapoints are uniformly selected from different source classes and move towards different target classes.

To address the second challenge, we design an encoder to map fingerprints of the victim model, piracy models, and homologous models into a joint representation space. We adopt *contrastive learning* [9] (which aims to shorten the distances of the samples in the same classes and push away any samples of other classes) to project homologous models farther away from the victim model than the piracy models.

In summary, we propose a more accurate, robust and general IP protection framework against the model extraction attacks. Our main contributions are:

- We present one of the first attempts to leverage UAP distribution dependency to measure the decision boundary similarity between models. We show that UAP outperforms adversarial perturbation for model fingerprinting.
- We propose a novel model ownership verification framework based on UAP fingerprinting that achieves a highly competitive detection rate in terms of AUC.
- Compared with prior fingerprinting works, we demonstrate the capability of our framework for detecting post-modificated piracy models.
- We adopt contrastive learning in encoder training to address the similarity gap between homologous models and piracy models. A new data augmentation approach is proposed to create "views" for fingerprints.

2. Background and Related Work

Model extraction violates the confidentiality of machine learning models [6, 7, 16, 19, 37]. In a model extraction attack, the attacker only has black-box access to a victim model and aims at stealing it through posing queries. The obtained model is expected to be functionally similar. To extract a model, the attacker first needs to collect a set of unlabeled natural data. Then the natural data are mixed with carefully crafted synthesized data to query the victim model. The returned labels are then used to train a piracy model. This process repeats for several iterations until the piracy model recovers a satisfying utility of the victim.

Model fingerprinting relies on finding existing features that characterises the model. Recent works rely on the different transferabilities of the victim model's adversarial examples [3, 10, 23, 25, 36, 40] on piracy models and independent models. They are widely used to solve problems like model modifications [13] and model extraction attacks [5, 24]. Cao *et al.* [5] present an adversarial example based algorithm to generate datapoints near the decision boundary and utilize the transferability gap of those data to identify the piracy models. However, the performance of this work is not stable across different model architectures. Lukas *et al.* [24] also adopt the transferability to craft synthesized datapoints named "conferrable examples" that only transfer to piracy models instead of homologous models. Crafting conferrable examples involves training up to 30 models and then backpropogating through them to obtain a gradient update. This leads to a huge overhead cost.

3. Problem Formulation

3.1. Model Definition

Now we formally define the IP protected DNN models and the *piracy models*. Consider a problem domain denoted by $\mathcal{X} \subset \mathbb{R}^M$. Each element $\mathbf{x} \in \mathcal{X}$ is labeled by one of N

classes, say i-th class, denoted by a one-hot vector $l(\mathbf{x}) \in \mathbb{R}^N$. A DNN model is a function $f : \mathbb{R}^M \rightarrow \mathbb{R}^N$ which takes as input $\mathbf{x} \in \mathbb{R}^M$, and outputs a vector $f(\mathbf{x}) \in \mathbb{R}^N$ with i-th entry $f(\mathbf{x})_i$ denoting the model's confidence that $\mathbf{x}$ is from the i-th class.

Definition 1 *(IP Protected DNN Model). A DNN model owned by a model owner u is denoted by $f_{\mathcal{V},u} : \mathbb{R}^M \rightarrow \mathbb{R}^N$. It is trained by the model owners on their dataset $D_{\mathcal{V},u} \subset \{(\mathbf{x}, l(\mathbf{x})) \mid \mathbf{x} \in \mathcal{X}\}$, aiming to optimize*

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{X}}(\arg\max_k(f_{\mathcal{V},u}(\mathbf{x})_k) = i \wedge l(\mathbf{x})_i = 1). \quad (1)$$

Two different model owners, say u and v, might have slightly different training datasets, model structures, training processes, their trained models ($f_{\mathcal{V},u}$ and $f_{\mathcal{V},v}$) are highly similar but considered independent, we refer to as *homologous models*. In contrast, in model extraction attacks, an adversary may query a victim model $f_{\mathcal{V},u}$ with her chosen inputs and obtains the model outputs which can be used as labels to train a model. We refer to it as *piracy model* and give a formal definition as follows.

Definition 2 *(Piracy Model). A piracy model obtained by an attacker who launches model extraction attacks on a victim model $f_{\mathcal{V},u}$ is denoted by $f_{\mathcal{P},u} : \mathbb{R}^M \rightarrow \mathbb{R}^N$. It is trained on her dataset $D_{\mathcal{P},u} \subset \{(\mathbf{x}, f_{\mathcal{V},u}(\mathbf{x})) \mid \mathbf{x} \in \mathcal{X}\}$, aiming to optimize*

$$\mathbb{P}_{\mathbf{x}\sim\mathcal{X}}(\arg\max_k(f_{\mathcal{P},u}(\mathbf{x})_k) = \arg\max_k(f_{\mathcal{V},u}(\mathbf{x})_k). \quad (2)$$

3.2. Threat Model

The threat model considered in this paper involves a model owner who deploys its trained model $f_{\mathcal{V},u}$ as a cloud service and an adversary who tries to launch model extraction attacks against $f_{\mathcal{V},u}$ and deploys the piracy model $f_{\mathcal{P},u}$ for financial benefits. The model owner here acts as both the *victim* and the *defender* who aims to verify whether a suspect model $f_{\mathcal{S}}$ is a piracy or homologous model of $f_{\mathcal{V},u}$.

Attacker's Capability and Knowledge. The attacker has black-box access to a victim model $f_{\mathcal{V},u}$ and knows its problem domain. To evade potential model extraction detection schemes, the adversary may also apply various modifications (*e.g.*, fine-tuning, compression, pruning and adversarial training) to piracy models.

Defender's Capabilities and Knowledge. The defender (victim) has white-box access to its model $f_{\mathcal{V},u}$ (*i.e.*, model parameters, hyper-parameters, and training dataset) and black-box access to a suspected model $f_{\mathcal{S}}$ with a limited number of queries. Specifically, the defender has no knowledge about the suspect model's architecture, parameters, hyper-parameters, nor the attacker's data used during the extraction.

3.3. Design Overview

In this paper, we propose to use UAPs to capture the global geometric information of a DNN model's decison boundary for model extraction detection. Universal Adversarial Perturbation suggests that a carefully selected perturbation vector $\mathbf{v} \in \mathbb{R}^M$ of a model f can fool the model on almost all datapoints drawn from the problem domain $\mathcal{X} \subset \mathbb{R}^M$. Formally, a UAP $\mathbf{v}$ is (ξ, δ)-universal such that

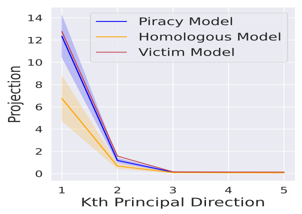
$$\mathbb{P}_{\mathbf{x}\sim\mathcal{X}}(\arg\max_k f(\mathbf{x}+\mathbf{v})_k \neq \arg\max_{k'} f(\mathbf{x})_{k'}) \geq 1-\delta, \quad s.t., \ \ ||\mathbf{v}||_2 \leq \xi \quad (3)$$

where $f(\cdot)$ is the output probability vector. A UAP $\mathbf{v}$ can be viewed as a natural defect of the model f which exposes the geometric correlations between different local gradients of the model's decision boundary. In fact, for a given model, there are a bunch of UAPs that exist in a low-dimensional subspace in which most of the normal vectors of decision boundaries lie, as pointed by Moosavi-Deafooli *et al.* [27].

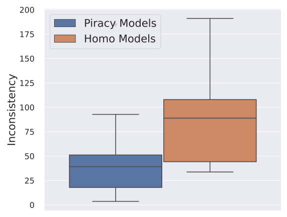
The UAP subspaces of two homologous models are independent since the decision boundaries are formed via two independent training processes. In contrast, Papernot *et al.* [30] use chi-square test to statistically verify that datapoints' gradients of piracy models are dependent on those of the victim model. We observe that this dependency is maintained by UAPs. We postpone the detail of this observation to Section 4.1 and continue our design overview. Since calculating a UAP requires white-box access to the model, it is infeasible for the defender to obtain the UAPs of suspect model $f_{\mathcal{S}}$ and compare their similarity with $f_{\mathcal{V},u}$. Alternatively, with white-box access to the victim model $f_{\mathcal{V},u}$, the defender (victim) could generate a UAP $\mathbf{v}$ and verify whether $\mathbf{v}$ lies in the UAP subspace of the suspect model $f_{\mathcal{S}}$. Specifically, we propose the following two primitives for the verification:

Fingerprint Generation. To verify whether a vector $\mathbf{v}$ lies in the UAP subspace of a model f, we propose to design a fingerprints generation function $\mathcal{F}$ which captures the *fingerprints* of how the model f behaves around n datapoints $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with regards to $\mathbf{v}$, denoted by $\mathcal{F}(f, \mathbf{v}, [\mathbf{x}_1, \ldots, \mathbf{x}_n])$.

Fingerprint Verification. With fingerprints of both the victim model and the suspect model, the defender needs to determine whether the suspect model is a piracy model or a homologous model. Particularly, we propose to design an encoder E_θ with parameters θ to map the fingerprints of models to latent space, such that the mapped fingerprints of a victim model and its piracy model have a large similarity (*e.g.*, cosine similarity) while the mapped fingerprints of a victim model and a homologous model has a small similar-



(a) SVD on DNN models' UAPs.



(b) Inconsistency distributions.

Figure 2. (a) Projections of $f_{\mathcal{V},u}$, piracy and homologous models on top-5 principal directions of $f_{\mathcal{V},u}$, the dark line indicates mean value over 20 models and light intervals indicate STD; (b) Inconsistency distribution of piracy models and homologous models calculated according to Eq. 5.

ity. Particularly, the encoder aims to optimize:

$$\begin{aligned}&\max_{\theta}\ \mathbb{E}(sim(f_{\mathcal{V},u},f_{\mathcal{P},u})) - \mathbb{E}(sim(f_{\mathcal{V},u}, f_{\mathcal{V},v})),\\ &\text{where}\ \ sim(f_a, f_b) = cosine(E_\theta(\mathcal{F}_a), E_\theta(\mathcal{F}_b))\\ &\qquad\qquad \mathcal{F}_a = \mathcal{F}(f_a, \mathbf{v}, [\mathbf{x}_1, \ldots, \mathbf{x}_n])\\ &\qquad\qquad \mathcal{F}_b = \mathcal{F}(f_b, \mathbf{v}, [\mathbf{x}_1, \ldots, \mathbf{x}_n]).\end{aligned} \tag{4}$$

4. UAP based Fingerprinting

In this section, we first explain the observation on which our design is based. Then we introduce the design of fingerprint generation and fingerprint verification.

4.1. Observation Explanation

We first show the relation among UAP subspaces of the victim model, homologous models, and piracy models.

OBSERVATION: *A victim model's UAP subspace is consistent with its piracy model's UAP subspace and is inconsistent with a homologous model's UAP subspace.*

We model the dependency of a suspect model $f_{\mathcal{S}}$ on a victim model $f_{\mathcal{V},u}$ as their *consistency*, which is defined as ℓ_2 distance between the projections of these two models' UAPs on a set of orthogonal basis formed by principal directions of $f_{\mathcal{V},u}$'s UAP matrix. To obtain this basis, we perform singular-value decomposition on $f_{\mathcal{V},u}$'s UAP matrix and choose its right singular vector basis.

Let $V_{f_{\mathcal{S}}} = \{\mathbf{v}^1_{f_{\mathcal{S}}}, \ldots, \mathbf{v}^L_{f_{\mathcal{S}}}\}$ be UAPs of the suspect model $f_{\mathcal{S}}$. $V_{f_{\mathcal{V},u}} = \{\mathbf{v}^1_{f_{\mathcal{V},u}}, \ldots, \mathbf{v}^L_{f_{\mathcal{V},u}}\}$ be UAPs of the victim model. After performing SVD on $V_{f_{\mathcal{V},u}}$, there is a r-dimensional orthogonal basis $\{v_1, v_2, \cdots, v_r\}$, where r is the rank of $V_{f_{\mathcal{V},u}}$. We define the distribution inconsistency of UAPs between $V_{f_{\mathcal{S}}}$ and $V_{f_{\mathcal{V},u}}$ as $Inconsist_{f_{\mathcal{V},u}}(f_{\mathcal{S}})$:

$$\textstyle\sum_{0\leq m\leq r}\left(\sum_{0\leq i\leq L}(\mathbf{v}^i_{f_{\mathcal{S}}}\cdot v_m)^2 - \sum_{0\leq j\leq L}(\mathbf{v}^i_{f_{\mathcal{V},u}}\cdot v_m)^2\right)^2. \tag{5}$$

We generate m (input dimension) UAPs for each model on FMNIST dataset to form a square UAP matrix with its rank equals m. Figure 2a shows that the piracy models' UAPs has similar projections as the victim, whereas the homologous models loosely follow principal directions. Figure 2b shows that $Inconsist_{f_{\mathcal{V},u}}(f_{\mathcal{P},u})$ is 3 times smaller than $Inconsist_{f_{\mathcal{V},u}}(f_{\mathcal{V},v})$ on FMNIST dataset. We conclude that UAPs of piracy and homologous models differ and can be used to differentiate these two types of models.

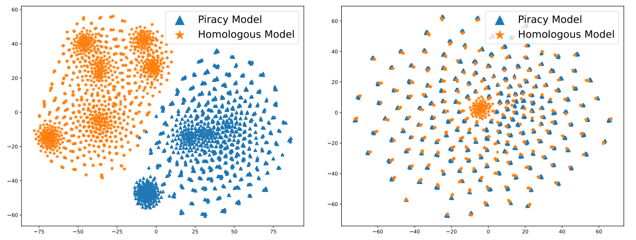


Figure 3. t-SNE visualization of fingerprints. UAP-based fingerprints (left) are naturally distinguishable compared with local adversarial perturbation based fingerprints (right). (FMNIST)

4.2. Fingerprint Generation

We now define a fingerprint generation function as described in Section 3.3 as follows:

$$\begin{aligned}&\mathcal{F}(f, \mathbf{v}, (\mathbf{x}_1, \cdots, \mathbf{x}_n))\\ =&[f(\mathbf{x}_1), f(\mathbf{x}_1+\mathbf{v}), \cdots, f(\mathbf{x}_n), f(\mathbf{x}_n+\mathbf{v})].\end{aligned} \tag{6}$$

The goal of $\mathcal{F}$ is to capture how the given model's outputs change around the given datapoints before and after adding the UAP. Intuitively, if $\mathbf{v}$ is a UAP of $f_{\mathcal{S}}$, adding $\mathbf{v}$ to samples will significantly reduce the confidence of $f_{\mathcal{S}}$ on its original predict class. Otherwise, adding $\mathbf{v}$ will not push samples to a decision boundary.

The next step is to select n datapoints that could better profile the decision boundary. Besides using more datapoints (*i.e.*, larger n), we would like the datapoints spread uniformly throughout entire decision boundaries to capture diverse information. Thus, we perform K-means to cluster all training datapoints of the victim model into n clusters according to their representation vector in the last layer of the model, and select one datapoint from each cluster.

We compare the effectiveness of UAP based fingerprints to local adversarial perturbation(LAP) based fingerprints of three types of models (the victim, piracy and homologous). As shown in Figure 3, for UAP based fingerprints, models with the same type form one cluster, which is distant from the clusters of models with different types. In contrast, LAP based fingerprints of models with different types mix together and are indistinguishable. Please be referred to supplementary materials for more discussions.

4.3. Fingerprint Verification

We leverage an encoder to learn knowledge contained in fingerprints and output a human-comprehensible similarity score. The encoder projects the features of fingerprints into

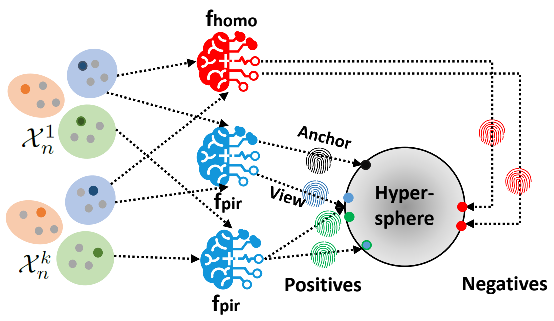


Figure 4. Illustration of contrastive learning. $\mathcal{X}_n^1 \cdots \mathcal{X}_n^k$ are k sets of datapoints used to create "views" (blue fingerprint is an augmented view for black one) . Piracy fingerprints are positive with each other (green) and negative with homologous fingerprints (red). Encoder projects fingerprints to a hyper-sphere.

a latent space and one can easily compares two fingerprints by their representations' cosine similarity in the space.

Simply training an encoder (*e.g.*, AutoEncoder) can only extracts common features of piracy fingerprints and other fingerprints that have different features will not be mapped near to embedding space. As homologous models are highly similar to the victim model, it fails to project them away from piracy one. We leverage the supervised contrastive learning [20] to emphasize such differentiation on homologous models.

Precisely, we assign label 0 to victim and piracy fingerprints, label 1 to homologous fingerprints. As demonstrated in Figure 4, the encoder projects positive pairs onto the same part on the hyper-sphere (left side) and projects negative pairs onto the opposite part (right side). For self-supervised contrastive learning, a positive pair $(\mathbf{x}, \tilde{\mathbf{x}})$ refers to an input $\mathbf{x}$ and its view $\tilde{\mathbf{x}}$, all other inputs and their views are negative to $\mathbf{x}$. For supervised contrastive learning, the positive pairs are all inputs having the same label as $\mathbf{x}$ and their views, negative pairs are all other inputs and their views.

To generate positive pairs for a given fingerprint in contrastive learning, we propose a novel data augmentation strategy as follows.

Multi-views Fingerprints Augmentation. We denote $\mathcal{X}_n$ is the n datapoints selected from n different clusters. For each datapoint $\mathbf{x}_i$ in $\mathcal{X}_n$, we choose its k nearest neighbors to form $\mathcal{K}_i$ according to their representations in the output layer. We perform sampling without replacement in each cluster $\mathcal{K}_i$ and obtain k sets of datapoints, denoted by $\mathcal{X}_n^1, \cdots, \mathcal{X}_n^k$ (See Figure 4 for an illustration). In this way, we further generate k positive views $\{\mathcal{F}(f, \mathbf{v}, \mathcal{X}_n^1), \cdots, \mathcal{F}(f, \mathbf{v}, \mathcal{X}_n^k)\}$ for the given fingerprint $\mathcal{F}(f, \mathbf{v}, \mathcal{X}_n)$. Please see supplementary materials for more details, including the evidence that positive views are the most similar fingerprints among others.

Supervised Contrastive Loss. We now describe our supervised contrastive loss as follows. Within a multiviewed batch, let $i \in I \equiv \{1, ..., k_N\}$ be batch index and $C(i) = I \backslash \{i\}$. Let $\Psi(i) := \{\mu \in C(i) | \tilde{y_\mu} = y_i\}$ be indexes of positive pairs for the i-th sample. Then our supervised contrastive loss is:

$$\mathcal{L} = \sum_{i \in I} -\frac{1}{|\Psi(i)|} \sum_{\mu \in \Psi(i)} \log \frac{e^{sim(\mathbf{z}_i, \mathbf{z}_\mu)/\tau}}{\sum_{\nu \in C(i)} e^{sim(\mathbf{z}_i, \mathbf{z}_\nu)/\tau}}, \quad (7)$$

where N is the size of mini-batch. The encoder consists of two parallel networks, which is similar to SimCLR [9] The overall procedure of model extraction detection is represented by algorithm 1.

Algorithm 1: Ownership Verification.

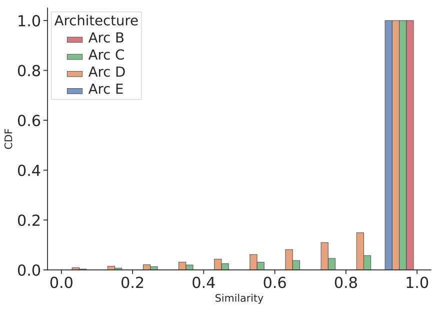
```
Input: suspect model $f_\mathcal{S}$, victim's model $f_{\mathcal{V},u}$, it's UAP $\mathbf{v}$ and
       training data $\mathcal{D}$, number of clusters $n$, number of
       fingerprints views $k$, a set of piracy models $\Phi$ and
       homologous models $\Upsilon$, batch size $N$ and loss function $\mathcal{L}$
       for constrastive learning in Eq. 7.
Output: Trained encoder $E_\theta$, similarity $\mathbf{s}$ between $f_\mathcal{S}$ and $f_{\mathcal{V},u}$.
/* Fingerprints Generation */
1  Function $\mathcal{F}(f, \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}, \mathbf{v})$:
2      $\mathcal{X} \leftarrow \{\}$
3      for $i \in \{1, .., n\}$ do
4          $t_i = f(\mathbf{x}_i) \oplus f(\mathbf{x}_i + \mathbf{v})$
5          $\mathcal{X} = \mathcal{X} \cup \{t_i\}$
6      end
7      return $\mathcal{X}$
/* Preparing trainset for encoder $E$ */
8  $B \leftarrow \{\}$;  $\mathcal{M} \leftarrow \{f_{\mathcal{V},u}\} \cup \Phi \cup \Upsilon$
9  $\{C_1, C_2, \cdots, C_n\} =$ K-Means$(f_{f_{\mathcal{V},u}}, \mathcal{D}, n)$
10 for $f \in \mathcal{M}$ do
       /* Sample one point from each cluster
          without replacement */
11     for $i \in \{1, ..., k\}$ do
12         $\{\mathbf{x}_1, \cdots, \mathbf{x}_n\}^i \leftarrow C_1 \times C_2 \cdots \times C_n$
13         $B = B \cup \mathcal{F}(f, \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}^i, \mathbf{v})$
14     end
15 end
/* Training by contrastive loss */
16 Initial parameters $\theta$ of the encoder $E$
17 $E_\theta$ ←Training$(B, \mathcal{L})$
/* Verifying suspect model */
18 $\mathbf{s} = cosine(E_\theta(f_\mathcal{S}), E_\theta(f_{\mathcal{V},u}))$
19 return $\mathbf{s}$
```

5. Experiments

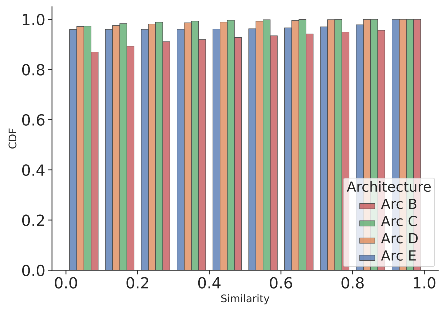
5.1. Setup

Datasets. We evaluate our approach on three popular image classification datasets: FashionMNIST (FMNIST) [38], CIFAR-10 [22], and TinyImageNet [1].

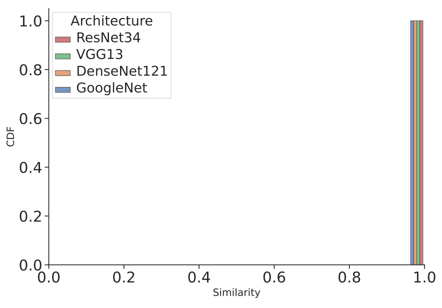
Model Architectures. For FMNIST, SOTA classification accuracy can be achieved using simple CNN models. To ensure the diversity of models, we vary attributes such as kernel size, number of layers, activation



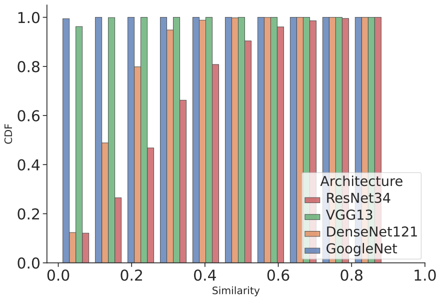
(a) Sim CDF between $f_{\mathcal{V},u}$ and $f_{\mathcal{P},u}$(FMNIST)



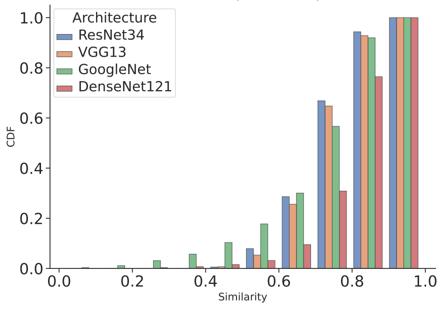
(b) Sim CDF between $f_{\mathcal{V},u}$and $f_{\mathcal{V},v}$(FMNIST)



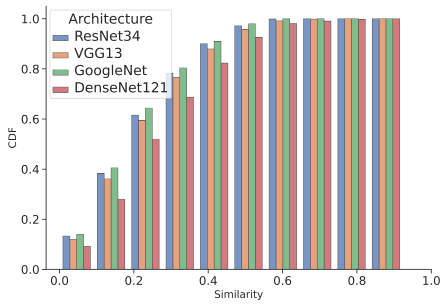
(c) Sim CDF between $f_{\mathcal{V},u}$and $f_{\mathcal{P},u}$(CIFAR10)



(d) Sim CDF between $f_{\mathcal{V},u}$and $f_{\mathcal{V},v}$(CIFAR10)



(e) Sim CDF between $f_{\mathcal{V},u}$ and $f_{\mathcal{P},u}$(T-ImageNet)



(f) Sim CDF between $f_{\mathcal{V},u}$ and $f_{\mathcal{V},v}$ (T-ImageNet).

Figure 5. Cumulative distribution function (CDF) of similarities between fingerprints of $f_{\mathcal{V}}$ and suspect models. For x in X-axis, the Y-axis is the percentage of fingerprints that has similarity smaller than x. The derivation of CDF is the probability density function.

Table 1. FashionMNIST classifier components.

	Attribute	Value
Architecture	Activation	ReLU[3,5], PReLU[4], ELU[1,2]
	Dropout	Yes[5], No[1,2,3,4]
	Conv ker size	3[1,3,4], 5[2,5]
	#Conv layers	2[1,3,5], 3[2], 4[4]
Optimziation	Algorithm	SGD, ADAM, RMSprop
	Batch size	64,128,256

functions, dropout, training batchsize and optimizer. Details are shown in Table 1. We regroup these attributes into 5 different model architectures. The numbers inside brackets indicate the attributes is assigned to which architectures. Attributes without numbers are randomly selected during each training process. For CIFAR-10 and TinyImageNet, we evaluate our encoder in 5 different architectures: ResNet18 and ResNet34 [12], VGG16 [33], DenseNet121 [15], GoogLeNet [34].

Model Preparation. For each dataset, we only assign half of the train set D as the victim model's trainset, denoted as D_v. Piracy models are generated by following the extraction attack in [39]. All generated piracy models recover $85\%, 83\%, 40\%$ performance of $f_{\mathcal{V},v}$ for FMNIST, CIFAR10 and TinyImageNet, respectively. Homologous models are trained on D_{homo} which is sampled from D and has equal size as D_v, overlapping with it.

To avoid contingency, we generate 10 models for each type (piracy or homologous) and architecture (except for architecture reserved for the victim model). In total we train 81 DNN models for CIFAR-10 and TinyImageNet respectively. We take both 4 model architectures and 3 optimizers into account and generate 241 models. All models achieve SOTA performance. The UAP used in this work achieves $> 80\%$ attack success rate. Please see more results and analyses in supplementary materials.

Encoder Training. Excluding the architecture of $f_{\mathcal{V},v}$, we use the remaining architectures to train other models. We use 5 piracy models and 5 homologous models to train the encoder. The rest of the models are used to test the encoder. For three datasets, each fingerprint is consists of 100 datapoints and we generate 200 views for each fingerprint. For encoder training, we adopt a large batch size equals 512 as recommended in the contrastive learning [9]. Note that, in experiments, to demonstrate the generalizability and robustness of our approach, we generate lots of models to train and test our framework. In practice, a defender can safely claim its ownership with only 10 models and 20 fingerprints.

Evaluation Metrics. The similarity between two fingerprints are defined as the cosine similarity of their representation vectors projected by the trained encoder. The similarity between two models is the average similarities on all generated fingerprints.

5.2. Fingerprint Identification and Matching

Figure 5 reports the results of similarity for piracy and homologous fingerprints with different network architectures. The CDF of similarity distribution is calculated during each 0.1 interval. For x in X-axis, the Y-axis is the

Table 2. Mean and STD of model similarities between the victim model and suspect model and p-value (lower is better) using 20 fingerprints (FMNIST).

		Piracy Model			Homologous Model		
Architecture & Optimizer		Mean	STD	P value	Mean	STD	P value
Arc B	SGD	0.9990	0.0012	0.0	10ˆ(-5)	0.0009	1.0
	Adam	0.9974	0.0048	0.0	0.0858	0.2613	0.9167
	Rmsprop	0.9964	0.0060	0.0	0.0002	0.0015	1.0
Arc C	SGD	0.9322	0.1882	10ˆ(-15)	0.0009	0.0149	1.0
	Adam	0.9959	0.0050	0.0	0.0185	0.0842	1.0
	Rmsprop	0.9734	0.0927	0.0	0.0074	0.0417	1.0
Arc D	SGD	0.9980	0.0027	0.0	0.1082	0.3010	0.8445
	Adam	0.9972	0.0036	0.0	0.1024	0.2492	0.5663
	Rmsprop	0.9977	0.0030	0.0	0.0295	0.0938	0.6942
Arc E	SGD	0.8821	0.203	10ˆ(-16)	0.0377	0.1360	0.9923
	Adam	0.9515	0.1392	0.0	0.0010	0.0036	1.0
	Rmsprop	0.9491	0.1450	0.0	0.0002	0.0015	1.0

percentage of models that have similarity smaller than x.

We conclude that **1)** Similarities of piracy fingerprints and homologous fingerprints distribute differently. The former gathers near 1 whereas the latter gathers near zero. **2)** Our encoder has good generalizability in the sense that the similarity gaps exist regardless of models' architectures. For all types of architectures, a large amount (*e.g.*, 100% for CIFAR10) of fingerprints have similarities above 0.8 whereas for homologous models, only a small portion of fingerprints (*e.g.*, 20% for CIFAR10) have similarity above 0.4. However, the similarity gap does vary between architectures. The encoder performs the best on architecture that is trained on (*e.g.*, Arc.A for FMNIST and ResNet34 for CIFAR10). **3)** The variance of homologous fingerprints are larger than that of piracy models (*i.e.*, in CIFAR10, the largest similarity gap between architectures is 0.9 versus 0.0 with CDF granularity equals 0.1). This indicates that piracy model is restricted to a small subspace whereas homologous models lie in a larger subspace.

Table 2 and Table 3 show the average similarities between suspect models and the victim. The largest similarity gaps between piracy and homologous models for three datasets are 0.99, 0.95, 0.58, respectively. The smallest gaps are still over 0.77, 0.69, 0.43. TinyImageNet has the smallest similarity gap because its lower SOTA accuracy results in worse extraction performance. We will study the influence of extraction performance on our framework in ablation study.

Hypothesis Tests. In practice, defenders can adopt Two-Sample t-Test to safely verify the suspect model in less than 20 fingerprints. Formally, let Ω and Ω_{homo} be two sets of fingerprint similarities calculated from suspect models and homologous models. We define the null hypothesis as: $\mathcal{H}_0 : \mu < \mu_{homo}$, where $\mu = \overline{\Omega}$, $\mu_{homo} = \overline{\Omega}_{homo}$. The t-Test will either reject H_0 with a controllable significance level α to claim a piracy model, or give inconclusive results.

The P value column in Table 2 and Table 3 is calculated using t-Test performed on 20 randomly sampled fingerprints. We repeat this t-Test for 30 times and present the average value. By setting the predefined significance level α to 0.05, we successfully reject H_0 for all the piracy models, giving a detect success rate equals 100%

Table 3. Mean and STD of model similarities between the victim and suspect model and p-value (lower is better) using 20 fingerprints (CIFAR10 & TinyImageNet).

	CIFAR10				TinyImageNet			
Archi	Type	Mean	STD	P value	Type	Mean	STD	P value
ResNet	Piracy	0.9945	0.0032	0.0	Piracy	0.7463	0.0955	10ˆ(-14)
	Homo	0.0476	0.0156	1.0	Homo	0.2500	0.1462	0.5278
VGG	Piracy	0.9917	0.0050	0.0	Piracy	0.7568	0.0937	10ˆ(-16)
	Homo	0.1950	0.0920	0.99	Homo	0.2602	0.1530	0.1574
GoogLeNet	Piracy	0.99	0.0066	0.0	Piracy	0.7280	0.1621	10ˆ(-9)
	Homo	0.2984	0.1682	0.69	Homo	0.2404	0.1440	0.8277
DenseNet	Piracy	0.9924	0.0044	0.0	Piracy	0.8215	0.1002	10ˆ(-15)
	Homo	0.0552	0.1955	1.0	Homo	0.2943	0.1656	0.7088

Comparison to Existing Methods. We use the area under the ROC curve (AUC) to measure the performance of our work. Compared with prior work (IPGuard [5]), the performance of that is the AUC of 0.83, 0.75, 0.61 in three datasets (FMNIST, CIFAR10 and T-ImageNet), our framework exceeds theirs and achieves the AUC of 1.0, 1.0, 0.98.

5.3. Ablation Study

Number of Datapoints n. Recall that our fingerprint generation function $\mathcal{F}(f, \mathbf{v}, \mathcal{X}_n)$ depends on a set of datapoints $\mathcal{X}_n$ where n is the number of datapoints. As demonstrated in Figure 6, with the increment of n, the similarity gap between piracy models and homologous models increases, until n reaches a plateau at 70 datapoints. Larger n means that fingerprint captures more decision boundaries and is more informative. In our experiments, we fix n as 100 to balance the trade-off between effectiveness and efficiency.

Top-K Confidence Values. We evaluate the performance of our framework when the suspect model only returns its top-k confidence values. As demonstrated in Fig. 6, the similarity gap remains when $k = 1$, indicating that even hard-label can disclose global information of decision boundaries. The similarity gap is positively consistent with k and becomes stable at 3. We find the average sum of returned top-3 confidence scores equals 0.9996, meaning that there is not much information loss.

Performance of Model Extraction. An attacker may stop early before reaching the optimal recovery rate during extraction. By varying the query and iteration numbers, we obtain 20 models for each interval of length 0.02 of recovery rate in $[0.78, 0.94]$. Figure 6 shows that when the recovery rate is 0.78, our framework can still detect the piracy with a high similarity 0.88. One explanation is when performing model extraction, the piracy models roughly form a decision boundary globally aligned with the victim, which enables our detection. Then it refines its local gradients which improves recovery rate and increases similarity.

Universal *vs.* Local Adversarial Perturbations. To com-

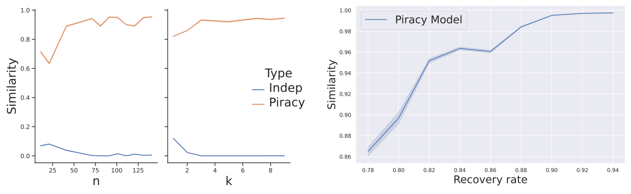
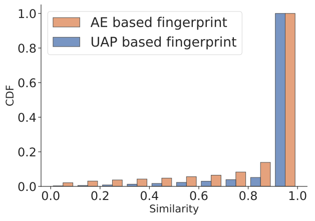
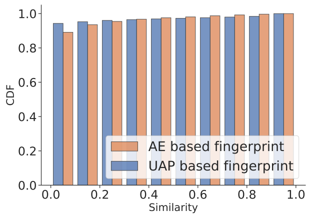


Figure 6. Ablation Study on parameters n, k (left) and Recovery rate (right).



(a) Sim CDF between $f_{\mathcal{V},u}$ and $f_{\mathcal{P},u}$ (FMNIST)



(b) Sim CDF between $f_{\mathcal{V},u}$ and $f_{\mathcal{V},v}$ (FMNIST)

Figure 7. The similarity score given by the contrastive encoder in UAP-based and adversarial example based fingerprints.

pare the information capture capabilities of UAP and AP, We replace UAP by AP and re-conduct our experiments. Specifically, the fingerprint is now $\mathcal{F}_{ap}(f,(\mathbf{x}_1,\cdots,\mathbf{x}_n)) = [f(\mathbf{x}_1), f(\mathbf{x'}_1),\cdots,f(\mathbf{x}_n), f(\mathbf{x'}_n)]$, where $\mathbf{x}'$ is the AP of $\mathbf{x}$ crafted by DeepFool [28] ($\epsilon = 22$). The perturbation norm of $\mathbf{x}'$ approximates UAP, other settings is unchanged. Figure 7 shows the similarity score given by the contrastive encoder. We observe that for AP, the similarities of piracy models are less concentrated in 1 and that of homologous models are less concentrared in 0. This indicates AP based fingerprints have worse performance than UAP based one.

We also study the influence of the usage of contrastive loss and overlapping rate between homologous' and victim's datasets. See supplementary material for details.

5.4. Resistance against Model Modifications

A smart attacker may deliberately modify the piracy copy in order to evade the detection, we evaluate the robustness of our framework against four post-processing techniques on FMNIST dataset .

Fine-tuning. Fine-tuning involves continuing training piracy models using additional data. In our experiment, We fine-tune piracy models on datapoints sampled from test dataset for 10 iterations.

Pruning & Quantization. Pruning [41] and quantization [11] are two usual techniques to compress models and reduce memory while preserve model's functionality. In our experiment we choose pruning rate in $[0.2, 0.6]$ and we convert models from FP32 into INT8. As Figure 8 (left) demonstrated, for the fine-tuning, quantization and pruning, the variation of similarities of piracy models are small. All modified piracy models by those three techniques still have

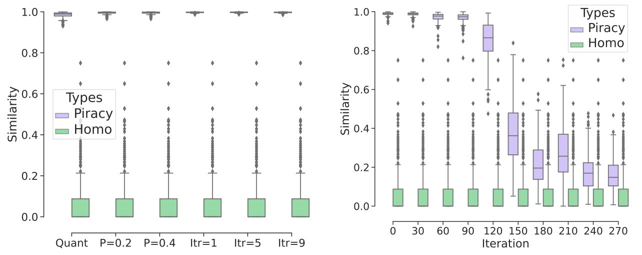


Figure 8. Resistance against model modifications. Similarity distribution of piracy models after quantization, prune and fine tuning (left) and adversarial training (right). On the left, purple boxes assemble on top because their similarities achieve 1.0.

similarities more than 0.88. We hypothesize that they have little effect on the model's decision boundaries.

Adversarial Training. Adversarial training [25] aims to promote the robustness of models intrinsically. We train the piracy models for a maximum 270 adversarial iterations. In each iteration, we craft 128 adversarial examples using DeepFool [28] as new datapoints.

As shown in Figure 8 (right), when the iteration is larger than 120, the similarity continues to drop from 0.99 to 0.89 but still remains high. This is because the adversarial training will continuously shape the model's decision boundaries by pushing the decision boundary towards adversarial examples. The similarity gap is imperceptible after 270 adversarial training iterations with the model utility drops 0.17. So the attacker faces a dilemma to sacrifice the stolen model's utility for evading our detection framework.

6. Discussion

In this paper, we propose a novel framework against model extraction attacks based on the subspace dependency of UAPs of the victim model and piracy model. Incorporating with contrastive learning, we project victim model close to piracy models and far away from homologous models. Evaluation on three benchmark datasets show that our framework is highly effective, general and robust.

Limitations. Our approach can be improved by making a better effectiveness-efficiency trade-off as the information in fingerprints relates to query number. We leave the overhead of training data preparation of encoder as another future work. Our discovery on transferability of encoder suggests a potential solution (see details in supplementary).

Acknowledgements. Z. Peng, S. Li, and H. Zhu were partially supported by the National Key Research and Development Program of China under Grant 2018YFE0126000 and the National Natural Science Foundation of China under Grant 6213000013. M. Xue was partially supported by the Australian Research Council (ARC) Discovery Project (DP210102670) and COVID-19 Recognition Fund of The University of Adelaide.

References

[1] Tiny-ImageNet dataset. https://www.kaggle.com/c/tiny-imagenet. 5

[2] Yossi Adi, Carsten Baum, Moustapha Cissé, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proceedings of 27th USENIX Security Symposium (Security)*, 2018. 1

[3] Tao Bai, Jun Zhao, Jinlin Zhu, Shoudong Han, Jiefeng Chen, Bo Li, and Alex Kot. AI-GAN: Attack-inspired generation of adversarial examples. In *Proceedings of International Conference on Image Processing (ICIP)*, 2021. 2

[4] Santiago Zanella Béguelin, Shruti Tople, Andrew Paverd, and Boris Köpf. Grey-box extraction of natural language models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021. 1

[5] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of Asia Conference on Computer and Communications Security (AsiaCCS)*, 2021. 1, 2, 7

[6] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic extraction of neural network models. In *Proceedings of 40th Annual International Cryptology Conference (CRYPTO)*, 2020. 1, 2

[7] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring connections between active learning and model extraction. In *Proceedings of 29th USENIX Security Symposium (Security)*, 2020. 1, 2

[8] Varun Chandrasekaran, Hengrui Jia, Anvith Thudi, Adelin Travers, Mohammad Yaghini, and Nicolas Papernot. SoK: Machine learning governance. *CoRR*, abs/2109.10870, 2021. 1

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020. 2, 5, 6

[10] Nezihe Merve Gürel, Xiangyu Qi, Luka Rimanic, Ce Zhang, and Bo Li. Knowledge enhanced machine learning pipeline against diverse adversarial attacks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021. 2

[11] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2016. 8

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[13] Zecheng He, Tianwei Zhang, and Ruby B. Lee. Sensitive-sample fingerprinting of deep neural networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[14] Dorjan Hitaj and Luigi V. Mancini. Have you stolen my model? evasion attacks against deep neural network watermarking techniques. *CoRR*, abs/1809.00615, 2018. 1

[15] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2017. 6

[16] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High accuracy and high fidelity extraction of neural networks. In *Proceedings of 29th USENIX Security Symposium (Security)*, 2020. 1, 2

[17] Hengrui Jia, Christopher A. Choquette-Choo, Varun Chandrasekaran, and Nicolas Papernot. Entangled watermarks as a defense against model extraction. In *Proceedings of 30th USENIX Security Symposium (Security)*, 2021. 1

[18] Hengrui Jia, Mohammad Yaghini, Christopher A. Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-Learning: Definitions and practice. In *Proceedings of 42nd IEEE Symposium on Security and Privacy (S&P)*, 2021. 1

[19] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: protecting against DNN model stealing attacks. In *Proceedings of IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019. 2

[20] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Proceedings of Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 5

[21] Valentin Khrulkov and Ivan V. Oseledets. Art of singular vectors and universal adversarial perturbations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[23] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 2

[24] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. Deep neural network fingerprinting by conferrable adversarial examples. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021. 1, 2

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 2, 8

[26] Erwan Le Merrer, Patrick Pérez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020. 1

[27] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 3

[28] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method

to fool deep neural networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[29] Ding Sheng Ong, Chee Seng Chan, KamWoh Ng, Lixin Fan, and Qiang Yang. Protecting intellectual property of generative adversarial networks from ambiguity attacks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[30] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of Asia Conference on Computer and Communications Security (AsiaCCS)*, 2017. 1, 3

[31] David Rolnick and Konrad P. Kording. Reverse-engineering deep relu networks. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020. 1

[32] Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of backdoor-based watermarking in deep neural networks. In *Proceedings of ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec)*, 2021. 1

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015. 6

[34] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[35] Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. DAWN: dynamic adversarial watermarking of neural networks. In *Proceedings of ACM Multimedia Conference (MM)*, 2021. 1

[36] Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. The space of transferable adversarial examples. *CoRR*, abs/1704.03453, 2017. 2

[37] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *Proceedings of 25th USENIX Security Symposium (Security)*, 2016. 1, 2

[38] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017. 5

[39] Honggang Yu, Kaichen Yang, Teng Zhang, Yun-Yun Tsai, Tsung-Yi Ho, and Yier Jin. CloudLeak: Large-scale deep learning models stealing through adversarial examples. In *Proceedings of 27th Annual Network and Distributed System Security Symposium (NDSS)*, 2020. 6

[40] Jiawei Zhang, Linyi Li, Huichen Li, Xiaolu Zhang, Shuang Yang, and Bo Li. Progressive-Scale boundary blackbox attack via projective gradient estimation. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021. 2

[41] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 8

[42] Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. Hermes Attack: Steal DNN models with lossless inference accuracy. In *Proceedings of 30th USENIX Security Symposium (Security)*, 2021. 1