# Mitigating Query-based Neural Network Fingerprinting via Data Augmentation

MEIQI WANG, Tsinghua University, China
HAN QIU, Tsinghua University, China
TIANWEI ZHANG, Nanyang Technological University, Singapore
MEIKANG QIU, Dakota State University, USA
BHAVANI THURAISINGHAM, The University of Texas at Dallas, USA

Protecting the intellectual property (IP) of deep neural network (DNN) models becomes essential and urgent with the rapidly increasing cost of DNN training. Fingerprinting is one promising IP protection method that queries suspicious models with specific fingerprint samples to infer and verify IP by comparing the predictions with pre-defined labels. Based on utilizing unique features of target models, various DNN fingerprinting methods are proposed to effectively verify the IP of models remotely with a meager false-positive ratio. In this paper, we propose a novel attack to mitigate query-based fingerprinting methods based on data augmentation methods. We propose a randomized transformation on input samples to significantly mislead the fingerprint samples' prediction and compromise the IP verification. Then, our attack can keep the model utility with an acceptable accuracy drop in the data-free scenario (i.e. without any samples) or achieve much higher precision in the data-limited scenario (i.e. with a small number of samples with the same distribution). An intensive evaluation of three well-known model structures and three well-known datasets shows that our attack can effectively mitigate five query-based DNN fingerprinting methods in top-tier conferences.

CCS Concepts: • **General and reference** → **Empirical studies**; • **Security and privacy**; • **Computing methodologies** → **Computer vision**; **Neural networks**;

Additional Key Words and Phrases: data augmentation, deep neural network, fingerprinting, intellectual property

## 1 INTRODUCTION

Nowadays, deep neural networks (DNNs) tend to become more and more complex with the rapidly increasing number of layers and parameters [23] to achieve better performance [47]. Training these complex high-performance DNN models always requires massive training resources [26]. For instance, StyleGAN3 [18] trained by Nvidia consumes about 92 GPU years and more than 225 MWh of electricity on GPU clusters. Besides exponentially increased training time and power consumption, training complex and high-performance DNN models also require a large amount of training/test data [8, 48], experts' knowledge [36], etc. These training efforts make a well-trained DNN model an important intellectual property (IP) [7].

Authors' addresses: Meiqi Wang, Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China, 100084, wangmq22@mails.tsinghua.edu.cn; Han Qiu, Institute for Network Sciences and Cyberspace, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China, 100084, qiuhan@tsinghua.edu.cn; Tianwei Zhang, the School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798, tianwei.zhang@ntu.edu.sg; Meikang Qiu (corresponding author), the Dakota State University, Madison, South Dakota, USA, qiumeikang@ieee.org; Bhavani Thuraisingham, the Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080, USA, bhavani.thuraisingham@utdallas.edu.


1550-4859/2023/5-ART $15.00
https://doi.org/10.1145/3597933

However, recent research work points out that the ownership of these well-trained DNN models is vulnerable [10, 25]. First, DNN models are vulnerable to model stealing attacks [49]. For instance, a learning-based stealing attack is proposed to reconstruct the DNN model weights, which can extract a complex model with an error rate within 10% via queries for inference [16]. Besides query-based model stealing attacks at the application layer, [49] steals a DNN model without any performance loss by analyzing the unencrypted PCIe traffic at the hardware level. Second, although model stealing attacks usually require specific conditions and sophisticated attackers to perform, malicious users that have access to the deployed DNN models may illegally copy and post-process these models for profit [44]. Consider that these threats are more common and severe in edge AI computing (e.g.the smart city), and IP verification is also problematic. These emerging threats make IP protection for DNN models an essential and urgent task.

Existing IP protection methods for deep learning models can be roughly divided into two categories. (1) *Watermarking*: carefully-crafted watermarks are embedded by the model owner in his model by a parameter regularizer [39] or backdoor data poisoning [22, 25, 45]. Later, the watermarks can be extracted from the model parameters or output as the IP evidence. (2) *Fingerprinting*: the model owner discovers unique sample-label pairs that can exclusively and exactly characterize the target model such as various proposed approaches [5, 27, 30, 42]. Compared with watermarking, fingerprinting does not need to modify the target models, which is more promising, especially for some scenarios where the model owner does not have the right or capability to modify the models. For instance, the models deployed on autonomous driving systems (ADS) [12], clouds [32], or industrial control systems [11] are usually remotely and out of the model owners' control. It is not easy to inject watermarks into those models such that methods like fingerprinting can effectively provide IP verification. Thus, we focus on fingerprinting in this paper.

We note that the model supposed to be protected is the *target model* and the model to be verified is the *suspicious model*. The core step of DNN fingerprinting relies on the remote query-based inference on a suspicious model with generated fingerprint samples and verifying the IP by comparing their predictions with the pre-defined labels. If the ratio of the same predicted label is larger than a threshold, this suspicious model will be determined as a *stolen model*; otherwise, it supposes to be an independently trained model for the same task without any illegal action named as an *independent model*. Existing fingerprinting works [5, 27, 42] try to craft perturbations on clean samples or random initialized samples to generate fingerprint samples. Generating such perturbations normally uses optimization-based or gradient-based approaches, which are similar to adversarial attack methods [6, 46]. However, compared with adversarial attacks, fingerprint samples release the L-norm constrains [5] and limit the generated samples' transferability [27, 42]. Thus, these fingerprint samples can effectively verify the IP of one target model even if the adversary modifies the illegally-acquired target model with model compression [17], model pruning [28], or fine-tuning [43].

With promising results of published fingerprinting methods, we would like to address the following challenge from an adversary's perspective: *can we mitigate the fingerprinting-based DNN IP protection while preserving the stolen model's utility in a lightweight manner?* In this paper, we *first* solve this challenge by proposing an *effective utility-preserving*, and *lightweight* attack method to mitigate multiple existing DNN fingerprinting methods. Specifically, we propose a data augmentation-based approach, i.e. *Certain Scale with Random Padding* (CSRP), which aims to obfuscate the input samples' gradients by randomly distorting and resizing to drop and pad pixels of one image. Then, depending on whether the attacker has training data, we considered two scenarios: the data-free and the data-limited scenarios. For *effectiveness*, we can mitigate multiple existing state-of-the-art DNN fingerprinting methods by significantly modifying the output labels of their fingerprint samples to compromise the IP verification. For *utility-preserving*, we have little influence on benign samples such that the adversary can still profit from an illegally acquired model. For *lightweight*, in the data-free scenario, the preprocessing-only process of input samples has a time complexity of no more than $O(N)$. In the data-limited scenario, we only need

a few epochs of fine-tuning with 5% of training data, which requires a negligible cost. Our contributions are as follows:

- To the best of our knowledge, we propose the *first* effective attack to disable various state-of-the-art DNN fingerprinting methods with two different scenarios.
- Our attack can preserve the utility of the illegally acquired models by introducing a tiny influence on benign sample classification.
- We evaluate our attack through extensive experiments on three widely used architectures (ResNet-56, VGG-16, and MobileNet) and three famous datasets (CIFAR-10, CIFAR-100, and TinyImageNet) with a total of over 1,000 models and around 2,000 GPU hours.

The roadmap of this paper is as follows. The research background is in Section 2. We list the system overview and threat model in Section 3. The details of our method and analysis are given in Section 4. The experimental details, results analysis, and ablation study are in Section 5. We discuss and list potential future works in Section 6 and conclude in Section 7.

## 2 BACKGROUND

### 2.1 Edge AI Computing and IP Threats

Edge computing promotes the data and computing required by the Internet from the cloud to the edge, making real-time data processing possible. With the rapid development of artificial intelligence (AI), AI can be embedded into every element of the Internet of Things (IoT) accessible by end users, which promotes the birth of the Artificial Intelligence of Things (AIoT) [9]. Thus, AI models can be distributedly trained and inferred on edge [29], in which each node can process data autonomously and share and transmit information to each other. For instance, many smart cameras are deployed in the smart city to real-time detect persons and analyze actions [38]. We assume that each camera has its sensors to acquire data and has close access to the edge node's processors and storage, which can infer (compute) a DNN model to process data and store data, model parameters, and the results. However, we notice that there is a great chance that the DNN models deployed in these edge nodes are those illegally pirated models which threaten the IPs for lower cost. Furthermore, the verification of their IPs is easily invalidated for two main reasons: (1) Due to the edge devices' limited computing and storage resources, the DNN models deployed on edge are usually modified by model compression, so whether the modified models can still retain their copyright is a very challenging issue. (2) Since most models calculated, verified, and maintained only on edge are black-box for the cloud center, this will limit their IP verifications only to remote access. Thus, lightweight tampering on the edge is not easily noticed. These emerging threats make IP protection for the DNN models on edge an important and urgent task.

### 2.2 DNN IP Protection

**Watermarking.** The early approach to protect the IP of DNN models is to watermark a DNN model [22, 25] relying on using out-of-distribution features [1, 45] or handcrafted parameters [3]. For instance, the model owner modifies a small subset of original training data by embedding pre-defined noise (e.g., triggers in backdoor attacks [24]). These modified samples will be specifically assigned to an incorrect pre-defined label (different from the ground-truth label). Then, these out-of-distribution features will be embedded into the target model by training the model with both original training data and those modified data as DNN watermarking. This watermarked model will behave normally on benign samples but predict the same labels as the pre-defined labels for watermarked samples to verify the model's IP. However, the clear limitations of the watermarking approach can be summarized as follows. (1) The watermark embedding process will decrease the model accuracy, which is unacceptable in critical applications [5]. (2) Watermarking procedures are usually combined with the training phase, which is hard to be deployed for those off-the-shelf models [13].

**Fingerprinting.** The recent more promising approach to protect the DNN models' IP is the DNN fingerprinting approach [5]. The general idea of the DNN fingerprinting method is to verify the target model ownership via its unique and robust features (e.g., decision boundaries [5]) without any modification to this model. There are two steps of DNN fingerprinting, including generating unique fingerprint samples and inference for verification. For the first step, many existing works propose different generating solutions for classification models, such as [5, 27, 41, 42]. They craft samples differently by adding gradient-based perturbations exclusively for the target model to achieve setting pre-defined labels for these samples (see more details in Section 2.2). For the second step, the model owner uses these fingerprint samples to query a suspicious model for inference. Existing works are all based on the same principle: A stolen model is supposed to give the desired unique labels as the ownership evidence, and an independent model is supposed to predict normal ground-truth labels still. Unlike watermarking, the model owner can construct the fingerprint samples without any modifications to the training phase, preserving the models' accuracy. Additionally, fingerprinting conducts ownership verification through inference query only, which does not require access to any model information, such as model parameters, thus bringing much more convenience and applicability to IP protection.

Formally, consider a target model $f$ and its generated $N$ fingerprint sample sets $\mathcal{X}_f = \{(\mathbf{x}_i, y_i)\}_N, i \in \{1, N\}$ for IP verification, where $y_i$ is the label of $i$-th sample $\mathbf{x}_i$. These fingerprint samples should all meet the verification criteria as $y_i = f(x_i), \forall i \in \{1, N\}$. Given a suspicious DNN model $\widehat{f}$, the query-based IP verification relies on a matching rate $T(\widehat{f})$, which denotes the inference results of these $N$ fingerprint samples. By comparing the $T(\widehat{f})$ with a pre-defined threshold $T_F$, whether this suspicious model $\widehat{f}$ can be classified as a stolen model or an independent model can be determined. Using a lower threshold $T_F$ is more likely to find more stolen models but may cause false positives on independent models. Thus, more sophisticated methods such as using the Area Under Curve (AUC) [5] are deployed for better selecting the threshold $T_F$ for IP verification. In this paper, since we focus on mitigating the fingerprinting method to significantly decrease $T(\widehat{f})$, the matching rate is used as our metric for evaluation (see more details in Section 5.1).

## 2.3 Target DNN Fingerprinting Methods

Generating fingerprint sample sets $\mathcal{X}_f = \{(\mathbf{x}_i, y_i)\}_N, i \in \{1, N\}$ in existing works relies on the gradient-based or optimization-based approaches to craft perturbations on clean or random initialized samples. In this paper, we reproduce four kinds of state-of-the-art DNN fingerprinting methods collected from the top-tier conferences in recent years, including IPGuard [5], conferrable adversarial example (CAE) [27], vanilla characteristic example (VCE), and robust characteristic example (RCE) [42]. These four fingerprinting methods are briefly illustrated as follows.

**IPGuard.** Based on observing a unique classification boundary in a DNN model, IPGuard aims to extract some specific data points near the classification boundary of the target model as its unique and robust features for fingerprinting. IPGuard formulates an optimization problem to find such points to restrain the gap between the data points and the boundary. Then, they use a gradient descent method to iteratively modify the clean samples along the objective function gradient in this optimization problem until a set of fingerprint samples are generated.

**CAE.** By constraining the transferability, CAEs can be used as fingerprint samples that can only transfer to the models acquired by stealing and modifying the target model but not to independent models. A conferrable ensemble method by including several independent models is proposed to limit the transferability of the fingerprint samples which are generated with the gradient descent method. A conferrability score indicates the restriction on the transferability of fingerprint samples which can guide the effective fingerprint sample generation and guarantee low false-positive for IP verification.

**VCE and RCE.** Different from the above two fingerprinting methods that try to craft perturbations on clean samples in the training dataset, [42] proposes to craft perturbations on random samples to eliminate the influence

of training data distribution. The projected gradient descent (PGD) algorithm [20] is used to modify the random initial samples to generate VCEs as fingerprint samples for randomly pre-assigned labels. Then, by including not only target models but also modified target models (e.g. pruned models), RCE is proposed to further enhance the robustness to differentiate the modified target models. RCE claims better robustness by adding noise to the target model parameter. We notice that a gradient mean (GM) method [2] is deployed in [42] to further enhance the robustness of RCE (RCE-GM), which is also included in our evaluation. Besides enhancing robustness, low-transferability (LT) is also considered to limit false-positive for independent models. RCE-LT is also proposed to reduce transferability further and limit false positive issues (i.e. misclaiming the ownership of independent models). The core idea of RCE-LT is to apply a low-frequency mask on the Discrete Cosine Transform (DCT) of RCE-based samples. This mask is a high pass filter in the frequency domain to remove critical information in RCEs-LT.

## 2.4 Existing Methods to Mitigate DNN Fingerprinting

In previous works, attackers are assumed to modify one illegally acquired model to avoid fingerprinting-based IP verification. For instance, they use post-processing methods, such as model pruning and model fine-tuning, to slightly change the model's decision boundaries. These methods can maintain relatively high accuracy on benign samples far away from the boundaries but may affect the fingerprint samples that are assumed to be closer to decision boundaries. Therefore, these methods are widely used as attacking methods against fingerprinting methods in [5, 42] for evaluation. On the other hand, attackers may also compress the illegally acquired model for deployment. For instance, the models used in edge terminal devices (e.g. portable devices and sensor networks) are always limited to computing and storage resources. Therefore, the post-processing methods such as model compression are commonly used as required if the models are illegally acquired from a large and complex target model. Thus, model compression, model pruning, and model fine-tuning are three feasible methods to modify one illegally acquired model that maintain an acceptable model performance and also may potentially mitigate the verification based on fingerprinting. In this paper, we follow previous works [5, 42] to use them as the baselines of our attacking method in evaluation.

**Model compression.** High-performing DNN models are usually massive and complex in computing, which makes the inference process slow and requires large storage space and computational power. Therefore, for applications in which real-time predictions are needed, or model capacity is limited, complex models need to be compressed into simpler, smaller, and faster ones with comparable performance when deployed in use [4]. Model compression has many approaches; here we refer to the quantization method. Note that model pruning is also one of the compression approaches and will be discussed separately. The experimentation of compression in this paper is a quantization method that replaces high precision (e.g. 32-bit floating-points) with low precision (e.g. 8-bit integers) to reduce the model size and change the model itself.

**Model pruning.** Model pruning is another popular model compression method. When models are deployed for inference and prediction, redundant parameters can be pruned to reduce computing power and time. Before pruning, the quantity of parameters to prune is required to be set as a fractional format between 0 and 1, which represents the fraction of parameters to prune. For pruning experimentation in [5], the weights with as small absolute values as close to zero will be pruned. We increase the pruning rates with a step size of 0.1 from 0.1 to 0.5, decreasing accuracy significantly.

**Model fine-tuning.** Initially, model fine-tuning is used to modify a pre-trained model's parameters wholly or partially while maintaining or improving the model's performance [1]. Unlike model compression or pruning, fine-tuning a pre-trained model requires access to at least a partial training dataset. This is another scenario where attackers have an illegally acquired model and a partial training dataset. Model fine-tuning has many variations, such as fine-tuning last layers (FTLL), fine-tuning all layers (FTAL), re-training last layers (RTLL), and

re-training all layers (RTAL) [1]. Here, we choose FTAL to mitigate DNN fingerprinting methods since FTAL can maximize the model modification among all the above fine-tuning methods [5]. Note that for other fine-tuning variations, our method has a similar performance. We follow previous works to fine-tune one model for a few epochs with a small training dataset.

## 3 PRELIMINARIES

### 3.1 System Overview

We assume an adversary inserts a preprocessing layer at the front of the stolen DNN model $\widehat{f}$, which will be applied to all input samples when others (e.g.model owners or general users) query this model for IP verification or normal inference service. In this paper, the preprocessing is supposed as a randomized transformation data augmentation $g(\cdot)$. This attack scenario overview can be summarized as in Figure 1 (a) and (b). When a set of fingerprint samples $\{(\mathbf{x}_i, y_i)\}_N, i \in \{1, N\}$ is sent to query for IP verification, the randomized transformation will mitigate the features carried by these samples and mislead the predicted labels, i.e.$\widehat{f}(g(\mathbf{x}_i) \neq y_i)$. An adversary can mitigate the fingerprinting method by significantly decreasing the verification ratio and manipulating the IP verification results. For the clean samples $\mathbf{x}$ as inputs, the accuracy score (ACC) is slightly influenced, i.e.$\widehat{f}(g(\mathbf{x})) \approx f(\mathbf{x})$, such that the adversary can still illegally profit via providing inference services. Therefore, we aim to find a better $g(\cdot)$ to achieve the above purposes.

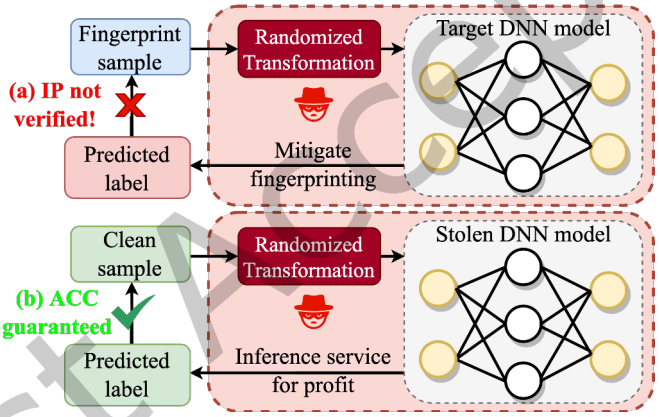


Fig. 1. Scenario overview: mitigating fingerprint samples' inference while keeping the stolen model utility via randomized transformation.

Here, we assume that the data augmentation in each edge node's inference service is computed locally and independently. For instance, in the distributed system, each node has its processors (most likely with at least one acceleration hardware) to make data augmentation and DNN inference, and its storage to store the program, the input data, augmented data, model parameters, and inference results. Thus, only the inference results and the input data will likely share across the system. Besides, the DNN models, the augmented data, and the augmentation function $g(\cdot)$ are maintained and computed locally in the black-box scenario. Note that different nodes may have different $g(\cdot)$ according to different tasks and models.

### 3.2 Threat Model

**Adversary's goal.** The adversary tries to use the data augmentation function $g(\cdot)$ as the basis to mitigate various fingerprinting methods. Thus, $g(\cdot)$ must effectively decrease the fingerprint samples' matching rate to a similar

level compared with independent models such that the IP verification will be confused and not succeed. Then, using $g(\cdot)$ only or combined with fine-tuning for arbitrary input samples must preserve the model utility, like prediction accuracy. In summary, the attacker aims to modify and deploy one illegally acquired DNN model to achieve three goals, i.e. *effectiveness*, *utility-preserving*, and *lightweight* goals.

- *Effectiveness*: For a set of fingerprint samples $\mathcal{X}_f = \{(\mathbf{x}_i, y_i)\}_N, i \in \{1, N\}$, the matching rate $T(\widehat{f})$ of $\widehat{f}$ must be similar with the matching rate $T$ of many independent models.
- *Utility-preserving*: Our proposed transformation function $g(\cdot)$ cannot affect the prediction accuracy of clean samples via preprocessing: $f(g(\mathbf{x})) \approx f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}_{clean}$.
- *Lightweight*: The complexity of the attack needs to be low. The cost of such an attack must be significantly less than training a model from scratch and require no data or a small amount of data to perform.

**Adversary's capability and knowledge.** We assume the adversary will deploy the illegally acquired DNN model $f$ for inference services. We assume the entire service can only be a remote query with input samples and return only predicted labels (i.e.without any other information such as the confidence score). We also assume the illegally acquired DNN model is correct without DNN backdoors [33] or any embedded DNN watermarks [14]. This model's only IP verification method is various DNN fingerprinting methods.

### 3.3 Data-free and Data-limited Attack Scenarios

We classify two scenarios following the settings in previous work [40] including *data-free* and *data-limited* scenarios.

**Data-free scenario.** We assume that the adversary neither has the training dataset $\mathcal{X}_{train}$ to train an independent DNN model from scratch nor can he acquire partial clean dataset $\mathcal{X}_{clean}$ to fine-tune the target DNN model $f$. He can only manipulate the input samples without any prior knowledge about whether it is a benign sample or a fingerprint sample. We consider that an attacker illegally acquires a DNN model $f$ and aims to deploy it with a modified inference process $\widehat{f}(\cdot)$ for illegal profit. The attacker will try to mitigate potential DNN fingerprint samples by modifying the input samples of their deployed model $\widehat{f}$. The IP verification will be compromised as long as the matching rate $T(\widehat{f})$ of fingerprint samples is significantly decreased (e.g. $y_i \neq \widehat{f}(\mathbf{x}_i)$ for most $(\mathbf{x}_i, y_i) \in \mathcal{X}_f$). Particularly, the attacker adds one preprocessing function $g(\cdot)$ which randomly transforms an input image $\mathbf{x} \in \mathcal{X}$ to an output $g(\mathbf{x})$ with the same dimension. For all input samples, the inference process is $\widehat{f}(\cdot) = f(g(\cdot))$. Since one of the adversary's initial motivations is illegally profiting via inference, adding $g(\cdot)$ must preserve the model accuracy within an acceptable range.

**Data-limited scenario.** For the data-limited scenario, we assume that the adversary has a small amount of clean dataset (e.g. 5% of the $\mathcal{X}_{train}$ in this paper) such that he can fine-tune the illegally acquired model for a few epochs. Thus, he can manipulate the input samples with the data augmentation function and modify the illegally acquired model to further achieve the adversary's goals. Obviously, not only fingerprint samples but also benign samples will be affected after preprocessing. If the input sample is intensively augmented for mitigating the fingerprint samples, a significant decrease in model accuracy is inevitable, i.e. $f(g(\mathbf{x})) \neq f(\mathbf{x})$, for most $\mathbf{x} \in \mathcal{X}_{clean}$. For preserving the model utility, the adversary can fine-tune the model $f$ with (limited amount of) augmented clean data by $g(\cdot)$ and deploy it as a fine-tuned model $f^g$. The fine-tuned model $f^g$ can learn the pattern of the data augmentation such that it can perform better on those benign inputs with the same augmentation function $g(\cdot)$ to preserve the model utility. For data-limited scenarios, the modified inference process is $\widehat{f}(\mathbf{x}) = f^g(g(\mathbf{x}))$.

# 4 METHODOLOGY

## 4.1 Data Augmentation for Data-free Scenario

This paper proposes a randomized input transformation, CSRP, as our data augmentation function $g(\cdot)$ for input samples. The core idea is to significantly change the gradient of one sample by distorting the pixel positions and dropping some pixels. Notably, we resize an input image into a large one with a scare ratio $\sigma$, randomly pad values with a padding parameter $p$, and then resize it back to its original size. Therefore, the transformed image will significantly differ considering the pixel coordinates and values since a random ratio of pixels will be dropped. This random transform operation will cause an obfuscation of gradients of this sample's inference process, which can mitigate the features carried by those gradient-based fingerprint samples. Since this transform does not fundamentally change the image contents, the accuracy score for clean samples will only be slightly decreased.

**ALGORITHM 1:** CSRP

**Input:** original image $I \in \mathbb{R}^{H \times W}$
**Output:** distorted image $I^{CSRP} \in \mathbb{R}^{H \times W}$
**Parameters:** scale limit $\sigma$; padding parameter $p$

/* Step 1: Resizing to a certain scale */
1 $\{H_{max}, W_{max}\} = \lfloor \{H, W\} \times \sigma \rfloor$;
2 $\{H_{resize}, W_{resize}\} = \{H_{max} - p, W_{max} - p\}$;
3 $I'(x, y) = I(\mathsf{round}(x \cdot H/H_{resize}), \mathsf{round}(y \cdot W/W_{resize}))$ s.t. $I' \in \mathbb{R}^{H_{resize} \times W_{resize}}$;

/* Step 2: Padding randomly to a large image */
4 $x_1 \sim \lfloor \mathcal{U}(0, p+1) \rfloor$, $y_1 \sim \lfloor \mathcal{U}(0, p+1) \rfloor$;
5 $x_2 = p - x_1$, $y_2 = p - y_1$;
6 $I_m = \mathsf{padding}(I', ((x_1, x_2), (y_1, y_2)), value = 0)$ s.t. $I_m \in \mathbb{R}^{H_{max} \times W_{max}}$;

/* Step 3: Resizing back to the size of $I$ */
7 $I^{CSRP}(x, y) = I_m(\mathsf{round}(x \cdot H_{max}/H), \mathsf{round}(y \cdot W_{max}/W))$ s.t. $I^{CSRP} \in \mathbb{R}^{H \times W}$;
8 **return** $I^{CSRP}$;

The core of our solution is a powerful random transformation function, Certain Scale with Random Pad (CSRP). The details of CSRP are explained in Algorithm 1 with three steps. (1) For one image $I \in \mathbb{R}^{\{H,W\}}$ with the size of $\{H, W\}$, we obtain a certain larger image size $\{H_{max}, W_{max}\}$ with a certain scale limit $\sigma$ calculated by $\{H_{max}, W_{max}\} = \lfloor \{H, W\} \times \sigma \rfloor$. Then, with a padding parameter $p$, we can obtain a certain size $\{H_{resize}, W_{resize}\} = \{H_{max} - p, W_{max} - p\}$. For getting a resized larger image $I'$, we pick pixel values from the original image $I$ by inserting pixels and calculating the certain pixel coordinate ratio (Line 1-3). (2) The second step is to pad the resized image $I'$ with padding parameter $p$ to generate an even larger image $I_m$. The padding coordinates $(x_2, y_2)$ are calculated from a random seed $(x_1, y_1)$ selected from a uniform distribution $\lfloor \mathcal{U}(0, p+1) \rfloor$. Then, we get a padded image $I_m$ by padding zeros in these coordinates (Lines 4-6). (3) In the end, we need to resize this padded image $I_m$ back to its original size to get the output image $I^{CSRP}$. (Line 7). Finally, as the randomized transformation result, the image $I^{CSRP}$ will be sent to the target model $f$ for inference.

Further, we give the theoretical analysis of Algorithm 1's time complexity to prove the lightweight of our method. Assume that the input image has a size of $N \times N$. As mentioned above, both the first and third steps in Algorithm 1 can be formulated as resizing processes, and the second step is the padding process. Because all those processes are pixel-wise manipulation and the time complexity of modifying one pixel in the image is

$O(1)$, thus we calculate the time complexity in a one-pixel unit. We compute in three steps: (1) The first resizing normally modifies $(\sigma \times N - p)^2$ pixels, for which complexity is $O(2log(N))$ using recursive methods. However, in a better implementation of CSRP, we find that the number of modified pixels is almost the same regardless of image size, which will achieve more padding for larger $N$ to make the features of the fingerprint sample less decisive. Therefore, the resizing value is a constant number $C$ with complexity $C \times O(1)$ equal to $O(1)$. (2) As mentioned in (1), the edge padding process uses recursive algorithms to fill $p$-pixel value proportional to $N$. Thus the time complexity of padding is at least $O(log(N))$ and no more than $O(N)$. (3) The second resizing modifies $(\sigma \times N)^2$ pixels, for which complexity is also $O(2log(N))$. Therefore, in summary, combining the three steps, the time complexity of CSRP is at least $O(log(N))$ and no more than $O(N)$, which can achieve relatively fast computing speed and low overhead.

In summary, the CSRP contains two resizing processes, which can be seen as randomly dropping pixels and reconstructing the image contents, thus destroying the gradient information of the fingerprint samples. The visual results of the CSRP are given in Figure 2. We notice that larger scale and padding parameters will lead to significant modification on the samples, which has a better effect on mitigating fingerprint samples but will decrease the model utility. In our threat model, we have no knowledge about which kind of fingerprinting methods will be used. Therefore, we need to locate one proper set of parameters to mitigate various fingerprint samples, which is very challenging.

## 4.2 Further Fine-tuning for Data-limited Scenario

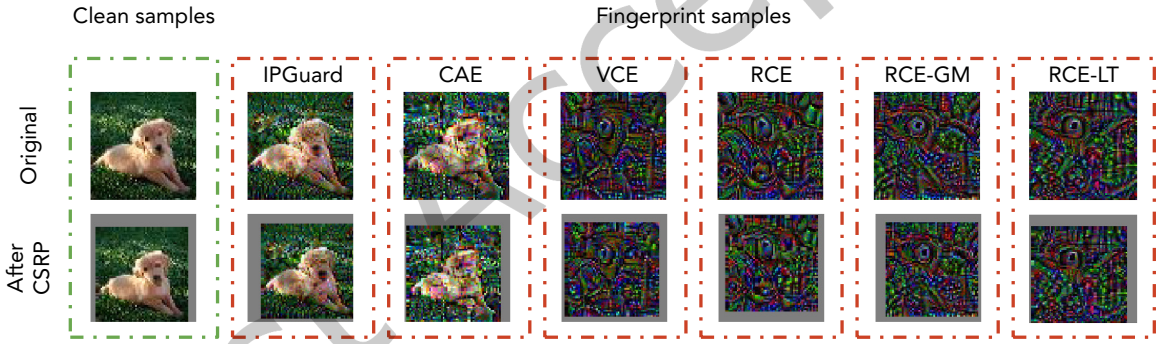


Fig. 2. Visual presentation of clean and fingerprint samples with an example from the TinyImageNet dataset.

We can use a more intensive data augmentation method for the data-limited scenario and fine-tune the model with augmented data. Such an operation can help the model to learn the pattern of CSRP and effectively enhance the model utility. We briefly illustrate the reason as shown in Figure 3. Initially, the benign and fingerprint samples are supposed to be located in the source and target classes in the decision domain. However, CSRP is a tunable random augmentation method that can disturb the features of both the benign and fingerprint samples. Along with more intensive augmentation of CSRP, the fingerprint samples will be misled to other classes to compromise the IP verification. However, the benign samples are also affected and thus are misclassified.

We adopt fine-tuning with augmented clean data in the data-limited scenario to avoid a significant decrease in model utility. We assume that 5% of training data is augmented with CSRP to fine-tune all the layers of the illegally acquired model. Then, the generalization capability of this model is improved: the model can recognize such augmentation and better predict the inference of benign samples preprocessed by CSRP to improve the model

utility preserving. As given in Figure 3, the boundary's learning and changing process is roughly indicated as the red dotted line moving to the green solid line, which is altered against the augmented benign samples. Note that compared with other methods to change the decision boundary, such as model compression, we use fine-tuning here only to improve the model utility purpose. In other words, the mitigation for fingerprint samples is achieved by CSRP with intensive augmentation, while the model utility preserving is achieved by our fine-tuning with partial augmented training data.

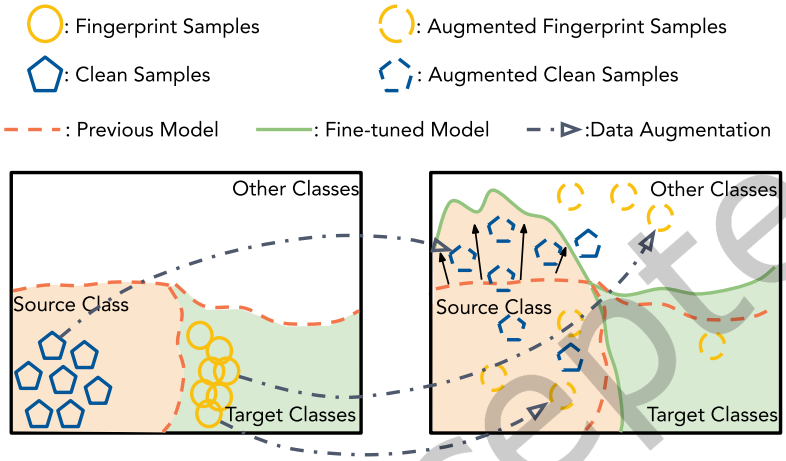


Fig. 3. Visual sketch map of the fine-tuning process and data augmentation of clean and fingerprint samples. We divide the decision areas into three parts for clarity: clean samples' source class, fingerprint samples' target class, and other classes besides these two. *Left*: Samples before data augmentation and decision boundary before fine-tuning. *Right*: Samples after data augmentation and decision boundary after fine-tuning.

### 4.3 Interpretation and Analysis

We interpret and analyze our method with the Local Interpretable Model-Agnostic Explanations (LIME) [35]. LIME interprets a model $f$ by approximating it locally with an interpretable model $f_l$, such as linear models. The key insight of LIME is to explore the explanation model $f_l$ locally around the instance $x$ that needs to be predicted and explained. The interpretation of LIME can be formulated as an optimization problem defined in Eq. (1).

$$\xi(x) = \arg\min_{f_l \in F} \mathcal{L}(f, f_l, \pi_x) + \Omega(f_l), \tag{1}$$

where $F$ is a set of interpretable models, $\pi_x$ identifies the locality and $\Omega(f_l)$ defines the complexity of $f_l$ which means how difficult to interpret. The dominant loss function $\mathcal{L}$ measures how unfaithful $f_l$ is to approximate $f$ in the locality defined by $\pi_x$. Moreover, to minimize the loss independent from $f$ (i.e. to be model-agnostic), the exploration should be conducted by sample sampling and perturbing. Thus, when given $F$ as a set of linear models, the loss $\mathcal{L}(f, f_l, \pi_x)$ will be defined more concretely in Eq. (2).

$$\mathcal{L}(f, f_l, \pi_x) = \sum_{z, z' \in Z} \pi_x(z)(f(z) - f_l(z'))^2, \tag{2}$$

where $z'$ is randomly sampled and perturbed based on $x'$ (where $x'$ is a binary vector for $x$'s interpretable representation). $z$ denotes original representation of $z'$ and $\pi_x(z)$ measures the distance (or weight) between $z$ and

$x$, i.e. locality around $x$. $f_l(z^{'})$ represents the prediction of the perturbed samples $z^{'}$ and $f(z)$ is its ground-truth labels.

Here we present LIME for the DNN image classification task and choose a sparse linear explanation following the previous work [35]. The critical pixels will be highlighted by letting the interpretable representation be a binary vector and indicating whether the contiguous blocks of pixels of similar pixels (called super-pixels) are present. LIME gives intuition to observe which regions play an important role in the model predictions and to explain why the model predicts that class. In our evaluation, we use LIME (interpreted as the green areas) to achieve the goal of two interpretations. One is to explain why CSRP can mitigate the fingerprinting methods effectively, and the other is to show why fine-tuning method can improve the model accuracy in the data-limited scenario.

**An example of the data-free scenario** can be presented in Figure 4. The clean sample (a) is predicted with the label 'German shepherd' and we use IPGuard to craft the fingerprint sample with the label 'espresso' in (b). The corresponding LIME explanation is given in (e) and (f), respectively to show the regions that affect the model prediction are significantly different. The fingerprint sample makes a targeted change in the image features by adding the global perturbation, so its explanation area expands to almost the whole image. Then, we use CSRP to mitigate the IPGuard fingerprint sample to get (c) with a correct prediction label 'German shepherd' again, and its LIME explanation result is given in (g). Note that we have done another CSRP for the same fingerprint sample to eliminate the effect caused by CSRP's randomness, which is given in (d), and its LIME is presented in (h). The explanation results, (g) and (h), present why fingerprint samples are correctly predicted again after CSRP: due to the facial features like the original clean sample (e) rather than other irrelevant features. This example indicates that the CSRP can effectively mitigate the features crafted by the IPGuard fingerprinting method and make the original features influence the inference again.

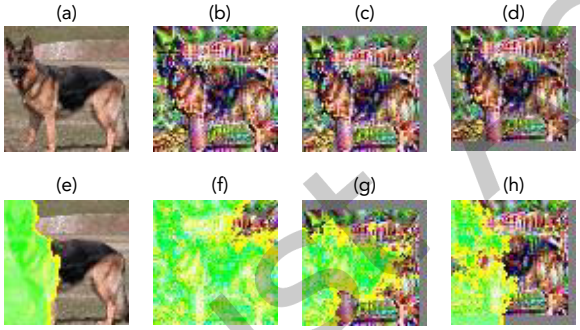


Fig. 4. An example to interpret our solution. Clean sample (a) and its LIME explanation (e), an IPGuard crafted fingerprint sample (b) and its LIME explanation (f), the IPGuard crafted fingerprint sample after CSRP augmenting (c), (d) and their LIME explanation (g), (h).

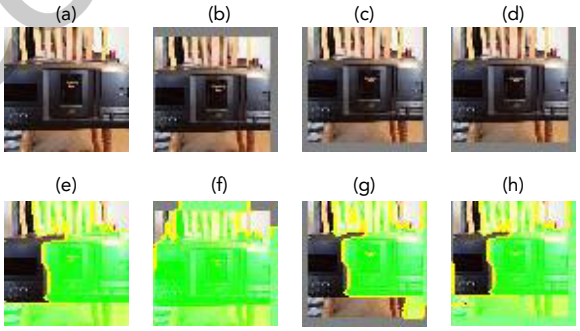


Fig. 5. The second example to interpret our solution. Clean sample (a) and its LIME explanation (e), the clean sample after CSRP augmenting (b) and its LIME explanation (f), the clean sample after CSRP augmenting and fine-tuning (c), (d) and their LIME explanation (g), (h).

**An example of the data-limited scenario** can be presented in Figure 5. The clean sample (a) is predicted with the label 'CD player', while the attacked clean sample (b) after violent CSRP is predicted with the wrong label 'computer keyboard', which makes the model accuracy drop a lot. The corresponding LIME explanation is given in (e) and (f), respectively to show that the region that affects the model to predict 'CD player' is probably the play interface, while affecting the prediction of 'computer keyboard' is perhaps the whole body shape. Then, we

fine-tune the model using a limited amount of augmented clean data. Thus, the object that needs to be interpreted by LIME turns into the fine-tuned model. We get (g) and (h) as results. Also, conducting twice is to eliminate the random effect. From the results, we can notice the explanation region is restored around to its original extent with the correct label 'CD player'. This example indicates that the fine-tuned model trained can effectively re-identify the original features of clean samples which are destructed or dropped by CSRP and make the inference more convincing.

# 5 EVALUATION

## 5.1 Experimental Setup

**Models & Datasets.** In our experiments, we choose the following three well-known model architectures: ResNet-56 [15], VGG-16 [37], and MobileNet [31] to train the target and independent models. Also, we use three famous datasets for object classification tasks, including CIFAR-10 (noted as "C10") [19], CIFAR-100 (noted as "C100") [19], and TinyImageNet (noted as "TI") [21]. More details of these three datasets are in Table 1.

Table 1. Statistics of each dataset.

| Dataset | # of training data | # of test data | # of classes |
|---|---|---|---|
| C10 | 50,000 | 10,000 | 10 |
| C100 | 50,000 | 10,000 | 100 |
| TI | 100,000 | 10,000 | 200 |

We train one target model for each dataset and each architecture. For C10, we train 50 epochs to achieve around 90% top-1 ACC. For C100, we train 50 epochs and get around 85% top-5 ACC. For TI, the top-5 ACC is 75% after 50 epochs of training. More details about ACC are presented in Table 2, Table 3 and Table 4. Additionally, we train independent models for fingerprinting evaluation. Note here that the independent models are the most challenging cases, which refer to the models that use the same training dataset, structure, and hyper-parameters but only with different initial parameters. These independent models are similar to the target model but should be excluded from IP verification. We train 100 models from scratch with different initial parameters as independent models for each dataset and architecture. Thus, in this paper, we trained 900 models in total from 3 different datasets and three different model architectures[1]. We use Pytorch 1.10 backend and conduct the experiments on a server equipped with two Intel Xeon 2678 V3 CPUs and 8 NVIDIA GeForce RTX 3080 Ti GPUs.

**DNN Fingerprinting Reproduce.** We set the hyper-parameter by following the results in these published methods. For IPGuard, a parameter $k$ is used to balance between the robustness and uniqueness, and here we use $k = 5$ following their original implementation [5]. For CAE, a parameter $\epsilon$ is used to clip the intermediate perturbations around the $\epsilon$ ball of the original clean samples, and here we use $\epsilon = 0.01$ to follow the previous work. For RCE, RCE-GM, and RCE-LT, the parameter $\delta$ is used to bound the noise added to the neural network parameter, and here we choose $\delta = 0.001$ to implement. Besides, following the previous work [42], we sample input gradients for $q = 10$ times for GM. For RCE-LT only, the parameter $k$ is used to control the high-pass frequency band size. Note that we did try different hyper-parameters to generate different fingerprint samples, but our evaluation results will hold. For each target model, we generate a set of fingerprint samples $\mathcal{X}_f = \{(\mathbf{x}_i, y_i)\}_N$ with $N = 100$.

**Baselines.** We consider other possible approaches that an attacker will try to mitigate fingerprint samples by considering a comprehensive scenario, including compression, pruning, and fine-tuning. Note that these methods are generally considered as potential attacks on DNN fingerprinting methods in previous works [5, 27, 42]. We

[1]Note here only the training for these 900 independent models from scratch takes about 1,600 GPU hours.

perform the model pruning with rates of 0.1 and 0.5 to get pruned models. Note higher pruning rate will lead to a significant drop in accuracy on clean samples. We also compress with 8-bit to get quantized models. We apply fine-tuning for 16 epochs for the target models. We save the modified model every two epochs. We fine-tune the model 10 times independently under the same setting to avoid randomness during fine-tuning. There are 80 fine-tuned models for each target model for evaluation.

**Metrics.** We evaluate our method for effectiveness and utility-preserving. For the effectiveness, we calculate the *matching rate* (MR) $T(\widehat{f})$ (Eq. (3)) for $N$ fingerprint samples $\mathcal{X}_f$ by checking how much ratio of these fingerprint samples will be predicted as the pre-defined labels for our modified inference process $\widehat{f}(\cdot) = f(g(\cdot))$.

$$T(\widehat{f}) = \frac{1}{N} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{X}_f} \mathbb{1}(f(g(\mathbf{x_i}) = y_i)), \tag{3}$$

For utility-preserving, we use the accuracy on the clean dataset (ACC) of our modified inference process $\widehat{f}(\cdot)$. We list the top-1 accuracy score (noted as "ACC-top1") for the C10 dataset since it has only ten classes to classify. For the other two datasets (i.e. C100 has 100 classes and TI has 200 classes to classify), we list only the top-5 accuracy score (noted as "ACC-top5").

## 5.2 Attack Target DNN Fingerprinting Methods

Table 2. Evaluation of matching rate (MR) on C10 dataset for three model structures: ResNet-56, VGG-16, and MobileNet.

| Setting | Attack | | ACC-top1 | IPGuard | CAE | VCE | RCE | RCE-GM | RCE-LT |
|---|---|---|---|---|---|---|---|---|---|
| ResNet-56 CSRP $\sigma = 1.16$ $p = 3$ | Target model | | 0.87 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.87 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.87 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.87 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.69 | 0.52 | 0.89 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Independent | | [0.86,0.87] | [0.03,0.14] | [0.0,0.52] | [0.48,0.96] | [0.55,0.98] | [0.63,0.98] | [0.18,0.86] |
| | CSRP only | | 0.77 | 0.26 | **0.44** | 0.99 | 0.99 | **0.98** | 0.94 |
| | CSRP +fine-tuning | | 0.81 | **0.09** | **0.40** | 0.98 | **0.98** | **0.97** | 0.9 |
| VGG-16 CSRP $\sigma = 1.16$ $p = 3$ | Target model | | 0.91 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.91 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.91 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.91 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.89 | 0.71 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Independent | | [0.9,0.91] | [0.17,0.4] | [0.0,0.18] | [0.53,0.86] | [0.57,0.87] | [0.5,0.91] | [0.05,0.72] |
| | CSRP only | | 0.83 | **0.33** | 0.43 | **0.79** | **0.8** | **0.81** | **0.63** |
| | CSRP +fine-tuning | | 0.86 | **0.24** | 0.36 | **0.65** | **0.69** | **0.67** | **0.49** |
| MobileNet CSRP $\sigma = 1.16$ $p = 3$ | Target model | | 0.85 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.85 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.85 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.85 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.83 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Independent | | [0.84,0.86] | [0.0,0.04] | [0.0,0.65] | [0.11,0.75] | [0.13,0.7] | [0.12,0.71] | [0.0,0.33] |
| | CSRP only | | 0.77 | 0.14 | **0.42** | **0.6** | **0.59** | **0.57** | **0.18** |
| | CSRP +fine-tuning | | 0.79 | 0.06 | **0.47** | **0.69** | **0.67** | **0.65** | **0.33** |

We assume the IP verification will be performed via sending $N$ fingerprint samples ($N = 100$) and comparing the matching rate. We set one proper parameter of CSRP (e.g. $\sigma$ and $p$) for each dataset to present the attack results. Note that the CSRP has a random output for one same image. We test the matching rate $T(\widehat{f})$ with the set

Table 3. Evaluation of matching rate (MR) on C100 dataset for three model structures: ResNet-56, VGG-16, and MobileNet.

| Setting | Attack | | ACC-top5 | IPGuard | CAE | VCE | RCE | RCE-GM | RCE-LT |
|---|---|---|---|---|---|---|---|---|---|
| ResNet-56 CSRP $\sigma = 1.23$ $p = 5$ | Target model | | 0.88 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.88 | 0.97 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.88 | 1.0 | 0.98 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.88 | 1.0 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.76 | 0.27 | 0.67 | 0.98 | 0.97 | 0.96 | 0.8 |
| | Independent | | [0.87,0.89] | [0.0,0.03] | [0.03,0.34] | [0.01,0.12] | [0.01,0.12] | [0.01,0.1] | [0.0,0.06] |
| | CSRP only | | 0.69 | **0.03** | **0.12** | 0.14 | 0.14 | 0.15 | 0.08 |
| | CSRP +fine-tuning | | 0.81 | **0.01** | **0.14** | **0.1** | **0.09** | 0.11 | **0.04** |
| VGG-16 CSRP $\sigma = 1.23$ $p = 5$ | Target model | | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.89 | 1.0 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.89 | 0.92 | 0.92 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.89 | 0.66 | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Independent | | [0.89,0.9] | [0.0,0.02] | [0.0,0.51] | [0.02,0.2] | [0.04,0.21] | [0.04,0.23] | [0.0,0.1] |
| | CSRP only | | 0.73 | **0.0** | **0.12** | **0.08** | **0.08** | **0.07** | **0.04** |
| | CSRP +fine-tuning | | 0.84 | **0.01** | **0.21** | **0.12** | **0.12** | **0.11** | **0.04** |
| MobileNet CSRP $\sigma = 1.23$ $p = 5$ | Target model | | 0.83 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.83 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.83 | 1.0 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.83 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.8 | 0.82 | 0.91 | 1.0 | 1.0 | 1.0 | 0.98 |
| | Independent | | [0.82,0.84] | [0.0,0.03] | [0.02,0.14] | [0.01,0.11] | [0.01,0.12] | [0.02,0.11] | [0.0,0.05] |
| | CSRP only | | 0.64 | **0.03** | **0.04** | **0.1** | **0.09** | **0.1** | **0.04** |
| | CSRP +fine-tuning | | 0.76 | **0.01** | **0.05** | 0.15 | 0.13 | 0.15 | **0.02** |

of fingerprint samples ($X_f$) 100 times and calculate the average matching rate as $T_{avg} = \frac{1}{100}\sum_1^{100} T(\widehat{f}(X_f))$. CSRP can generate quite stable results as the variance of $T_{avg}$ is very limited. For those fine-tuned target models, we list the average matching rate of the 80 fine-tuned models since the variance is also minimal. For the pruned models, a larger pruning ratio (e.g. 0.5) may decrease the matching rate and significantly influence the ACC on clean samples.

To get the reference for comparison, We calculate the matching rate of the fingerprint sample set ($X_f$) on the 100 independent models and list the minimum and maximum matching rates. Note that the maximum matching rate on these independent models is still much less than the matching rate on target models or modified target models such that a proper threshold can be easily defined to accurately verify the IP from the independent models and the modified target models.

Therefore, we evaluate our effectiveness as follows: our attack succeeds as long as our method can always limit the matching rate ($T_{avg}$) for random transformed fingerprint samples less than the maximum matching rate on independent models. We list our evaluation results according to the used datasets as follows. (1) The C10 dataset on ResNet-56, VGG-16 and MobileNet is in Table 2. (2) The C100 dataset on ResNet-56, VGG-16 and MobileNet is in Table 3. (3) The TI dataset on ResNet-56, VGG-16 and MobileNet is in Table 4.

Although conducting CSRP only can effectively mitigate the fingerprint methods, ACC drops a lot which can not be used normally. Thus, in the data-limited scenario, we implement fine-tuning with augmented data. For C10, only no more than 10% of the original training dataset is enough. Furthermore, as for more classes in C100 and TI, we need 20% to achieve a better ACC. We should note that whether for C10, C100, or TI, only 1% of the original training dataset can significantly improve ACC. For instance, for C100 on ResNet-56, fine-tuning with 1% augmented data can improve ACC from 69% to 77%, but for better utility, we use 20% to indicate.

Table 4. Evaluation of matching rate (MR) on TI dataset for three model structures: ResNet-56, VGG-16, and MobileNet.

| Setting | Attack | | ACC-top5 | IPGuard | CAE | VCE | RCE | RCE-GM | RCE-LT |
|---|---|---|---|---|---|---|---|---|---|
| ResNet-56 CSRP $\sigma = 1.25$ $p = 14$ | Target model | | 0.74 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.74 | 1.0 | 0.97 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.74 | 1.0 | 0.98 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.74 | 1.0 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.59 | 0.27 | 0.87 | 0.99 | 0.99 | 0.99 | 1.0 |
| | Independent | | [0.74,0.75] | [0.0,0.03] | [0.0,0.23] | [0.03,0.17] | [0.03,0.16] | [0.03,0.19] | [0.0,0.08] |
| | CSRP only | | 0.53 | **0.01** | **0.12** | **0.07** | **0.08** | **0.08** | **0.02** |
| | CSRP +fine-tuning | | 0.68 | **0.02** | **0.18** | **0.16** | 0.18 | **0.18** | **0.06** |
| VGG-16 CSRP $\sigma = 1.25$ $p = 14$ | Target model | | 0.78 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.78 | 1.0 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.78 | 1.0 | 0.97 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.78 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.78 | 1.0 | 0.95 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Independent | | [0.78,0.79] | [0.0,0.02] | [0.02,0.49] | [0.02,0.18] | [0.02,0.13] | [0.02,0.17] | [0.0,0.08] |
| | CSRP only | | 0.62 | **0.02** | **0.11** | **0.1** | **0.1** | **0.09** | **0.04** |
| | CSRP +fine-tuning | | 0.74 | **0.01** | **0.22** | **0.12** | 0.14 | **0.15** | **0.06** |
| MobileNet CSRP $\sigma = 1.25$ $p = 14$ | Target model | | 0.77 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Compression | | 0.77 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Fine-tuning | | 0.77 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Pruning | 0.1 | 0.77 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| | | 0.5 | 0.72 | 0.86 | 0.99 | 1.0 | 1.0 | 1.0 | 1.0 |
| | Independent | | [0.76,0.78] | [0.0,0.01] | [0.0,0.1] | [0.04,0.15] | [0.02,0.14] | [0.03,0.15] | [0.0,0.06] |
| | CSRP only | | 0.57 | **0.0** | **0.02** | **0.04** | **0.06** | **0.05** | **0.01** |
| | CSRP +fine-tuning | | 0.71 | **0.01** | **0.05** | 0.18 | 0.21 | 0.18 | **0.06** |

By losing about 3-6% ACC in the data-limited scenario, we can effectively mitigate the fingerprinting methods by compromising the prediction of fingerprint samples. Moreover, our method is significantly better than all baseline methods for modifying the target models.

## 5.3 Ablation Study

**Comparison with AE defense methods.** We note the similarity between the fingerprint samples and the adversarial examples (AE) [50], which both aim to mislead one sample to a wrong pre-defined label. However, two major differences exist between mitigating fingerprint samples and defending AEs. (1) The attack on fingerprinting methods can be seen as a success as long as the predicted label for one fingerprint sample is inconsistent with its pre-defined label. This point makes mitigating fingerprint samples much easier since a successful defense on AEs usually requires the correct labels to be predicted for AEs [6]. (2) However, fingerprint samples have no constraints of perturbation levels (e.g. L2-norm or Linf-norm [6]) because imperceptibility is not the dominant requirement for IP protection, and contrary to AEs, the generated samples' need to have low transferability . Therefore, the features carried by these samples are normally more robust and harder to be removed by preprocessing-only solutions.

We include a preprocessing-only AE defense method, random distortion over grids (RDG) [34] to compare with CSRP-only in the data-free scenario. RDG can effectively mitigate various AEs and classify the correct labels. We choose one case (C10 and VGG-16) as the test case. The results are given in Table 5 by limiting the same ACC drop to see the mitigation results. Since RDG cannot give stable results for 100 tests, we list the minimum and maximum matching rates of RDG and CSRP-only respectively to compare. It is clear that compared with CSRP that always gets a lower matching rate than the maximum matching rate of independent models, the RDG could

give a very high matching rate for all the fingerprinting methods. Thus, RDG cannot be used to mitigate the DNN fingerprinting.

Table 5. Matching rate by comparing RDG with CSRP on 100 fingerprint samples on C10 dataset, VGG-16.

| Attack | ACC-top1 | IPGuard | CAE | VCE | RCE | RCE-GM | RCE-LT |
|---|---|---|---|---|---|---|---|
| Target model | 0.91 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| Independent | [0.9,0.91] | [0.17,0.4] | [0.0,0.18] | [0.53,0.86] | [0.57,0.87] | [0.5,0.91] | [0.05,0.72] |
| RDG | 0.83 | [0.15,0.7] | [0.37,0.93] | [0.65,1.0] | [0.57,1.0] | [0.63,1.0] | [0.42,0.98] |
| CSRP only | 0.83 | [0.25,0.42] | [0.32,0.53] | [0.73,0.85] | [0.74,0.87] | [0.72,0.88] | [0.54,0.69] |

Table 6. Evaluation of hyper-parameters on C100 dataset, ResNet-56 for the data-limited scenario.

| Settings | | ACC-top5 | IPGuard | CAE | VCE | RCE | RCE-GM | RCE-LT |
|---|---|---|---|---|---|---|---|---|
| Target model | | 0.88 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| Independent | | [0.87,0.89] | [0.0,0.03] | [0.03,0.34] | [0.01,0.12] | [0.01,0.12] | [0.01,0.1] | [0.0,0.06] |
| CSRP + fine-tuning | $\sigma = 1.16, p = 3, r = 0.2$ | **0.83** | **0.03** | **0.2** | 0.17 | 0.18 | 0.21 | 0.07 |
| | $\sigma = 1.23, p = 5, r = 0.2$ | 0.81 | **0.01** | **0.14** | **0.1** | **0.09** | 0.11 | **0.04** |
| | $\sigma = 1.23, p = 5, r = 0.5$ | **0.83** | **0.01** | **0.15** | 0.13 | **0.12** | 0.13 | **0.03** |

**Evaluation on hyper-parameters.** To further evaluate a trade-off between the attack effectiveness and utility-preserving, we reset different $\sigma$ and $p$ for each dataset. To achieve better mitigation on fingerprint samples, we choose larger $\sigma$ and $p$ for CSRP, which will as well make errors in clean samples' predictions (Referring to the results in Table 2, Table 3 and Table 4). We should note that when we choose smaller $\sigma = 1.16$ and $p = 3$ for C100 as an example, we will reach a much higher model accuracy to 83% for ResNet-56, as given in Table 6. Note that we only choose one case (C100 and ResNet-56) as the test case; other cases also hold. Although the $T(\widehat{f})$ at this time (e.g.17% for VCE, 18% for RCE, and 7% for RCE-LT) will be larger than the maximum matching rate of independent models(e.g.12% for VCE, 12% for RCE and 6% for RCE-LT), they are still far lower than the standard matching rates 100% and the baselines that are close to 100%, which is also acceptable for mitigating purpose. Moreover, in the data-limited scenario, fine-tuning also set two other hyper-parameters to balance this trade-off. One is the ratio ($r$) of the number of training data to the number of the original training dataset; the other one is the fine-tuning epochs. For instance, in Table 3, we initially use the 20% C100 dataset to fine-tune ResNet-56 for 6-8 epochs and get an accuracy of 81%. However, when we use 50% C100 dataset to fine-tune for only four epochs, we will get 83% accuracy, and the $T(\widehat{f})$ for all types of fingerprint samples will only increase by less than 1%, as shown in Table 6. Thus, choosing different hyperparameters will result in different goals, and it is important to balance all the sides.

## 6 DISCUSSION AND FUTURE WORK

**Trade-off between ASR and ACC.** The fingerprint samples have a very low matching rate on those independent models, which makes our attack hard to achieve the effectiveness goal in some cases. However, if we continue to use more intense transformation (e.g. by increasing the parameter $\sigma$ and $p$), we can still mitigate those fingerprint samples but with more ACC loss. For instance, in the C100 and MobileNet data-limited case, we can reduce the $T(\widehat{f})$ to 0.09 for VCE (lower than the maximum matching rate 0.11 for independent models) with larger scaling and padding parameter as $\sigma = 1.40$ and $p = 9$. However, the ACC-top5 will drop to 0.70, which loses 0.13 compared with the clean model. Indeed, a higher randomized transformation of CSRP will lead to more effectively mitigating fingerprinting-based IP verification but definitely will cause more ACC drop on clean samples. In

summary, choosing parameters will tune the gradient obfuscation level, which leverages a trade-off between attack effectiveness (ASR) and utility-preserving (ACC). We will consider a theoretical explanation for a more precise understanding of this trade-off as our first future work.

**Future adaptive fingerprinting.** Current DNN fingerprinting methods mainly consider only modification on the illegally acquired models but are vulnerable to data augmentation-based approaches, as indicated in our paper. We will explore how to adaptively design novel DNN fingerprinting methods from a defender's perspective to mitigate such data augmentation-based attacks as our second future work.

## 7 CONCLUSION

In this paper, we first mitigated various existing query-based DNN fingerprinting methods for IP protection from a data augmentation perspective. We showed that although these existing DNN fingerprinting methods are robust to various model modifications, they are quite vulnerable to gradient obfuscation-based preprocessing methods. Specifically, we designed a preprocessing-based method `CSRP` to randomly transform all input samples, which can not only effectively compromise the fingerprint samples' verification but also preserve an acceptable model utility. By deploying `CSRP`, the adversary can use an illegally acquired DNN model to provide inference service for a profit with or without any clean dataset and escape various query-based DNN fingerprinting methods to verify the IP.



## REFERENCES


[1] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *Proc. of the USENIX Security*. 1615–1631.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*. PMLR, 274–283.

[3] Jiawang Bai, Baoyuan Wu, Yong Zhang, Yiming Li, Zhifeng Li, and Shu-Tao Xia. 2021. Targeted Attack against Deep Neural Networks via Flipping Limited Weight Bits. *ICLR* (2021).

[4] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 535–541.

[5] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2021. IPGuard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 14–25.

[6] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proc. of the S&P*. 39–57.

[7] Abhishek Chakraborty, Ankit Mondai, and Ankur Srivastava. 2020. Hardware-assisted intellectual property protection of deep learning models. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[9] Bowei Dong, Qiongfeng Shi, Yanqin Yang, Feng Wen, Zixuan Zhang, and Chengkuo Lee. 2021. Technology evolution from self-powered sensors to AIoT enabled smart homes. *Nano Energy* 79 (2021), 105414.

[10] Lixin Fan, Kam Woh Ng, and Chee Seng Chan. 2019. Rethinking deep neural network ownership verification: Embedding passports to defeat ambiguity attacks. *Advances in Neural Information Processing Systems* 32 (2019).

[11] Faezeh Farivar, Mohammad Sayad Haghighi, Soheila Barchinezhad, and Alireza Jolfaei. 2019. Detection and compensation of covert service-degrading intrusions in cyber physical systems through intelligent adaptive control. In *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 1143–1148.

[12] Faezeh Farivar, Mohammad Sayad Haghighi, Alireza Jolfaei, and Sheng Wen. 2021. Covert attacks through adversarial learning: Study of lane keeping attacks on the safety of autonomous vehicles. *IEEE Transactions on Mechatronics* 26, 3 (2021), 1350–1357.

[13] Hui Guan, Shaoshan Liu, Xiaolong Ma, Wei Niu, Bin Ren, Xipeng Shen, Yanzhi Wang, and Pu Zhao. 2021. Cocopie: Enabling real-time AI on off-the-shelf mobile devices via compression-compilation co-design. *Commun. ACM* 64, 6 (2021), 62–68.

[14] Jia Guo and Miodrag Potkonjak. 2018. Watermarking deep neural networks for embedded systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[16] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High accuracy and high fidelity extraction of neural networks. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 1345–1362.
[17] Weixiong Jiang, Heng Yu, Xinzhe Liu, Hao Sun, Rui Li, and Yajun Ha. 2021. TAIT: One-shot full-integer lightweight DNN quantization via tunable activation imbalance transfer. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1027–1032.
[18] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2021. Alias-free generative adversarial networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*.
[19] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.
[20] Souvik Kundu, Mahdi Nazemi, Peter A Beerel, and Massoud Pedram. 2021. DNR: A Tunable Robust Pruning Framework Through Dynamic Network Rewiring of DNNs. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 344–350.
[21] Ya Le and Xuan Yang. 2015. Tiny imagenet visual recognition challenge. *CS 231N* 7, 7 (2015), 3.
[22] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. 2019. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications* (2019), 1–12.
[23] Chuan Li. 2020. OpenAI's GPT-3 Language Model: A Technical Overview. https://lambdalabs.com/blog/demystifying-gpt-3/.
[24] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor learning: A survey. *arXiv preprint arXiv:2007.08745* (2020).
[25] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. 2019. How to prove your model belongs to you: A blind-watermark based framework to protect intellectual property of DNN. In *Proceedings of the 35th Annual Computer Security Applications Conference*. 126–137.
[26] Wei Liang, Yiyong Hu, Xiaokang Zhou, Yi Pan, I Kevin, and Kai Wang. 2021. Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT. *IEEE Transactions on Industrial Informatics* 18, 8 (2021), 5087–5095.
[27] Nils Lukas, Yuxuan Zhang, and Florian Kerschbaum. 2021. Deep Neural Network Fingerprinting by Conferrable Adversarial Examples. In *ICLR*.
[28] Xiaolong Ma, Geng Yuan, Sheng Lin, Caiwen Ding, Fuxun Yu, Tao Liu, Wujie Wen, Xiang Chen, and Yanzhi Wang. 2020. Tiny but accurate: A pruned, quantized and optimized memristor crossbar framework for ultra efficient dnn implementation. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 301–306.
[29] Jingshu Peng, Zhao Chen, Yingxia Shao, Yanyan Shen, Lei Chen, and Jiannong Cao. 2022. Sancus: sta le n ess-aware c omm u nication-avoiding full-graph decentralized training in large-scale graph neural networks. *Proceedings of the VLDB Endowment* 15, 9 (2022), 1937–1950.
[30] Zirui Peng, Shaofeng Li, Guoxing Chen, Cheng Zhang, Haojin Zhu, and Minhui Xue. 2022. Fingerprinting Deep Neural Networks Globally via Universal Adversarial Perturbations. *arXiv preprint arXiv:2202.08602* (2022).
[31] Zheng Qin, Zhaoning Zhang, Xiaotao Chen, Changjian Wang, and Yuxing Peng. 2018. Fd-mobilenet: Improved mobilenet with a fast downsampling strategy. In *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 1363–1367.
[32] Han Qiu, Hassan Noura, Meikang Qiu, Zhong Ming, and Gerard Memmi. 2019. A user-centric data protection method for cloud storage based on invertible DWT. *IEEE Transactions on Cloud Computing* (2019).
[33] Han Qiu, Yi Zeng, Shangwei Guo, Tianwei Zhang, Meikang Qiu, and Bhavani Thuraisingham. 2021. Deepsweep: An evaluation framework for mitigating dnn backdoor attacks using data augmentation. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. 363–377.
[34] Han Qiu, Yi Zeng, Qinkai Zheng, Shangwei Guo, Tianwei Zhang, and Hewu Li. 2021. An Efficient Preprocessing-based Approach to Mitigate Advanced Adversarial Attacks. *IEEE Trans. Comput.* (2021).
[35] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1135–1144.
[36] Andriy Sarabakha and Erdal Kayacan. 2019. Online deep learning for improved trajectory tracking of unmanned aerial vehicles using expert knowledge. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 7727–7733.
[37] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[38] Linda Tessens, Marleen Morbee, Hamid Aghajan, and Wilfried Philips. 2014. Camera selection for tracking in distributed smart camera networks. *ACM Transactions on Sensor Networks (TOSN)* 10, 2 (2014), 1–33.
[39] Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. 2017. Embedding watermarks into deep neural networks. In *Proc. of the ICMR*. 269–277.
[40] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. 2020. Practical detection of trojan neural networks: Data-limited and data-free cases. In *European Conference on Computer Vision*. Springer, 222–238.
[41] Si Wang and Chip-Hong Chang. 2021. Fingerprinting deep neural networks-a deepfool approach. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[42] Siyue Wang, Xiao Wang, Pin-Yu Chen, Pu Zhao, and Xue Lin. 2021. Characteristic Examples: High-Robustness, Low-Transferability Fingerprinting of Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*. 575–582.

[43] Zirui Xu, Fuxun Yu, Chenchen Liu, and Xiang Chen. 2019. Reform: Static and dynamic resource-aware dnn reconfiguration framework for mobile device. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.
[44] Mingfu Xue, Jian Wang, and Weiqiang Liu. 2021. DNN intellectual property protection: Taxonomy, attacks and evaluations. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*. 455–460.
[45] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *Proc. of the AsiaCCS*. 159–172.
[46] Xiaokang Zhou, Yiyong Hu, Jiayi Wu, Wei Liang, Jianhua Ma, and Qun Jin. 2022. Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial iot. *IEEE Transactions on Industrial Informatics* 19, 1 (2022), 570–580.
[47] Xiaokang Zhou, Wei Liang, I Kevin, Kai Wang, and Laurence T Yang. 2020. Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations. *IEEE Transactions on Computational Social Systems* 8, 1 (2020), 171–178.
[48] Xiaokang Zhou, Wei Liang, Weimin Li, Ke Yan, Shohei Shimizu, I Kevin, and Kai Wang. 2021. Hierarchical adversarial attacks against graph neural network based IoT network intrusion detection system. *IEEE Internet of Things Journal* (2021).
[49] Yuankun Zhu, Yueqiang Cheng, Husheng Zhou, and Yantao Lu. 2021. Hermes Attack: Steal DNN Models with Lossless Inference Accuracy. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.
[50] Giulio Zizzo, Chris Hankin, Sergio Maffeis, and Kevin Jones. 2019. Adversarial machine learning beyond the image domain. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–4.