Teacher Model Fingerprinting Attacks Against Transfer Learning

Yufei Chen, *Xi'an Jiaotong University & City University of Hong Kong;* Chao Shen, *Xi'an Jiaotong University;* Cong Wang, *City University of Hong Kong;* Yang Zhang, *CISPA Helmholtz Center for Information Security*

https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yufei

This paper is included in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

Teacher Model Fingerprinting Attacks Against Transfer Learning

Yufei Chen[1,2] Chao Shen[1] Cong Wang[2] Yang Zhang[3]

[1]*Xi'an Jiaotong University* [2]*City University of Hong Kong*
[3]*CISPA Helmholtz Center for Information Security*

Abstract

Transfer learning has become a common solution to address training data scarcity in practice. It trains a specified student model by reusing or fine-tuning early layers of a well-trained teacher model that is usually publicly available. However, besides utility improvement, the transferred public knowledge also brings potential threats to model confidentiality, and even further raises other security and privacy issues.

In this paper, we present the first comprehensive investigation of the teacher model exposure threat in the transfer learning context, aiming to gain a deeper insight into the tension between public knowledge and model confidentiality. To this end, we propose a *teacher model fingerprinting attack* to infer the origin of a student model, i.e., the teacher model it transfers from. Specifically, we propose a novel optimization-based method to carefully generate queries to probe the student model to realize our attack. Unlike existing model reverse engineering approaches, our proposed fingerprinting method neither relies on fine-grained model outputs, e.g., posteriors, nor auxiliary information of the model architecture or training dataset. We systematically evaluate the effectiveness of our proposed attack. The empirical results demonstrate that our attack can accurately identify the model origin with few probing queries. Moreover, we show that the proposed attack can serve as a stepping stone to facilitating other attacks against machine learning models, such as model stealing.[1]

1 Introduction

The past decade has witnessed an unprecedented development of machine learning (ML). Yet, the progress of ML heavily relies on sophisticated models, sufficient computing resources, and a massive volume of training data, which remain major constraints to building high-performance ML models.

Transfer learning opens a pathway for overcoming obstacles raised by the lack of data or computing resources; it is essentially an ML paradigm to transfer the knowledge from a well-learned domain into a specified domain where training data are scarce. Concretely, transfer learning establishes a new model (a.k.a. *student model*) through borrowing early layers from a pre-trained model (a.k.a. *teacher model*), which requires far less efforts than training from scratch. It has been proved to be a promising ML practice and widely applied in a wide range of academic and industrial areas, such as computer vision [28], natural language processing [50], etc.

Despite the huge success, such cross-domain knowledge learning paradigm also raises security and privacy concerns:

- There exist many advanced ML models, each of which can potentially serve as a teacher model for transfer learning. Choosing an appropriate teacher model to train a student model requires a large number of engineering efforts. Thus, for a student model, the choice of its teacher model certainly belongs to the model owner's intellectual property (IP), and should be kept confidential.
- On the other side of the coin, from the responsible ML perspective, a teacher model owner does not want their model to be transferred to perform unethical or illegal tasks, such as facial recognition or weapon classification. Thus, a teacher model owner needs a way to track the parties that use their model to build student models.
- Malicious parties can intentionally publish vulnerable ML models online. When such models are used as teacher models for transfer learning, the corresponding student models may inherit some vulnerabilities. For instance, Zhang et al. demonstrate that fine-tuned transfer learning models are more prone to transferable adversarial examples [48]. Yao et al. show that an attacker can infect transfer learning models by distributing pre-trained models with backdoors [45].
- Also, after learning the teacher model that a student is transferred from, an attacker can perform more effective malicious attacks, such as adversarial example attacks [11] and model stealing/extraction attacks [40].

All these concerns require us to gain a deeper insight into

[1]Our code is available at https://github.com/yfchen1994/Teacher-Fingerprinting.

the teacher model exposure or transfer learning.

In this paper, we present the first comprehensive investigation of the teacher model exposure threat in the transfer learning context. In particular, we propose a novel teacher model fingerprinting attack that can effectively identify the teacher model of a transferred student. The key idea is to generate a set of *fingerprinting pairs* for each teacher model candidate, which will activate similar latent feature representations. Hopefully, most of them will also trigger similar latent features and bring a pair of similar responses to the student model, if the student model is transferred from the teacher model candidate. We formalize the query generation process as an optimization problem, which can be solved via gradient descent. Note that, the fingerprinting pairs for each teacher model can be applied to all student models transferred from the teacher model, i.e., they are reusable. This point demonstrates the efficiency and practicality of our method.

We conduct extensive experiments to investigate the effectiveness of our proposed method, and thoroughly evaluate how various factors affect attack performance. The evaluation shows that our attack can achieve high teacher model inference accuracy even with a limited number of queries in top-1 label exposure. Moreover, our attack still works well when synthesizing queries from samples unrelated to the target domain, or even from random noises. In addition, we show that our fingerprinting attack can serve as a stepping stone to performing more effective model stealing attacks.

Our study should not be confused with recent research efforts on model stealing or model reverse engineering:

Their primary goal is to directly steal the exact victim model, build a surrogate model with similar functionality, or infer other internal model information. By contrast, our proposed attack essentially focuses on exploiting the model sharing practice in transfer learning, to quickly identify the presence of pre-trained components. Once identifying the teacher model components, the attacker has stepped further to realize the aforementioned malicious or unethical goals.

Besides, from the technical perspective, our attack remedies the key weakness—significance attack expense caused by massive attack queries or shadow models—of prior arts. Our attack just needs to identify the transferred feature map with few queries, rather than recovering model parameters or architectures interactively.

Furthermore, previous studies are subject to extra attack assumptions, such as transparent model architectures [9] and fine-grained model outputs [40]. E.g., the teacher model fingerprinting method proposed by Wang et al. [42] requires access to posteriors of the target model prediction. However, in the real world, raw model outputs are usually perturbed or hidden for security [20] or other product deployment concerns. Instead, our work is set up on a more generic and realistic scenario. We assume the attacker exclusively receives the victim black-box model's final decision, i.e., the top-1 classification result, which gives the minimum amount of information. This assumption implies that our proposed attack is likely to be mounted on any transfer learning classifiers. Additionally, we assume the attacker has zero knowledge of the victim's training dataset, but they are allowed to utilize public data. These assumptions ensure our proposed method is feasible and applicable in practice.

Our main contributions can be summarized as follows:

- We take the first step to comprehensively investigate the teacher model exposure in the transfer learning context. In particular, we demonstrate a teacher model fingerprinting attack against transfer learning.
- We propose a novel optimization-based technique to implement our attack.
- Extensive evaluations demonstrate the efficacy of our method, and we further show that our teacher model fingerprinting attack can be used to facilitate model stealing attacks against ML models.

2 Preliminaries

2.1 Problem Statement

Transfer learning. Transfer learning is commonly used to solve the data-hungry problem of ML, especially for deep learning models [22, 34, 39]. The core is to leverage the feature maps learned by the teacher model, so as to avoid significant cost of training a large model from scratch.

In the source domain, the teacher model $\mathcal{T}$ has been trained on a large scale dataset $\mathcal{D}_\mathrm{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_\mathrm{T}}$,[2] and then it is released for other downstream ML developers. In a typical transfer learning workflow, the student model $\mathcal{S}$ is firstly constructed with early k layers copied from $\mathcal{T}$ for feature extraction, as well as newly added classification layers $\mathcal{C}_\mathrm{S}(\cdot)$ (e.g., fully connected layers, SVM, etc.) on the top. Then, $\mathcal{S}$ gets trained on a usually confidential dataset $\mathcal{D}_\mathrm{S} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N_\mathrm{S}}$, where $N_\mathrm{S} \ll N_\mathrm{T}$. In practice, there are two training strategies:

- **Feature extractor**, where all pre-trained layers are frozen to compose a feature extractor $\mathcal{F}_\mathrm{T}$.
- **Fine-tuning**, where part of the pre-trained parameters of $\mathcal{F}_\mathrm{T}$ will be updated for better fitting on $\mathcal{D}_\mathrm{S}$.

After being carefully trained, $\mathcal{S}$ is deployed into a Machine-Learning-as-a-service (MLaaS) platform to provide services. A customer can send a query $\mathbf{x}$ to the MLaaS service and get the corresponding result $f(\mathcal{S}(\mathbf{x}))$, via an authorized API $f(\cdot)$.

Teacher model fingerprinting. In this paper, we develop a novel approach to infer the origin of a student model with only black-box access. Our motivation is that the student model is likely to inherit fingerprintable model behaviors, i.e., the way to extract features in our case (a.k.a. *feature map*), from the teacher model. Some attackers can possibly elicit such fingerprintable behaviors by sending carefully crafted queries to the student model, which is sometimes concealed into an

[2] In this paper, the notation $\{\cdot\}$ refers to a *multiset*, which may contain repeated values.

MLaaS platform. Consequently, they are able to identify the teacher model based on the victim's responses. We refer to this process as *teacher model fingerprinting attacks*.

2.2 Threat Model

Attack motivations. We first list potential motivations to conduct the teacher model fingerprinting attack.

Breaking model confidentiality. Internal model setup, such as architecture or parameters, is supposed to be kept confidential for the sake of IP protection or security consideration. However, the model behind an MLaaS platform is no longer a secret once the teacher model has been identified. The attacker can effortlessly recover early layers with the publicly transparent teacher model information.

Stepping stone to advanced attacks. As mentioned before, our teacher model fingerprinting attack provides a cheap way to recover most parameters inherited from the teacher model, which helps to open the "black box." Once opening the black box, the attacker has stepped further to discover and exploit vulnerabilities of the student model. For instance, they can easily conduct various white-box adversarial attacks against the victim model, such as adversarial example attacks [10] and membership inference attacks [24]. Also, recent studies have shown that transfer learning may transfer vulnerabilities from the teacher [48], which the attacker can directly exploit.

Forensics. Despite the aforementioned malicious goals, we can also treat our teacher model fingerprinting attack as a new forensic tool for ML applications. For instance, a teacher model owner does not want their model to be transferred to perform unethical or illegal tasks, such as weapon classification. A potential IP protection solution is to embed watermarks to model [8, 25, 47]. However, we are witnessing an arms race right now, where watermarking schemes would be broken by new attacks [29]. Our fingerprinting approach can complement watermarking for IP protection. Also, in some cases, certain parties can intentionally publish a vulnerable teacher model, which will result in the corresponding student models inheriting vulnerabilities [45]. For both of these cases, our attack can be used by a third-party forensic service as a defense to track the origin of a student model.

Threat model in detail. Then, we describe the detailed threat model in real-world settings. For our teacher model fingerprinting attack, we consider an attacker with black-box access to the victim student model $\mathcal{S}$. The attacker is able to send an arbitrary input $\mathbf{x}$ to $\mathcal{S}$ via an authorized API $f(\cdot)$ and receive the corresponding response $f(\mathcal{S}(\mathbf{x}))$. Three cases may arise.

- **Case 1.** $f(\mathcal{S}(\mathbf{x})) = \mathcal{S}(\mathbf{x})$. In a low-level security setting, the response exposes the exact model output, which contains class labels and raw confidence values.
- **Case 2.** $f(\mathcal{S}(\mathbf{x})) = \mathcal{S}(\mathbf{x}) + \varepsilon$. The MLaaS provider returns a perturbed version of the model output to avoid privacy breaches like membership inference attacks [20]. It is notable that the perturbation should not change the top-1 predicted label [20], i.e., $\arg\max_i (\mathcal{S}(\mathbf{x}) + \varepsilon)_i == \arg\max_i \mathcal{S}(\mathbf{x})_i$ ($S(\mathbf{x})_i$: the i-th component of $S(\mathbf{x})$).
- **Case 3.** $f(\mathcal{S}(\mathbf{x})) = \arg\max_i \mathcal{S}(\mathbf{x})_i$. Only the top-1 label is returned, giving the minimal piece of information [26].

This work demonstrates the teacher model fingerprinting attack in the most restrictive case—top-1 label exposure, i.e., Case 3 where $f(\mathbf{x}) = \arg\max_i \mathcal{S}(\mathbf{x})_i$.[3] In this case, our proposed attack is compatible with Case 2: on the one hand, the output perturbation in Case 2 would not change the top-1 predicted label; one the other hand, even the perturbed model output at least provides information no less than Case 3.

In our basic setup, we assume the attacker has obtained most potential teacher model candidates $\{\mathcal{T}_i\}$ from public resources (e.g., ML frameworks like PyTorch, or websites like ModelZoo [1]). We also consider possible cases where the victim model does not come from one of the candidate teacher models, or it is not trained through transfer learning. Neither the teacher dataset $\mathcal{D}_\mathrm{T}$ in the source domain nor the student dataset $\mathcal{D}_\mathrm{S}$ in the target domain is available. However, the attacker can collect public datasets like ImageNet to help perform the teacher model fingerprinting attack. The attacker has two primary goals: one is to infer the teacher model accurately, while the other is to use as few probing queries as possible, to limit attack costs and keep the attack stealthy.

3 Teacher Model Fingerprinting

3.1 Overview

In the beginning, we formalize the teacher model fingerprinting attack in the transfer learning context:

Definition 3.1 (Teacher Model Fingerprinting Attack)**.** *Suppose there is an attacker given a set of N realistic inputs $\{\mathbf{x}_i\}$ (a.k.a. probing input), a set of teacher model candidates $\{\mathcal{T}_j\}$, and an authorized API $f(\cdot)$ to the target black-box student model $\mathcal{S}$ in only top-1 label exposure, the teacher model fingerprinting attack is to infer which $\mathcal{T}_j$ is adopted by $\mathcal{S}$.*

We start from the feature extractor transfer learning setting. Suppose there is a student model $\mathcal{S}$ composed of a feature extractor $\mathcal{F}_\mathrm{T}$, with early k components copied from the teacher model $\mathcal{T}$, and newly trained classification layers $\mathcal{C}_\mathrm{S}$. Our intuition is that given a *probing input* $\mathbf{x}$, we can artificially craft a *synthetic input* $\mathbf{x}'$ so that $\mathcal{F}_\mathrm{T}(\mathbf{x}') \approx \mathcal{F}_\mathrm{T}(\mathbf{x})$. Hopefully, we will activate similar outputs on the student model as $\mathcal{S}(\mathbf{x}') = \mathcal{C}_\mathrm{S}(\mathcal{F}_\mathrm{T}(\mathbf{x}')) \approx \mathcal{C}_\mathrm{S}(\mathcal{F}_\mathrm{T}(\mathbf{x})) = \mathcal{S}(\mathbf{x})$, and therefore have a greater chance to receive similar responses from the API. In this case, we refer to the pair $\langle \mathbf{x}, \mathbf{x}' \rangle$ as a *fingerprinting pair*. Although the above intuition stands on the feature extractor scenario, in Section 4.3.3, we will show that our attack still works in the fine-tuning setting.

Figure 1 provides a schematic view of our teacher model fingerprinting attack. Overall, it follows two steps:

[3] In the remaining, we abuse the notation $f(\mathbf{x})$ to represent $f(\mathcal{S}(\mathbf{x}))$.

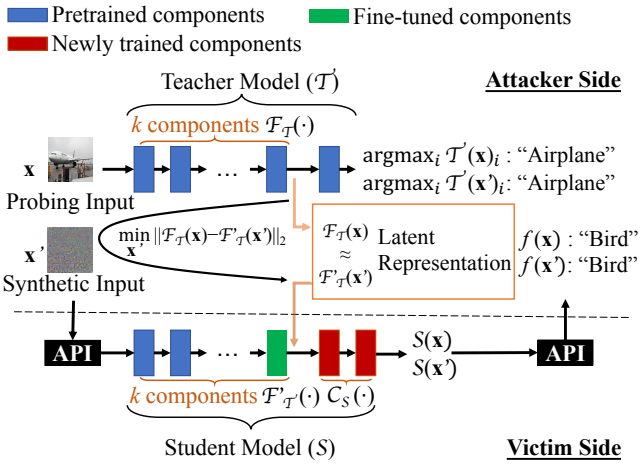


Figure 1: Illustration of our fingerprinting attack. Given a probing input $\mathbf{x}$ and a teacher model $\mathcal{T}$, the attacker aims to find a synthetic input $\mathbf{x}'$, which will activate a similar latent representation at the k-th component with $\mathbf{x}$: $\mathcal{F}_{\mathrm{T}}(\mathbf{x}') \approx \mathcal{F}_{\mathrm{T}}(\mathbf{x})$. If the first k components of the student model $\mathcal{S}$ comes from the teacher model $\mathcal{T}$, some of which may get slightly fine-tuned (still, $\mathcal{F}'_{\mathrm{T}}(\cdot) \approx \mathcal{F}_{\mathrm{T}}(\cdot)$), the student model is likely to produce a pair of similar responses $\mathcal{S}(\mathbf{x}') \approx \mathcal{S}(\mathbf{x})$.

Step I: generating a set of fingerprinting pairs $\{\langle \mathbf{x}_i, \mathbf{x}'_i \rangle\}$. To perform the teacher model fingerprinting attack, the attacker firstly collects a set of candidate teacher models and realistic probing images. For each candidate teacher model $\mathcal{T}_j$, the attacker needs to determine which components may constitute the feature extractor $\mathcal{F}_{\mathcal{T}_j}$ for the student model. Conventionally, the student model will adopt the whole convolution part from the teacher model. Then, for each probing input $\mathbf{x}_i$, the attacker aims to craft a synthetic input $\mathbf{x}'_i$, which will trigger similar latent representations on $\mathcal{F}_{\mathcal{T}_j}$. We will study how the choice of components of $\mathcal{F}_{\mathcal{T}_j}$, the number of $\{\mathbf{x}_i\}$, as well as the source of $\{\mathbf{x}_i\}$ affect the proposed attack in Section 4.3.2, Section 4.3.4, and Section 4.3.5, respectively.

This attack query generation procedure ensures our attack is a cheap one, since: i) the query generation procedure is locally conducted without any expense by the target MLaaS service, and ii) for fingerprinting pairs bonded to $\mathcal{F}_{\mathcal{T}_j}$, they are reusable to probe whether other student models come from $\mathcal{F}_{\mathcal{T}_j}$, without further computation costs on query generation.

Step II: inferring the teacher model according to black-box responses $\{\langle y_i, y'_i \rangle\}$. After obtaining fingerprinting pairs, the next step is to send them to the target black box, and infer which teacher model is used according to the responses. Ideally, if the target model relies on the feature extractor $\mathcal{F}_{\mathcal{T}_j}$, most generated fingerprinting pairs would produce a pair of responses matched on the same label. The inference stage follows the "one-of-the-best" strategy: choosing the candidate owning the most matched responses as the inference result. In Section 5, we will introduce a strategy for robust inferences.

3.2 Step I: Attack Query Generation

For our teacher model fingerprinting attack, we use the L_2-norm metric to measure the similarity of latent representations. Given a probing input $\mathbf{x}$ and a candidate feature extractor $\mathcal{F}_{\mathrm{T}}$, we formalize the attack query generation task as

$$\mathbf{x}' = \arg\min_{\tilde{\mathbf{x}}} \|\mathcal{F}_{\mathrm{T}}(\tilde{\mathbf{x}}) - \mathcal{F}_{\mathrm{T}}(\mathbf{x})\|_2, \text{ s.t. } \tilde{\mathbf{x}} \in [0, 255]. \tag{1}$$

For the constrained optimization problem Equation 1, we adopt the optimization strategy proposed by Carlini and Wagner [11]: by introducing a new variable $\mathbf{w}$, and setting

$$\tanh(\mathbf{w}) = \frac{2\tilde{\mathbf{x}}}{255} - 1, \tag{2}$$

we have $-1 \le \tanh(\mathbf{w}) \le 1$, and

$$-1 \le \frac{2\tilde{\mathbf{x}}}{255} - 1 \le 1 \Rightarrow 0 \le \tilde{\mathbf{x}} \le 255, \tag{3}$$

which satisfies the constraints. Therefore, the original problem Equation 1 can be converted to

$$\mathbf{w}' = \arg\min_{\mathbf{w}} \left\| \mathcal{F}_{\mathrm{T}} \left(\frac{255}{2} (\tanh(\mathbf{w}) + 1) \right) - \mathcal{F}_{\mathrm{T}}(\mathbf{x}) \right\|_2, \tag{4}$$

which can be solved by gradient descent algorithms. We use the Adam optimizer [21], with learning rate set as 10^{-3} for 30,000 iterations. In our implementation, we initialize $\mathbf{w}$ as $\mathbf{0}$.

However, due to type casting (floating points to integers) and optimization loss, $\mathcal{F}_{\mathrm{T}}(\mathbf{x})$ cannot be perfectly equal to $\mathcal{F}_{\mathrm{T}}(\mathbf{x})$. So after obtaining the synthetic input $\mathbf{x}'$, we will examine whether $\arg\max_i \mathcal{T}(\mathbf{x})_i == \arg\max_j \mathcal{T}(\mathbf{x}')_j$ on the candidate. If not, we will discard $\mathbf{x}'$ and generate a new one.

3.3 Step II: Teacher Model Inference

The next step is to infer whether the student model comes from one of the candidate teacher feature extractors. For each candidate feature extractor, the attacker can generate a set of fingerprinting pairs and send them to the black-box student model. Intuitively, for a specific feature extractor $\mathcal{F}_{\mathrm{T}}$, the more matched pairs of black-box responses are obtained, the more likely the target model is transferred from $\mathcal{F}_{\mathrm{T}}$. Here we define the *matching set* useful for further discussion.

Definition 3.2 (Matching Set)**.** *After sending N fingerprinting pairs, all the pairs triggering two same API responses ($y_i == y'_i$) compose the matching set S_{match}.*

We use three heuristics to measure how much "belief" the attacker has to infer the teacher feature extractor.

Matching proportion. A high matching proportion of fingerprinting pairs indicates a high possibility that the target student model uses the same feature extractor as the attacker. We compute the matching proportion as a primary metric

to depict how "perfectly matched" the candidate is with the target feature extractor.

$$P_{\text{match}} = \frac{|S_{\text{match}}|}{N}.$$

The attacker will obtain a fingerprinting vector composed of matching proportions $v_{\text{fgpt}} = [P^1_{\text{match}}, P^2_{\text{match}}, \cdots]$ for all candidate feature extractors $[\mathcal{F}_{\mathcal{T}_1}, \mathcal{F}_{\mathcal{T}_2}, \cdots]$. Then the attacker will choose the candidate feature extractor that achieves the highest P_{match} as their inference result. Moreover, there are cases where the actual teacher model does not belong to the candidate feature extractor set, or the target model is not trained by transfer learning. To handle these cases, our attack introduces a pre-defined threshold τ. If $\max(v_{\text{fgpt}}) < \tau$, the inference result will be set as `NULL`, which means that the target model is not transferred from the candidate feature extractors.

A high matching proportion is insufficient to prove that the candidate feature extractor is adopted by the student model. It is also possible to obtain matched pairs from unmatched features. That is, the latent features extracted from probing inputs and synthetic inputs are significantly different. We will dive into such "false matching" phenomenon in Section 5.

Eccentricity. Eccentricity is adopted by [30] to measure how an item "stands out" from the rest in a set. It is defined as

$$E(v) = \frac{\max(v) - \max_2(v)}{\sigma(v)},$$

where $\max(\cdot)$ and $\max_2(\cdot)$ refers to the first and second highest value, respectively, and $\sigma(\cdot)$ refers to the standard deviation. The higher the eccentricity of v_{fgpt}, the more distinguishing is $\max(v)$, and the more confident the attacker is to make the inference.

Empirical entropy. Here, we introduce the empirical entropy of a set as a heuristic to measure how much "information" the set presents. Formally, for a set $\{x_1, x_2, \cdots, x_n\}$ and a sample space $\mathcal{X}$, the empirical entropy is defined as:

$$H(\mathcal{X}) = -\sum_{x \in \mathcal{X}} \hat{p}(x) \log \hat{p}(x),$$

where

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(x_i == x),$$

and $\mathbb{1}(\cdot)$ is the indicator function. We calculate the empirical entropy of the **matching set** with the highest P_{match}, to estimate how much information is given to make the inference.

4 Evaluation

4.1 Dataset

We use the following datasets to train student models:

Dogs-vs-Cats. Our Dogs-vs-Cats dataset contains 12,500 dog and 12,500 cat images from the Internet [2]. We select the first 10,000 dog images and the first 10,000 cat images to compose the training set. The remaining 2,500 dog images and 2,500 cat images compose the testing set.

MNIST. The MNIST dataset [3] is a widely used dataset to build up toy image recognition ML models. It consists of 60,000 training and 10,000 testing handwritten digit samples, containing ten classes from digit 0 to digit 9.

CIFAR10 and CIFAR100. The CIFAR10 and CIFAR100 datasets [4] are another two widely adopted ML datasets. The CIFAR10 dataset contains ten classes, with 5,000 training samples and 1,000 test samples in each. The CIFAR100 dataset is similar to the CIFAR10, except that the former has 100 classes with 600 images in each, including 500 training and 100 testing images.

STL10. The STL10 dataset [5] is originally designed to develop supervised as well as unsupervised learning models. Our experiment uses the labeled set covering ten classes, with 500 training images and 800 test images per class.

CelebA. We use the first 50,000 facial images as training samples and the following 10,000 facial images as the test samples from the CelebA dataset [27], which are annotated with 40 binary attributes. For CelebA, we build up multi-label transfer learning models to tag each image with 40 attributes.

We also use different datasets to generate probing inputs:

VOC-Segmentation. Our experiments assume the attacker has obtained the training dataset for the segmentation task of the VOC2012 challenge [6], consisting of 1,464 samples. Besides, we further assume the image annotation information (i.e., segmentation and class information) is unavailable.

Random Noise. This dataset simulates the case that the attacker cannot acquire any realistic image to build up the attack dataset. In this scenario, they have to synthesize images with randomly generated pixels to compose the attack dataset. We assume the attacker samples each pixel by the normal distribution and normalizes it into an integer within $[0, 255]$.

For all images used in our experiments, they are firstly preprocessed into the 8-bit 224×224 RGB format.

4.2 Experiment Setup

Teacher models. We have downloaded pre-trained models AlexNet, DenseNet121, MobileNetV2, ResNet18, VGG16, VGG19, and GoogLeNet from the official repository of PyTorch. Furthermore, to examine whether our proposed attack can discriminate teacher feature extractors with the same architecture trained by different organizations, we have also downloaded an AlexNet model from the PyTorchCV package repository [7]. The detailed information of the pre-trained models is listed in Table 3. We adopt the whole convolution part of each pre-trained model as the feature extractor in our experiments. Particularly, for VGG16 and VGG19, we also adopt the fully connected layers, except the last output layer.

Student models. For the Dog-vs-Cats, MNIST, STL10, CIFAR10, and CIFAR100 dataset, we develop multi-class clas-

sification students models. As for the CelebA dataset, we develop multi-label models which annotate the input with 40 binary attributes simultaneously. We build up three student models for each pre-trained feature extractor individually, by appending different fully connected layers. The detailed transfer learning setup is described in Appendix E. Our basic transfer learning setup is the *feature extractor* approach. That is, the parameters of the pre-trained feature extractor are fixed. Furthermore, in Section 4.3.3, we will evaluate the attack performance when some components of the feature extractor get fine-tuned during the transfer learning process.

Other targets not trained with transfer learning. In our experiments, we also consider target models trained from scratch, which have the same model architectures with AlexNet and ResNet18 student models.

Basic setup. Here we introduce the basic attack setup, followed by most of our experiments unless otherwise specified. We assume the attacker has owned seven pre-trained feature extractors from public resources (models listed by Table 3 except for GoogLeNet), and they have crafted 100 fingerprinting pairs for each teacher feature extractor. For the ease of comparison, we also assume that for each candidate teacher model, the attacker knows how many layers of the teacher model will be used (e.g., k in Figure 1 is known). We will investigate the case when k is unknown in Section 4.3.2. We randomly select the attacker's probing images from the VOC-segmentation dataset, which does not overlap with the training dataset of the teacher models or student models. In Section 4.3.5, we will investigate the attack effectiveness when no realistic probing image is available. Only the top-1 classification label will be reported by the black-box target, except that the multi-label classifier trained on the CelebA dataset will return 40 facial attributes. We assume the input format and input pre-processing module are transparent to the attacker. For most real-world applications, the input format is described by the API reference book. Otherwise, it is also feasible to infer the input format efficiently with some reverse engineering techniques [44].

Choice of τ. We assume our target is a c-class classifier. When the classifier simply outputs random results, the probability that the attacker receives a pair of matched responses is $\frac{1}{c}$. To avoid such random matching, we should at least ensure $\tau > \frac{1}{c}$. A generic way is to preset a positive number $\beta > 1$ and let $\tau = \frac{\beta}{c}$. Also, we must ensure $\beta \leq 2$ so as to when $c = 2$, τ will not exceed 1. When there are no auxiliary dataset and model to help determine τ, we directly set $\tau = \frac{1.5}{c}$.

However, when c becomes large, $\tau \approx \frac{1}{c}$. Another feasible way is to empirically find a τ. In our experiment, we assume the attacker owns another classification dataset Fashion-MNIST [43]. For each candidate model, the attacker first trains five student models on the Fashion-MNIST dataset. Then, for a specific τ, the attacker randomly chooses around 50% teacher candidates from the candidate set (3/7 in our case), and launches inference attacks against the student models. After repeating this process 20 times, the attacker calculates the true positive rate (TPR) and false positive rate (FPR), where we assume that the student coming from a teacher candidate is a positive event. By increasing τ from 0 to 1, the attacker can obtain the ROC (receiver operating characteristic) curve for analysis, as presented in Figure 2. $\tau = 0.3$ is a good choice, as it achieves a high TPR and a low FPR. Finally, we set τ as $\max(\frac{1.5}{c}, 0.3)$.

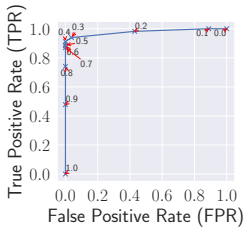


Figure 2: ROC curve with different τ. Each point refers to a τ.

4.3 Evaluation Results

4.3.1 Overview

We exhibit our basic experiment results on transfer learning targets in Figure 3. Each row of the subfigure represents a fingerprinting vector v_{fgpt} against a specific student model, in which each column reports the matching proportion P_{match} on one teacher feature extractor candidate. For all the 126 victim student models transferred from the seven teacher model candidates, the teacher model candidates with the highest P_{match} are consistent with the ground truths (i.e., 100% inference accuracy). Also, our attack against 13 out of 18 GoogLeNet student models and 31 out of the 36 trained-from-scratch models returns `NULL`. For the ease of analysis, in the remaining parts, we mainly consider the case that all the possible teacher candidates are obtained by the attacker, which is a common assumption by existing work [42, 45].

Our first observation is that, in general, the more classes that the transfer learning task involves, the stronger evidence the attacker can receive to infer the teacher model. We can see that for the CIFAR100 learning task, the highest P_{match} is significantly higher than other elements of v_{fgpt}, which conveys strong evidence that the student model is likely to be transferred from the corresponding teacher model. In the meanwhile, the evidence drawn from the Dogs-vs-Cats student models is not so obvious, as we can see high P_{match} on multiple teacher feature extractor candidates. For instance, when sending fingerprinting pairs generated from VGG19 to the three black boxes transferred from the VGG16, the attacker has achieved P_{match} as 0.66, 0.76, and 0.86, respectively. They are relatively high compared with these on the CIFAR100 targets (0.13, 0.04, and 0.08).

Our second observation is that the student dataset possibly affects the attack performance. For the four 10-class classification tasks, we can see that the discrimination between

elements of v_{fgpt} in an attack against MNIST student models is less evident than in the other three kinds of transfer learning student models. In fact, the attack performance is subject to the similarity between the student dataset and the probing dataset. We will study this phenomenon in Section 4.3.5.

4.3.2 For Unknown k

In practice, the number of components k used by the teacher model is usually available when the teacher model gets released [45]. But it is still worth investigating the rare case when k is unknown. In this case, we choose the following strategy to identify k: for all possible values of k, the attacker synthesizes attack queries. Then, the attacker launches attacks with different k, from the smallest value to the largest value. For each k, if the predicted result is not `NULL`, we will record the current candidate model and the corresponding k as the inference result. Otherwise, the inference process will stop.

Setup. Basically, in our experiment, we group pre-trained layers by separating them by the pooling layers, and we call each separated part a "block." In our experiment, we freeze or fine-tune blocks instead of individual layers. We report the number of blocks for each $\mathcal{F}_{\text{T}}$ in our experiment in Table 3. So, in our case, we choose to identify the number of pre-trained blocks instead of the number of pre-trained layers k. In our experiment, we consider teacher models removing the last pre-trained block and the last two pre-trained blocks, since in practice, deep features are wanted [45]. It is notable that we freeze all pre-trained components in this experiment.

Results. For the target models removing the last block, we achieve a 100% (126/126) inference accuracy. Meanwhile, for the target models removing the last two blocks, we achieve a 65.87% (83/126) inference accuracy.

One possible illustration of this phenomenon is that the earlier layers extract more generic visual features from the input, such as edges, dots, and textures. In contrast, the deep layers extract more abstract features (like complex patterns, objects, and even concepts), which is more specific knowledge of the teacher dataset. That means features from deeper layers carry the information on how the teacher model refines knowledge from inputs, and therefore they are the key to extracting teacher model fingerprints. We can think that most part of the teacher model fingerprint is removed when deep layers get discarded.

Figure 4 exhibits some synthetic inputs generated from different blocks. We can observe that synthetic inputs generated from lower-level blocks tend to present more concrete contents, i.e., the "plane" pattern presented in the probing input. Meanwhile, synthetic inputs from high-level blocks look more like abstract "noisy patterns." To understand this phenomenon, we need to review how features are extracted and propagated layer by layer. On early layers, local patterns of the synthetic input are captured, while on deep layers, a global and abstract description of the input is extracted. As a result, only if the synthetic inputs contain sufficient visual details and local patterns (edges, dots, etc.) can they activate similar low-level features with the probing inputs. Considering an extreme case: when we generate a synthetic input with the probing input on the input layer, we tend to produce a copy of the probing input. Consequently, they are more likely to result in matched responses on different models, because they are close to the probing inputs on too many low-level features. It leads to more unwanted matches on victims transferred from other teacher candidates. Hence, it is critical to extract fingerprints from deep layers.

4.3.3 Impact of Fine-Tuning

Fine-tuning is a commonly adopted strategy for better model performance or faster model convergence [14]. Here, we explore how fine-tuning techniques affect the attack.

Setup. Recall that we divide pre-trained feature extractors into blocks. We study the following cases: frozen feature extractor, and the last one to five blocks get fine-tuned. In general, when more blocks get fine-tuned, more disturbances are introduced to the pre-trained model.

Results. We evaluate the average inference accuracy for each case and plot them in Figure 5a. In general, the inference accuracy is likely to decrease when more parameters get fine-tuned. We present fingerprinting vectors when one and two blocks get fine-tuned in Figure 15 and Figure 16. We also statistically study the eccentricity of v_{fgpt} as exhibited in Figure 5b. Our key observation is that the eccentricity of fingerprinting vectors significantly decreases when we fine-tune pre-trained layers of ResNet18, MobileNetV2, and DenseNet121, while fine-tuning high-level pre-trained layers of other targets have less impact on the attack performance. After checking the pre-trained model architecture, we find another potential cause of this phenomenon: ResNet18, MobileNetV2, and DenseNet121 contain batch normalization layers between convolution layers. When fine-tuned on new datasets, the empirical mean and variance of the batch normalization layer are updated gradually. It may cause a significant change in the latent feature representation, breaking the basic assumption that the $\mathcal{F}'_{\text{T}}(\cdot) \approx \mathcal{F}_{\text{T}}(\cdot)$.

4.3.4 Impact of Query Budget

Then query budget is a primary concern of a query-based attack. A low query budget will not only limit the attack cost, but also help the attacker to keep stealthy.

Setup. In our experiment, we examine the effect of the query budget by simulating attacks that send 1, 2, 5, 10, 20, 50, and 100 fingerprinting pairs for each teacher candidate.

Results. Figure 6 depicts attack performance on different query budgets, i.e., the number of fingerprinting pairs for each teacher model candidate sent to the target black box. Therefore, the total number of queries is $2 * \text{query budget} * |\{\mathcal{F}_{\mathcal{T}_j}\}|$.

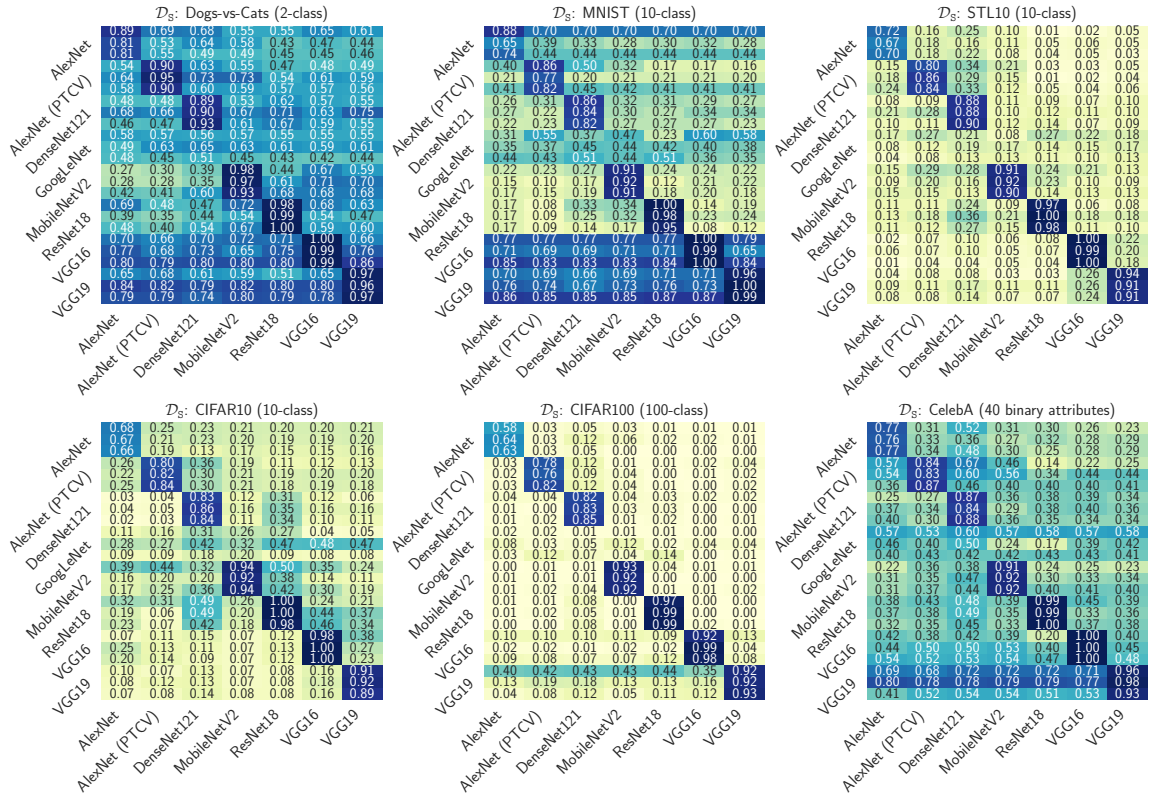


Figure 3: Teacher model fingerprinting vectors w.r.t. different classification tasks (100 fingerprinting pairs for each teacher model candidate, in the feature extractor setting). The x-axis represents the candidate teacher model, while the y-axis represents the actual teacher model. Each row refers to a v_{fgpt}, and every three adjacent rows correspond to three different student models from the same teacher model, e.g., the first three rows of each subfigure represent three different models transferred from AlexNet. Note: "GoogLeNet" in the y-axis shows the case that the actual teacher model does not belong to the candidate set (i.e., models annotated by the x-axis). In this case, our attack against 13 of 18 GoogLeNet student models returns `NULL` (i.e., "no answer").

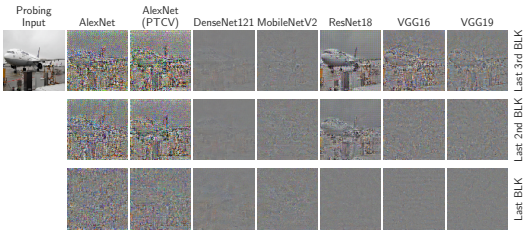
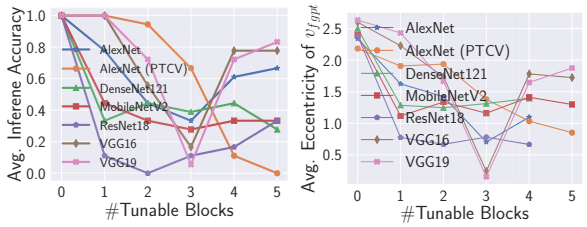


Figure 4: An example of how probing features affect synthetic inputs. The picture at the left top corner is the original probing input. Each row refers to synthetic inputs generated by features from different blocks, and each column refers to synthetic inputs for a specific candidate teacher model.

Overall, as the query budget grows, the inference accuracy increases rapidly. When the query budget reaches 50, the total inference accuracy achieves 100%. It suggests that our attack can avoid possibly high query charge caused by existing model reverse engineering attacks [32, 40]. We also see that the attack on more complex tasks (more classes and more complicated inputs) tends to achieve higher inference accuracy. For example, inference accuracy on CIFAR100 classifiers exceeds 80% even the query budget is limited to 10.

To gain a deeper insight, we also examine the average eccentricity of v_{fgpt} and the average entropy of S_{match} with the highest P_{match}. We find that there possibly exist some correlations among eccentricity, entropy heuristics, and inference accuracy. Firstly, the attack against a more complex transfer learning task is likely to bring higher "inference belief." For instance, the attack on CIFAR100 classifiers shows relatively high average eccentricity and entropy compared to other learning tasks. Secondly, we can observe that as the query budget increases, the average eccentricity of v_{fgpt} and the average entropy of S_{support} also rise. This phenomenon is consistent with our intuition: more queries are supposed to return more "evidence" to the attacker (higher entropy), and thus, the attacker has more confidence (higher eccentricity) to perform the inference, and they have a higher chance to hit the correct answer (higher accuracy).



(a) (b)

Figure 5: Performance on fine-tuned feature extractor layers: (a) average inference accuracy and (b) average eccentricity of v_{fgpt} w.r.t. seven different $\mathcal{F}_{\text{T}}$. Each line corresponds to one teacher model. When fine-turning the last four or the last five blocks, we decrease the learning rate from 10^{-3} to 10^{-5}, so as to achieve a stable learning process. Such learning rate change leads to some turning points for x=3 in the plot. For x=4 and x=5 in the plot, though the number of fine-tuned parameters grows up, a lower learning rate reduces the absolute change of these parameters, which might lead to a higher inference accuracy.

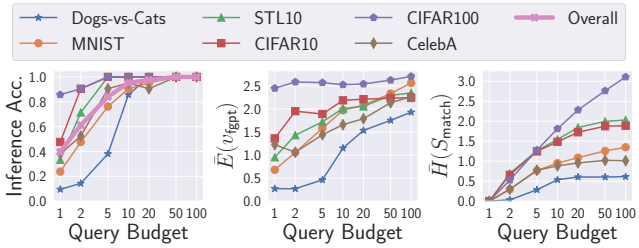


Figure 6: Performance on different query budgets. Each line depicts the attack performance on a specific transfer learning task. We have achieved 100% inference accuracy when the query budget reaches 50 (pairs for each teacher model candidate). On transfer learning tasks with more complex input patterns and more classes, we get higher eccentricity and entropy heuristics.

4.3.5 Impact of Probing Inputs

One of our basic attack assumptions is that any information about the population of $\mathcal{D}_{\text{S}}$ is unavailable. Previously, we use probing inputs from a public dataset, VOC-Segmentation, which is not overlapped with $\mathcal{D}_{\text{S}}$. But what if when there exists a "stronger" attacker, who has collected probing inputs following the same distribution with $\mathcal{D}_{\text{S}}$; or there is a "weaker" attacker, who has no access to any realistic data?

Setup. To this end, we investigate how probing datasets affect attacks with extra evaluations on the other three probing datasets:

- **MNIST and CelebA.** They are realistic images and share the same distribution with $\mathcal{D}_{\text{S}}$. We randomly choose 100 probing inputs from the MNIST and CelebA testing dataset, respectively. In this case, the $\mathcal{D}_{\text{S}}$ and the input probing dataset are still not overlapped.
- **Random Noise.** This is an extreme case where no realistic images are available. We randomly sample 100 probing inputs from the normal distribution.

Results. Overall, using the VOC-Segmentation dataset, in general, achieves good performance, while Random Noise leads to the worst performance. We plot the attack performance with MNIST, CelebA, and Random Noise probing data in Figure 11, Figure 12, and Figure 13 respectively.

The major obstacle to using Random Noise data as probing inputs is the low entropy of S_{match}. We find that the target model often classifies Random Noise inputs to a specific label, leading to a low entropy value of S_{match}. Meanwhile, when we choose MNIST as the probing dataset, we have achieved high entropy values and 100% inference accuracy (query budget is 100) on the MNIST-classification targets. We have similar observations when using CelebA as probing inputs for CelebA-classification models. This is expected: with sufficient probing inputs from the same distribution with $\mathcal{D}_{\text{S}}$, the attacker can increase the variety of output labels and increase the entropy of S_{match}. Consequently, they have more information to make a reliable inference. We will further analyze how the probing inputs affect the attack in Section 5.

5 More Robust Teacher Model Fingerprinting

The experimental results exhibited in Section 4 have shown that the naïve "one-of-the-best" strategy is effective in most cases. However, we can also find that it is possible to obtain a high P_{match} by a mismatched feature extractor. For instance, in Figure 3, for MNIST classification student models, we get $P_{\text{match}} = 0.86$ with an AlexNet candidate against a VGG19-based student model. It indicates there would exist a considerable number of "false matches" in the matching set S_{match}. One possible reason for the false-matching problem is that most attack queries (especially the synthetic input) belong to unrecognizable contents of the target black box. In this scenario, the target classifies a pair of "meaningless" inputs into the same class occasionally, even if they produce very different features. We can understand this phenomenon in an extreme case: supposing a black box only returns the label "1" to any input. In this case, all our sent fingerprint pairs will activate seemingly perfect matched responses, and we will surely get $P_{\text{match}} = 1$ for each candidate teacher feature extractor. But it is obvious that we cannot find out the actual teacher model as we are unable to affect the black-box's behavior.

We design inference strategies leveraging statistical testing methods to alleviate the impact of false-matching problems. Here we firstly give one definition for further discussion.

Definition 5.1 (Supporting Set)**.** *After removing the elements with the maximum occurrence from S_{match}, the remaining elements compose the supporting set $S_{\text{support}} = \{y | \forall y \in$*

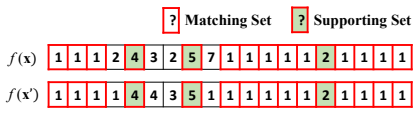


Figure 7: Illustration of the supporting set. The supporting set is the collection of the remaining elements after removing the elements with the maximum occurrence from S_{match}.

$S_{\text{match}}, y \neq \arg\max_i |\{\forall y_i \in S_{\text{match}}\}|\}$, *which will be leveraged as the evidence to infer the original model.*

The relationship between the fingerprinting attack responses, matching set, and supporting set is explained by Figure 7.

Then we consider the extreme case that the student model $\mathcal{S}$ has zero knowledge about the input. It randomly divides the input space into c classes, which we call a *random classifier*.[4] So how much is the possibility of getting a match on a fingerprinting pair? We formalize the following statistical hypothesis testing: given a candidate teacher model $\mathcal{T}$, a student model $\mathcal{S}$, a set of fingerprinting pairs $\{\langle \mathbf{x}_i, \mathbf{x}_i' \rangle\}$ and the corresponding responses $\{\langle y_i, y_i' \rangle\}$, we make the following hypothesis

$$H_0 : \mathcal{S} \textit{ is a random classifier.}$$
$$H_1 : \mathcal{S} \textit{ is not a random classifier.}$$

We can give a sufficient condition to reject the null hypothesis H_0 give a specific significance value α.

Theorem 1. *Given a c-class classification student model $\mathcal{S}$ in top-1 label exposure, a fingerprinting statistical hypothesis (H_0, H_1, as well as the significance value α), the size of the supporting set sufficient to reject the null hypothesis H_0 is*

$$\left\lceil \log_2 \frac{1}{\alpha} \right\rceil. \tag{5}$$

Proof. See Appendix A.

Only when H_0 is rejected can we make a more robust inference. That is, the attacker has high confidence (1-α) to avoid a wrong inference against a random classifier.

Here we provide a more robust version of the teacher model fingerprinting attack: (1) generating fingerprinting pairs → (2) selecting the best candidate → (3) if H_0 is rejected, accepting the inference result; otherwise, sending more queries and repeating, until reaching the maximum query budget.

Evalution. We repeat several empirical studies on our robust attack method. We set the significance value α as 0.01, and thus, the testing hypothesis threshold $\lceil \log_2 \frac{1}{\alpha} \rceil = 7$. Table 1 exhibits the robust version of inference results for Section 4.3.5. Overall, we have achieved 100% inference accuracy on the robust inferences that reject H_0 with $\alpha = 0.01$, even without realistic datasets (Random Noise as the probing dataset). We also examine the size of the supporting set on our inference results in Section 4.3.5 to investigate how probing inputs and transfer learning tasks affect the attack performance.

Impact of probing inputs. The probing input distribution indeed impacts the attack performance. From Figure 8, we can see that, with realistic probing inputs, the proportions of robust inferences are obviously higher than that with Random Noise. This is consistent with our observation in Section 4.3.5. Moreover, a dataset that shares a similar distribution with $\mathcal{D}_S$ is more helpful in reducing the query budget, because it can increase the variance of the black-box responses. For example, for MNIST classifiers, the supporting set size $|S_{\text{support}}|$ significantly rises when we use MNIST as the probing dataset.

Impact of transfer learning tasks. Figure 8 shows that more complex learning tasks (i.e., those have more complex inputs and more classes) tend to produce a larger $|S_{\text{support}}|$. For instance, when attacking CIFAR100 classifiers, most inferences successfully reject H_0 as their $|S_{\text{support}}|$ are over the threshold. One reason is that a complex learning task is likely to give a higher variance of the responses. This result is consistent with our observation in Section 4.3.4. To help readers understand this phenomenon, we can estimate the minimal size of S_{support}. Since Equation 5 points out that the minimal $|S_{\text{support}}|$ should be $\lceil \log_2 \frac{1}{\alpha} \rceil$, for a c-class classifier, $|S_{\text{match}} \setminus S_{\text{support}}|$ should be no less than $\left\lceil \frac{\lceil \log_2 \frac{1}{\alpha} \rceil}{c-1} \right\rceil$. Therefore, we have

$$|S_{\text{match}}| \geq \left\lceil \log_2 \frac{1}{\alpha} \right\rceil + \left\lceil \frac{\lceil \log_2 \frac{1}{\alpha} \rceil}{c-1} \right\rceil. \tag{6}$$

So, according to Equation 6, we can find that when c increases, we will get a lower bound of the size of the supporting set. It should be pointed out that this is just a sufficient condition to reject H_0 for the robust version of our attack.

6 Applications of the Fingerprinting Attack

In this section, we demonstrate how our attack helps to mount model stealing and identify vulnerable teacher models.

6.1 Practical Model Stealing

Setup. The basic model stealing strategy is to firstly identify the teacher feature extractor of the victim, and then build up a surrogate student model based on it. Next, the attacker sends queries to the black-box target and collects the top-1 label responses. Finally, the attacker trains the surrogate student model with the attack queries and corresponding responses. In our experiment, we choose four CIFAR-10 classifier victims, which are transferred from AlexNet, ResNet18, VGG16, and VGG19, respectively. The victim models own the same fully connected layer architecture: `FC`(128) → `BN`→ `SF`(10). We

[4] We emphasize that the word "random" here refers to that the parameters of $\mathcal{S}$ are randomly configured with respect to $\mathcal{D}_S$, rather than it outputs label randomly given a specific input.

Query Budget	probing: VOCSegmentation			probing: MNIST			probing: CelebA			probing: Random Noise		
	inference acc.		#robust / #original	inference acc.		#robust / #original	inference acc.		#robust / #original	inference acc.		#robust / #original
	original	robust		original	robust		original	robust		original	robust	
1	39.68% (50/126)	– (0/0)	0 (0/126)	42.06% (53/126)	– (0/0)	0 (0/126)	45.24% (57/126)	– (0/0)	0 (0/126)	19.84% (25/126)	– (0/0)	– (0/126)
2	61.11% (77/126)	– (0/0)	0 (0/126)	57.94% (73/126)	– (0/0)	0 (0/126)	57.94% (73/126)	– (0/0)	0 (0/126)	29.37% (37/126)	– (0/0)	– (0/126)
5	84.13% (106/126)	– (0/0)	0 (0/126)	69.84% (88/126)	– (0/0)	0 (0/126)	80.95% (102/126)	– (0/0)	0 (0/126)	42.06% (53/126)	– (0/0)	– (0/126)
10	95.24% (120/126)	100.00% (32/32)	25.40% (32/126)	80.95% (102/126)	100.00% (19/19)	15.08% (19/126)	89.68% (113/126)	100.00% (3/3)	2.38% (3/126)	50.79% (64/126)	– (0/0)	– (0/126)
20	97.62% (123/126)	100.00% (97/97)	76.98% (97/126)	(84.92% (107/126)	100.00% (52/52)	41.27% (52/126)	96.83% (122/126)	100.00% (87/87)	69.05% (87/126)	57.14% (72/126)	100.00% (16/16)	12.70% (16/126)
50	100.00% (126/126)	100.00% (125/125)	99.21% (125/126)	90.48% (114/126)	100.00% (96/96)	76.19% (96/126)	99.21% (125/126)	100.00% (117/117)	92.86% (117/126)	62.70% (79/126)	100.00% (36/36)	28.57% (36/126)
100	100.00% (126/126)	100.00% (126/126)	100.00% (126/126)	96.03% (121/126)	100.00% (114/114)	90.48% (114/126)	100.00% (126/126)	100.00% (122/122)	96.83% (122/126)	65.08% (82/126)	100.00% (41/41)	32.54% (41/126)

Table 1: Comparison between the original and robust version of the teacher model fingerprinting attack. We have achieved 100% inference accuracy on the remaining inferences after the hypothesis testing ($\alpha = 0.01$, $\lceil \log_2 \frac{1}{\alpha} \rceil = 7$).

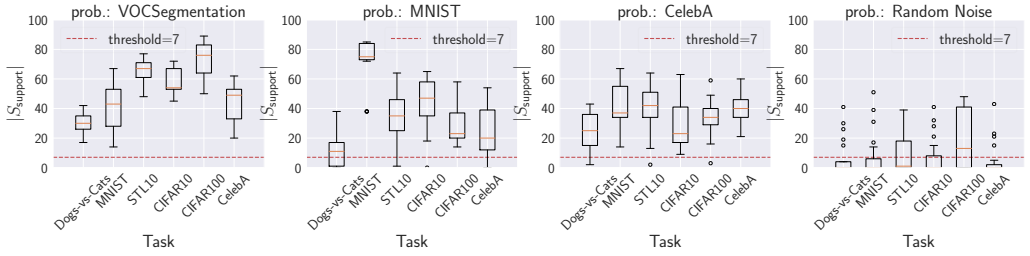


Figure 8: Supporting set size w.r.t. different learning tasks and different probing inputs (query budget=100, $\alpha = 0.01$, the hypothesis testing threshold $\lceil \log_2 \frac{1}{\alpha} \rceil = 7$). If the probing inputs follow a similar distribution with $\mathcal{D}_s$, $|S_{support}|$ is relatively high.

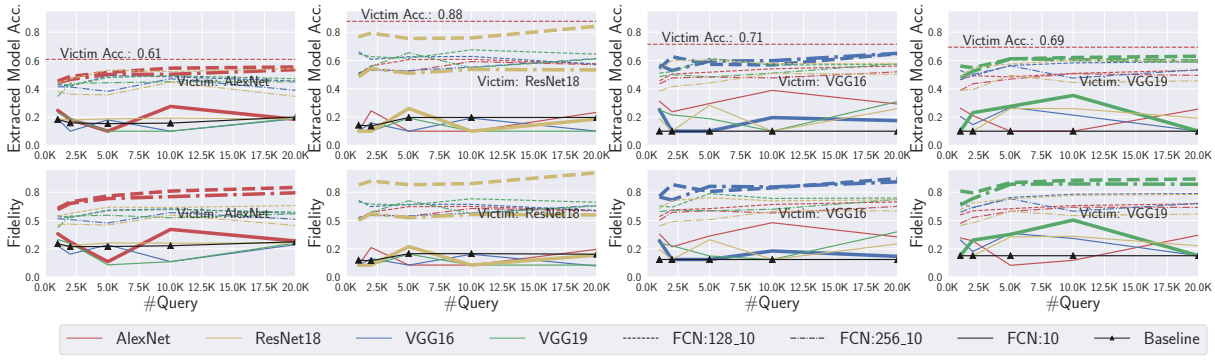


Figure 9: Model stealing against transfer learning models. The top row reports the accuracy metric, while the bottom row reports the fidelity metric of extracted models, respectively. For an extracted model transferred from the same teacher model with the target (i.e., inferred by a successful teacher model fingerprinting attack; marked with thicker lines), it has finally achieved higher accuracy and fidelity with sufficient queries (#Query $\geq$ 10,000).

choose training images from CIFAR100 as the attack queries (non-overlapping with the target's $\mathcal{D}_s$, but sharing a similar distribution). We evaluate the model stealing performance with four pre-trained models: AlexNet, ResNet18, VGG16, and VGG19. To examine the affect by surrogate model architectures, for each pre-trained model we build three surrogate model with three different fully connected layer architectures: `SF`(10), `FC`(128) $\rightarrow$ `BN`$\rightarrow$ `SF`(10), and `FC`(256) $\rightarrow$ `BN`$\rightarrow$ `SF`(10). Additionally, for each victim model, we train a surrogate model with the same model architecture but from scratch as the baseline. Each surrogate student model is trained for 20 epochs with batch size as 48.

Evaluation metric. We choose two evaluation metrics widely adopted by prior studies on model stealing attacks [19, 40]:

- **Accuracy**: accuracy of the extracted model on inputs. It measures how well the extracted model makes correct predictions on incoming inputs. This metric usually lies in the context where an attacker aims to steal the strong prediction "power" of the victim black-box ML service with only limited efforts.
- **Fidelity**: agreement between the target and the extracted model on inputs [40]. It measures the behavior similarity between the victim model and the extracted model. This metric usually lies in the context where an attacker aims to faithfully replicate the "functionality" of the black-box victim, in particular, these potential vulnerabilities.

We evaluate the extracted models on CIFAR10 testing set non-overlapping with the victim training set or the query set.

Results. Figure 9 shows the model stealing attack performance with different pre-trained models. We highlight model stealing results after successful teacher model fingerprinting attacks, i.e., the extracted model and the victim share the same teacher model, with thicker lines. We can observe that for model stealing after correctly identifying the teacher model, the fidelity and accuracy are obviously higher than the baseline. Moreover, we can find that even if the surrogate model has a different architecture of the fully connected layers, we may also achieve a relatively good model stealing.

6.2 Identifying Vulnerable Teacher Models

As vulnerable teacher models pose severe threats to downstream applications [45, 48], there is a surging need for solutions to detect the existence of vulnerable teacher models. We investigate the feasibility of using our attack to detect if a student model comes from a known vulnerable teacher model.

Setup. We adopt the method proposed by Yao et al. [45] to construct teacher models with latent backdoors. We choose MNIST, CIFAR10, and CIFAR100 as the dataset in our experiment. Then, we build the backdoored teacher model for each dataset from pre-trained AlexNet, AlexNet (PTCV), and ResNet18, respectively. Following a similar setup in [45], we first use data from class 0-4 to train the original teacher model, and use data from class 5-9 to train the student model. We choose class 5 as the target class in all our backdoor attacks. For each backdoored teacher model, we train three student models with different fully connected layer architectures. As a result, there are nine backdoored student models for each dataset, nine clean student models, and six candidate models owned by the attacker. We provide more details of the vulnerable teacher models in Appendix C.

Results. The inference accuracy of backdoored and clean teacher models is 19/27 and 15/27, respectively. In the given case, the most challenging task is to discriminate the clean teacher model from the backdoored teacher model, as the backdoored teacher models are fine-tuned from the clean teacher models. Nevertheless, our results still show it is possible to use our attack to identify vulnerable teacher models.

7 Discussions

7.1 Possible Countermeasures

Input distortion. One potential solution is to distort inputs by inserting small random noise or performing image transformations like image cropping and resizing. Hopefully, it would only slightly affect model performance on realistic inputs but significantly reduce the inference attack accuracy, since the latent feature is sensitive to the optimized "noise" pattern in a synthetic input. One primary advantage of this strategy is that the input distortion processor can be directly plugged between the raw system input and the model input, without modifying the student models. Also, the input distortion is lightweight and easy to implement, as most image processing or deep learning libraries support various transformation operations.

Injecting neuron distances [42]. Wang et al. propose a defense method, called *injecting neuron distances*, to deviate the student model's feature map from the teacher model [42]. The key is to retrain all student layers on the student dataset, with the optimization goal to minimize the cross-entropy loss while ensuring the dissimilarity between the student's and the teacher's intermediate representations is above a threshold. In this case, the intuition behind our attack, i.e., that the teacher model and student model share a similar feature map, does not hold. Despite the advantage of effectively disturbing the student model, *injecting neuron distances* brings additional costs to update the whole student model.

Evaluation. We investigate the feasibility of the two countermeasures on the CIFAR10 dataset. For *input distortion*, we randomly crop the input with 0.85 areas preserved, and then resize it to the size of the model's input layer. For *injecting neuron distances*, we slightly modify the implementation by [42]: instead of requiring a threshold, we directly introduce a negative neuron distance term into our learning objective: min $\texttt{CrossEntropy} - \lambda \cdot \texttt{NeuronDistance}$. In our experiment, we set $\lambda = 10^{-6}$.

To evade the defense, one attack strategy is to generate attack queries from shallower layers, so as to reduce the impact of parameter changes. We investigate the robustness of countermeasures against attack queries generated from different positions. We report the results in Table 2. We can observe that the two countermeasures can effectively defend our attack. Another possible adaptive attack is to perform the optimization Equation 1 simultaneously across different pre-trained layers. However, it would intensively increase the overhead. We will explore adaptive attacks in the future.

7.2 Limitations and Future Work

Language model. In this paper, our empirical studies focus on computer vision tasks. Recently, the transfer learning technique plays a vital role in a more broad range of fields. For instance, famous pre-trained language transformers, like

Defense	Attack Accuracy (Queries generated from)		
	Last BLK	Last 2nd BLK	Last 3rd BLK
Input Perturbation	13/21	11/21	10/21
Injecting Neuron Distances	7/21	5/21	5/21
Without Defense	21/21	20/21	15/21

Table 2: Performance of the two countermeasures. The average testing accuracy of models that are unprotected, protected by *input perturbation*, and protected by *injecting neuron distances* are 77.5%, 72.3%, and 79.1%, respectively. Note that since the *injecting neuron distances* trains all the model parameters, which may work like fine-tuning.

BERT and GPT-3, are widely adopted into various downstream applications, such as search ranking [17] and medical text mining [23]. We think it is possible to infer pre-trained language teacher models with the same strategy proposed in this paper. We plan to extend this work into sequential language models, and devise new fingerprinting attacks.

Advanced adversarial attacks. Our results show that we can lower the attack bound of model stealing attacks based on a successful teacher model fingerprinting attack. We believe our proposed attack can assist other advanced adversarial attacks, e.g., black-box adversarial example attack [42] and membership inference attack [36]. We will explore more adversarial attacks preceded by our attack in real-world settings.

Fingerprint erase. Our empirical study indicates that the teacher model fingerprint may still exist even after fine-tuning. In the feature, we plan to study which factors may impact the model fingerprint, and explore ways to erase teacher model fingerprints from student models as fundamental defenses.

8 Related Work

Model reverse engineering. Model reverse engineering aims to infer model parameters, structures, or other model-related information such as hyper-parameters, by just inspecting model responses to a specific set of attack queries.

One line is to reproduce mimic models in the parameter level, i.e., they try to recover the exact model parameters. Besides, Carlini et al. [9] propose a cryptanalytic extraction approach by sending queries at critical points. Another line tries to duplicate target models at the functionality level. Recently, Orekondy et al. [33] propose more advanced attack methods to steal model parameters. Jagielsk et al. [19] discuss the inherent limitations of the learning-based model stealing strategy, and develop a more practical functionality-equivalent stealing attack. For transfer learning, Wang et al. [42] propose a teacher model fingerprinting strategy. Its main idea is to craft a fingerprinting image to produce a nearly all-zero latent vector on the victim, when it is derived from the candidate teacher model. Yet, it requires access to the raw confidence values. In contrast, our attack works well when only top-1 labels are available. Figure 10 compares our attack with the attack proposed by Wang et al. [42]. It can be seen that our attack can achieve competitive inference accuracy when the query budget exceeds 10. We also propose a robust fingerprinting attack to overcome the false-matching problem. Besides, there are also efforts in inferring structures [32], hyper-parameters [41], or other model properties [16].

Figure 10: Comparison with the attack proposed by Wang et al. [42]. To simulate the setup in [42]: we assume the classification scores are available; each student model is composed of a fixed feature extractor and a classification layer. There are 42 student models in total (7 teacher models * 6 datasets).

The above studies reveal that there is no such an exact line separating the white-box and black-box models. Nonetheless, they need numerous queries and complex computations to change a black-box model into a white-box one, especially for complicated models like VGG16. When it comes to transfer learning, we believe our method has the advantage to quickly and efficiently reverse engineer most parts of the black-box victim—the complex transferred components, which is hard to achieve by directly combining the prior arts.

Adversarial attacks in the transfer learning setup. Recent studies reveal that the transfer learning technique may expose ML models to various threats. Zhang et al. [48] show that adversarial examples become more transferable on a model trained by fine-tuning. Wang et al. [42] propose an adversarial example attack by crafting malicious perturbations to activate mimic latent representations to be classified to the target class. Besides, Yao et al. [45] present a latent backdoor attack by releasing a malicious teacher model, with which the carefully mined backdoor will infect the student model. In addition to the security side, transfer learning also suffers from privacy threats. For example, Zou et al. [51] reveal that the public knowledge would raise the membership leakage risk in the transfer learning practice. There also exist a broad range of other adversarial attacks against ML [12, 13, 15, 18, 31, 35, 36, 37, 38, 46, 49], which are out of the scope of this paper.

9 Conclusions

In this paper, we take the first step to investigate the teacher model exposure threat in the transfer learning context. In particular, we propose a novel teacher model fingerprinting attack that can efficiently trace back the origin of a student model. Extensive experiment results demonstrate that our attack can accurately identify the teacher model even in restrictive attack

scenarios, such as top-1 label exposure and no realistic data available. Besides, we propose a robust attack to eliminate false positive inference results. We also show that our attack can facilitate advanced attacks or help model forensics. Our findings highlight the urgent need for new model confidentiality protection measures specified for transfer learning.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2020AAA0107702), National Natural Science Foundation of China (U21B2018, 62161160337, 62132011), Shaanxi Province Key Industry Innovation Program (2021ZDLGY01-02), the Research Grants Council of Hong Kong under Grants N_CityU139/21, R6021-20F, R1012-21, and the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4). Chao Shen, Cong Wang, and Yang Zhang are the corresponding authors.

References

[1] https://modelzoo.co.

[2] https://www.kaggle.com/c/dogs-vs-cats.

[3] http://yann.lecun.com/exdb/mnist/.

[4] https://www.cs.toronto.edu/~kriz/cifar.html.

[5] https://cs.stanford.edu/%7Eacoates/stl10/.

[6] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html.

[7] https://github.com/osmr/imgclsmob/.

[8] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring. In *USENIX Security Symposium (USENIX Security)*, pages 1615–1631. USENIX, 2018.

[9] Nicholas Carlini, Matthew Jagielski, and Ilya Mironov. Cryptanalytic Extraction of Neural Network Models. In *Annual International Cryptology Conference (CRYPTO)*, pages 189–218. Springer, 2020.

[10] Nicholas Carlini and David Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *CoRR abs/1705.07263*, 2017.

[11] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 39–57. IEEE, 2017.

[12] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 343–362. ACM, 2020.

[13] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. BadNL: Backdoor Attacks Against NLP Models with Semantic-preserving Improvements. In *Annual Computer Security Applications Conference (ACSAC)*, pages 554–569. ACSAC, 2021.

[14] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 153–160. JMLR, 2009.

[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1322–1333. ACM, 2015.

[16] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 619–633. ACM, 2018.

[17] Luyu Gao, Zhuyun Dai, and Jamie Callan. Understanding BERT Rankers Under Distillation. In *ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR)*, pages 149–152. ACM, 2020.

[18] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing Links from Graph Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 2669–2686. USENIX, 2021.

[19] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pages 1345–1362. USENIX, 2020.

[20] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 259–274. ACM, 2019.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[22] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General Visual Representation Learning. In *European Conference on Computer Vision (ECCV)*, pages 491–507. Springer, 2020.

[23] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 2020.

[24] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium (USENIX Security)*, pages 1605–1622. USENIX, 2020.

[25] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to Prove Your Model Belongs to You: A Blind-Watermark based Framework to Protect Intellectual Property of DNN. In *Annual Computer Security Applications Conference (ACSAC)*, pages 126–137. ACM, 2019.

[26] Zheng Li and Yang Zhang. Membership Leakage in Label-Only Exposures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 880–895. ACM, 2021.

[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Attributes in the Wild. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738. IEEE, 2015.

[28] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning Transferable Features with Deep Adaptation Networks. In *International Conference on Machine Learning (ICML)*, pages 97–105. JMLR, 2015.

[29] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial Frontier Stitching for Remote Neural Network Watermarking. *CoRR abs/1711.01894*, 2017.

[30] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing Social Networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 173–187. IEEE, 2009.

[31] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1021–1035. IEEE, 2019.

[32] Seong Joon Oh, Max Augustin, Bernt Schiele, and Mario Fritz. Towards Reverse-Engineering Black-Box Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[33] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4954–4963. IEEE, 2019.

[34] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237. ACL, 2018.

[35] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In *USENIX Security Symposium (USENIX Security)*, pages 1291–1308. USENIX, 2020.

[36] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

[37] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium*

on Security and Privacy (S&P), pages 3–18. IEEE, 2017.

[38] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine Learning Models that Remember Too Much. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 587–601. ACM, 2017.

[39] Qianru Sun, Liqian Ma, Seong Joon Oh, Luc Van Gool, Bernt Schiele, and Mario Fritz. Natural and Effective Obfuscation by Head Inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5050–5059. IEEE, 2018.

[40] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security Symposium (USENIX Security)*, pages 601–618. USENIX, 2016.

[41] Binghui Wang and Neil Zhenqiang Gong. Stealing Hyperparameters in Machine Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 36–52. IEEE, 2018.

[42] Bolun Wang, Yuanshun Yao, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. With Great Training Comes Great Vulnerability: Practical Attacks against Transfer Learning. In *USENIX Security Symposium (USENIX Security)*, pages 1281–1297. USENIX, 2018.

[43] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR abs/1708.07747*, 2017.

[44] Qixue Xiao, Yufei Chen, Chao Shen, Yu Chen, and Kang Li. Seeing is Not Believing: Camouflage Attacks on Image Scaling Algorithms. In *USENIX Security Symposium (USENIX Security)*, pages 443–460. USENIX, 2019.

[45] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y. Zhao. Latent Backdoor Attacks on Deep Neural Networks. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 2041–2055. ACM, 2019.

[46] 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 *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2020.

[47] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph. Stoecklin, Heqing Huang, and Ian Molloy. Protecting Intellectual Property of Deep Neural Networks with Watermarking. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pages 159–172. ACM, 2018.

[48] Yinghua Zhang, Yangqiu Song, Jian Liang, Kun Bai, and Qiang Yang. Two Sides of the Same Coin: White-box and Black-box Attacks for Transfer Learning. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2989–2997. ACM, 2020.

[49] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor Attacks to Graph Neural Networks. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 15–26. ACM, 2021.

[50] Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. Transfer Learning for Low-Resource Neural Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1575. ACL, 2016.

[51] Yang Zou, Zhikun Zhang, Michael Backes, and Yang Zhang. Privacy Analysis of Deep Learning in the Wild: Membership Inference Attacks against Transfer Learning. *CoRR abs/2009.04872*, 2020.

Appendix

A Proof of Theorem 1

Proof. For a c-class classifier $\mathcal{S}$ whose response is only the top-1 label (i.e. $y = \arg\max_i \mathcal{S}(\mathbf{x})_i$), define

$$p_k = \int_{\mathbf{x}\in\mathcal{X}} \Pr\left(\arg\max_i \mathcal{S}(\mathbf{x})_i == k\right), \forall k \in [1,2,\cdots,c], \quad (7)$$

where $\mathcal{X}$ refers to the input space.

Without loss of generality, we assume that $1 \geq p_1 \geq p_2 \geq \cdots \geq p_c \geq 0$.[5] Also noticing that $\sum_{i=1}^{c} p_i = 1$, we have

$$1 = \sum_{i=1}^{c} p_i \geq \sum_{i=1}^{k} p_i \geq k p_k \Rightarrow p_k \leq \frac{1}{k}. \quad (8)$$

Suppose the attacker generates N fingerprinting pairs $\{\langle \mathbf{x}_i, \mathbf{x}_i' \rangle\}$ with a set of probing input $\{\mathbf{x}_i\}$, and receives N pairs of responses $\{\langle y_i, y_i' \rangle\}$, among which M pairs satisfy $y == y'$. For simplicity, we further suppose the first M fingerprinting pairs produce the matched responses (i.e., $y_i == y_i'$ when $1 \leq i \leq M$, and $y_j \neq y_j'$ when $M+1 \leq j \leq N$). If the victim is a random classifier, we have

$$\begin{aligned}\Pr\left(\{y_i'\}|\{\langle \mathbf{x}_i, \mathbf{x}_i' \rangle\}, \{y_i\}, \mathcal{S}\right) &= \prod_{i=1}^{N} p_{y_i'} = \prod_{i=1}^{M} p_{y_i} \prod_{j=M+1}^{N} p_{y_j'} \\ &\leq \prod_{i=1}^{M} p_{y_i} \leq \prod_{j\in\{j|y_j\neq 1, j\leq M\}} p_{y_j}.\end{aligned} \quad (9)$$

Then let $K = |\{i|y_i \neq 1, i \leq M\}|$, we can obtain

$$\prod_{j\in\{j|y_j\neq 1, j\leq M\}} p_{y_j} \leq \prod_{j\in\{j|y_j\neq 1, j\leq M\}} p_2 = p_2^K \leq \left(\frac{1}{2}\right)^K. \quad (10)$$

Therefore, we get

$$\Pr\left(\{y_i'\}|\{\langle \mathbf{x}_i, \mathbf{x}_i' \rangle\}, \{y_i\}, \mathcal{S}\right) \leq \left(\frac{1}{2}\right)^K. \quad (11)$$

Finally, let $\left(\frac{1}{2}\right)^K \leq \alpha$, we can derive that

$$K \geq \log_2 \frac{1}{\alpha}. \quad (12)$$

That is, if $K \geq \lceil \log_2 \frac{1}{\alpha} \rceil$ (K is a positive integer), we can prove that $\Pr\left(\{y_i'\}|\{\langle \mathbf{x}_i, \mathbf{x}_i' \rangle\}, \{y_i\}, \mathcal{S}\right) \leq \alpha$, i.e., reject the null hypothesis H_0 to conclude that $\mathcal{S}$ is not a random classifier, and hence continue further fingerprinting inference.

According to our definition of supporting set (Definition 5.1), there exist two possible scenarios:

- Scenario 1: class 1 does not appear in the supporting set. By our result Equation 12, if the size of the supporting set $|S_{\text{support}}|$ is not smaller than $\lceil \log_2 \frac{1}{\alpha} \rceil$, the attacker will reject H_0 and continue further inference.
- Scenario 2: class 1 appears in the supporting set. Suppose class k is out of the supporting set, then we have:

$$\begin{aligned}K &= |S_{\text{support}}| - |\{i|y_i == 1, i \leq M\}| + |\{j|y_j == k, j \leq M\}| \\ &\geq |S_{\text{support}}| \geq \lceil \log_2 \frac{1}{\alpha} \rceil.\end{aligned} \quad (13)$$

The first inequality is based on that fact that $|\{j|y_j == k, j \leq M\}| \geq |\{i|y_i == 1, i \leq M\}|$.

[5] In most cases, $p_c > 0$. However, here we consider a more general case, where the black box's output space may not cover all the c classes.

Therefore, when the size of the supporting set satisfies $|S_{\text{support}}| \geq \lceil \log_2 \frac{1}{\alpha} \rceil$, we will reject the null hypothesis H_0 and continue further fingerprinting inference. ■

Remark. *Equation 12 gives an easy to calculate but pretty strict condition. In fact, we can simply induce a more precise one to reject H_0:*

$$\prod_{j \in \{j|y_j \neq 1, j \leq M\}} p_{y_j} \leq \prod_{i=2}^{c} \left(\frac{1}{i}\right)^{|\{j|y_j == i, j \leq M\}|} \leq \alpha. \quad (14)$$

In the supporting set, we firstly reassign the class label $\{y_i\}$ from the most frequent class to the least frequent class with 2 to c, and then check the condition Equation 14.

B Attack Performance with MNIST, CelebA and Random Noise as Probing Dataset

As a supplementary to Section 4.3.5, we plot the attack performance with MNIST, CelebA and Random Noise as the attack probing dataset in Figure 11, Figure 12 and Figure 13.

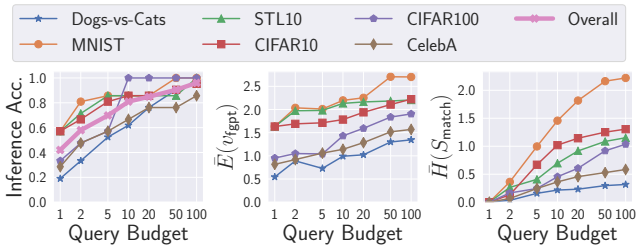


Figure 11: Performance with MNIST images as probing inputs. We have achieved a high average eccentricity of v_{fgpt} and a high average entropy of S_{match} on the MNIST classification task.

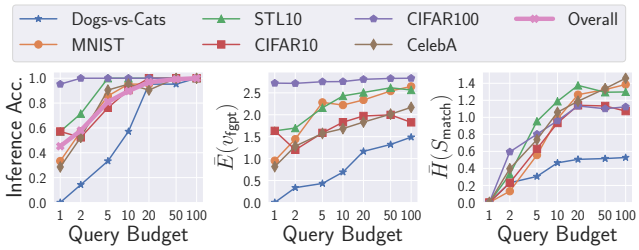


Figure 12: Performance with CelebA images as probing inputs. Compared to Figure 11, the three heuristics obviously increase on the CelebA classification task.

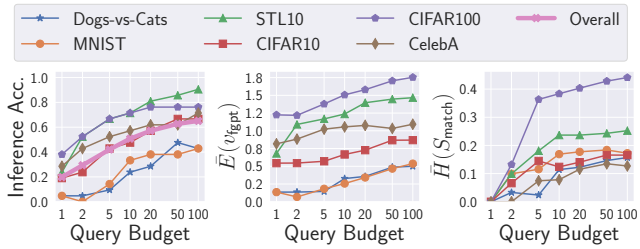


Figure 13: Performance with Random Noise as probing inputs. We have got the worst results on the three heuristics.

C Teacher Model with Latent Backdoor

In our experiment, we separate CIFAR10, MNIST, and STL10 with the same ratio with the `Digit` attack in [45]. To achieve a more robust backdoor attack, we increase the number of target images to 200, and the trigger size is 40×40 pixel. We exhibit some attack results for the STL10 dataset in Figure 14.

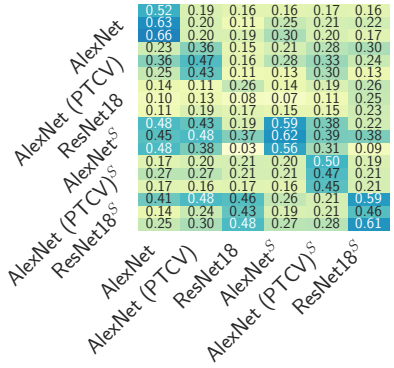


Figure 14: Teacher model fingerprinting vectors against clean and backdoored STL10 student models. *ModelName*^{S} refers to the backdoored model for STL10 dataset.

D Teacher Model Information

In our experiments, we have downloaded pre-trained models from the PyTorch official repository, including AlexNet, DenseNet121, MobileNetV2, ResNet18, VGG16, VGG19, and GoogLeNet. Besides, we have also downloaded another pre-trained AlexNet from the PyTorchCV repository to elaborate that our proposed attack can discriminate different teacher models with the same model architecture. The sources of our obtained teacher models are listed in Table 3.

E Transfer Learning Setup

The basic architecture of our black-box student model is one pre-trained feature extractor concatenated with fully con-

Model	Provider	$\mathcal{D}_\mathrm{T}$	#BLKs	Candidate?	URL	MD5 Checksum
AlexNet	PyTorch	ImageNet	5	✓	https://download.pytorch.org/models/alexnet-owt-4df8aa71.pth	aed0662f397a0507305ac94ea5519309
AlexNet (PTCV)	PyTorchCV	ImageNet	5	✓	https://github.com/osmr/imgclsmob/releases/download/v0.0.384/alexnetb-1900-55176c6a.pth.zip	07e23324e570d9e1f10e280a53c509e3
DenseNet121	PyTorch	ImageNet	11	✓	https://download.pytorch.org/models/densenet121-a639ec97.pth	a7047f0b44515469c3965e85cc31e512
MobileNetV2	PyTorch	ImageNet	18	✓	https://download.pytorch.org/models/mobilenet_v2-b0353104.pth	f20b50b44fdef367a225d41f747a0963
ResNet18	PyTorch	ImageNet	5	✓	https://download.pytorch.org/models/resnet18-5c106cde.pth	e5f5fcaec0feff7287f61b7bf461e8b2
VGG16	PyTorch	ImageNet	6	✓	https://download.pytorch.org/models/vgg16-397923af.pth	463aeb51ba5e122501bd03f4ad6d5374
VGG19	PyTorch	ImageNet	6	✓	https://download.pytorch.org/models/vgg19-dcbb9e9d.pth	92881fe292bd7d2408ecff58a101fd03
GoogLeNet	PyTorch	ImageNet	10	✗	https://download.pytorch.org/models/GoogLeNet-1378be20.pth	beba28483167f26c03ed26a6b569bbf9

Table 3: Sources of the teacher models used in our experiments.

nected layers. We adopt the whole convolution part of each pre-trained model listed in Table 3 as the feature extractor $\mathcal{F}_\mathrm{T}$, and then concatenate it with two or three fully connected layers. Particularly, for VGG16 and VGG19, we also adopt components except of the last layer from the pre-trained fully connected layers, which constitute the last teacher feature extractor block. For each feature extractor, we build up three different student models individually to perform a more comprehensive evaluation. For single-label classification tasks, we use the Softmax activation function at the student model output. While for the multilabel classification task on CelebA dataset, we use the Sigmoid activation function at the student model output. Table 4 reports the student model architectures.

In our experiments, we use the Adam optimizer [21] to train the student models. When the number of fine-tuned blocks is no larger than three we set the learning rate to 10^{-3}, and otherwise, we set the learning rate to 10^{-5}. Table 5 reports the average testing accuracy of the victim student models.

$\mathcal{D}_\mathrm{S}$	Model Architecture*
Dogs-vs-Cats	$\mathcal{F}_\mathrm{T} \to$ FC(128) $\to$ BN $\to$ SF(2)
	$\mathcal{F}_\mathrm{T} \to$ FC(512) $\to$ BN $\to$ FC(128) $\to$ BN $\to$ SF(2)
	$\mathcal{F}_\mathrm{T} \to$ FC(1024) $\to$ BN $\to$ FC(256) $\to$ BN $\to$ SF(2)
MNIST, STL10, CIFAR10	$\mathcal{F}_\mathrm{T} \to$ FC(128) $\to$ BN $\to$ SF(10)
	$\mathcal{F}_\mathrm{T} \to$ FC(512) $\to$ BN $\to$ FC(128) $\to$ BN $\to$ SF(10)
	$\mathcal{F}_\mathrm{T} \to$ FC(1024) $\to$ BN $\to$ FC(256) $\to$ BN $\to$ SF(10)
CIFAR100	$\mathcal{F}_\mathrm{T} \to$ FC(512) $\to$ BN $\to$ SF(100)
	$\mathcal{F}_\mathrm{T} \to$ FC(1024) $\to$ BN $\to$ FC(256) $\to$ BN $\to$ SF(100)
	$\mathcal{F}_\mathrm{T} \to$ FC(4096) $\to$ BN $\to$ FC(512) $\to$ BN $\to$ SF(100)
CelebA	$\mathcal{F}_\mathrm{T} \to$ FC(256) $\to$ BN $\to$ SG(40)
	$\mathcal{F}_\mathrm{T} \to$ FC(1024) $\to$ BN $\to$ FC(256) $\to$ BN $\to$ SG(40)
	$\mathcal{F}_\mathrm{T} \to$ FC(2048) $\to$ BN $\to$ FC(512) $\to$ BN $\to$ SG(40)

* For simplicity, FC(n) refers to a fully connected layer with n neuron. In our experiment, we choose the ReLU activation for fully connected layers, where are all followed by a dropout layer with rate 0.5. SF(n) refers to a Softmax layer with n outputs, SG(n) refers to a Sigmoid layer with n outputs, and BN refers to a batch normalization layer.

Table 4: Student model architectures used in our experiments.

Fine tuning*	$\mathcal{D}_\mathrm{S}$	Teacher model: AlexNet	AlexNet (PTCV)	Dense-Net121	Mobile-NetV2	ResNet18	VGG16	VGG19
Fixed	Dogs-vs-Cats	95.17	95.90	99.00	98.49	98.73	98.91	98.89
	MNIST	99.28	99.30	99.29	99.13	98.14	96.86	96.40
	STL10	75.55	84.14	96.80	95.05	94.95	92.27	92.25
	CIFAR10	62.35	71.39	91.80	89.87	88.02	71.57	67.50
	CIFAR100	30.71	40.30	68.42	64.17	60.52	29.30	32.05
	CelebA	87.65	87.95	88.27	87.73	86.33	85.59	85.62
Last BLK	Dogs-vs-Cats	96.66	97.09	98.65	98.21	98.33	98.67	98.73
	MNIST	99.43	99.47	99.21	98.98	99.38	99.09	99.10
	STL10	88.11	88.37	96.03	94.20	92.04	93.46	94.08
	CIFAR10	87.35	88.20	87.79	84.16	89.55	87.24	87.13
	CIFAR100	57.86	59.45	62.64	54.89	59.95	57.28	57.27
	CelebA	88.36	88.49	88.57	87.90	88.02	86.92	86.84
Last two BLKs	Dogs-vs-Cats	96.48	96.46	98.95	98.38	97.65	98.38	98.53
	MNIST	99.48	99.53	99.17	99.13	99.26	99.37	99.37
	STL10	86.10	85.40	95.34	94.03	86.37	91.08	91.28
	CIFAR10	87.69	88.28	90.22	85.84	89.30	89.17	88.10
	CIFAR100	58.32	56.50	65.70	56.55	54.52	48.02	41.71
	CelebA	88.81	88.59	88.41	88.13	88.33	88.01	88.00
Last three BLKs	Dogs-vs-Cats	95.07	95.53	98.83	98.33	96.88	98.13	96.27
	MNIST	99.52	69.57	99.20	99.24	99.42	99.10	99.41
	STL10	80.59	79.69	95.27	93.01	84.23	90.12	87.15
	CIFAR10	86.32	85.52	91.22	87.30	89.02	84.53	80.84
	CIFAR100	50.42	48.51	66.73	58.06	49.75	44.35	40.92
	CelebA	88.51	88.34	88.45	88.28	88.27	88.03	77.47
Last four BLKs	Dogs-vs-Cats	94.14	94.69	98.39	98.31	96.53	96.31	87.83
	MNIST	99.44	99.40	99.37	99.34	99.41	99.37	99.10
	STL10	75.69	71.31	91.56	92.86	77.40	87.22	85.94
	CIFAR10	86.29	86.37	91.55	87.64	89.07	76.10	66.99
	CIFAR100	48.88	43.89	62.26	57.79	43.77	44.65	45.47
	CelebA	88.67	88.47	88.60	88.14	88.17	87.97	87.57
Last five BLKs	Dogs-vs-Cats	97.31	97.61	99.19	98.73	98.87	99.04	99.09
	MNIST	99.56	99.56	99.50	99.53	99.60	99.61	99.59
	STL10	85.81	84.58	97.19	95.67	93.81	95.78	95.83
	CIFAR10	89.77	90.51	94.70	89.99	94.67	93.51	93.48
	CIFAR100	34.30	35.33	51.31	40.92	39.86	44.35	48.55
	CelebA	88.44	88.52	89.07	88.59	88.49	88.91	88.85

* For simplicity, "Fixed" refers to that pre-trained parameters are fixed, "Last BLK" refers to that the last block of the teacher feature extractor gets fine-tuned, "Last two BLKs" refers to that the two last blocks of the teacher feature extractor get fine-tuned, and so on.

Table 5: Average testing accuracy (%) of the victim student models in our experiments.

F Fingerprinting vectors for fine-tuned student models

In this part, we exhibit fingerprinting vectors when the last block gets fine-tuned and the last two blocks get fine-tuned in Figure 15 and Figure 16, respectively.

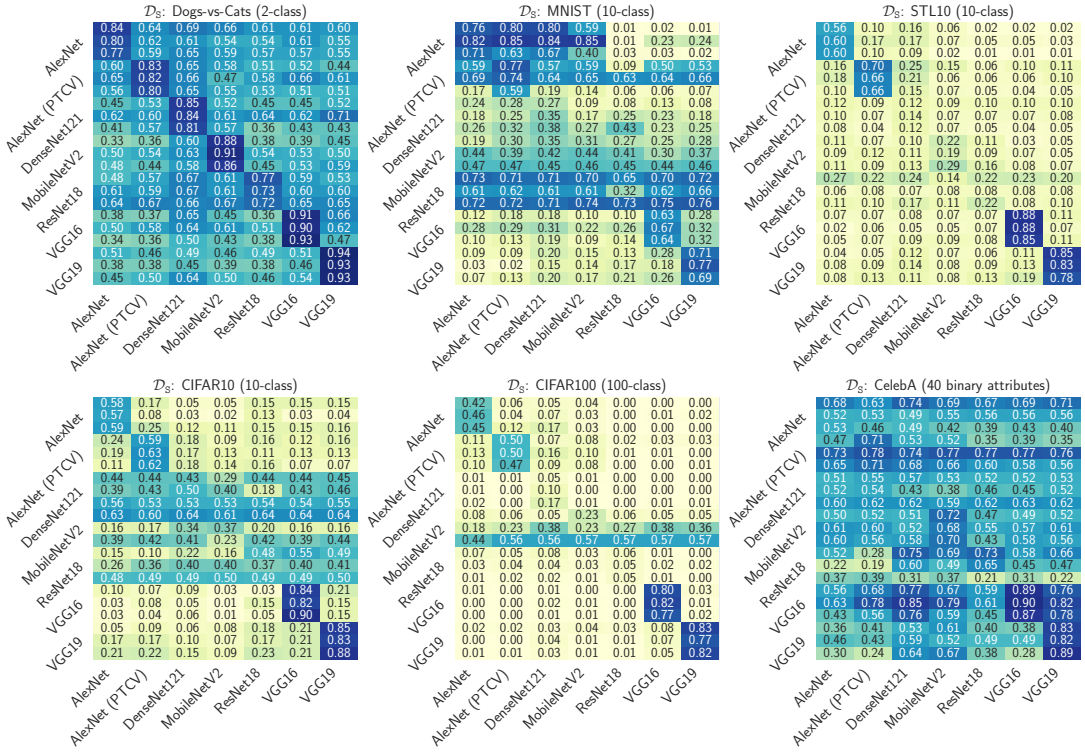


Figure 15: Teacher model fingerprinting vectors w.r.t. different classification tasks (100 fingerprinting pairs for each teacher model candidate, when the last block gets fine-tuned).

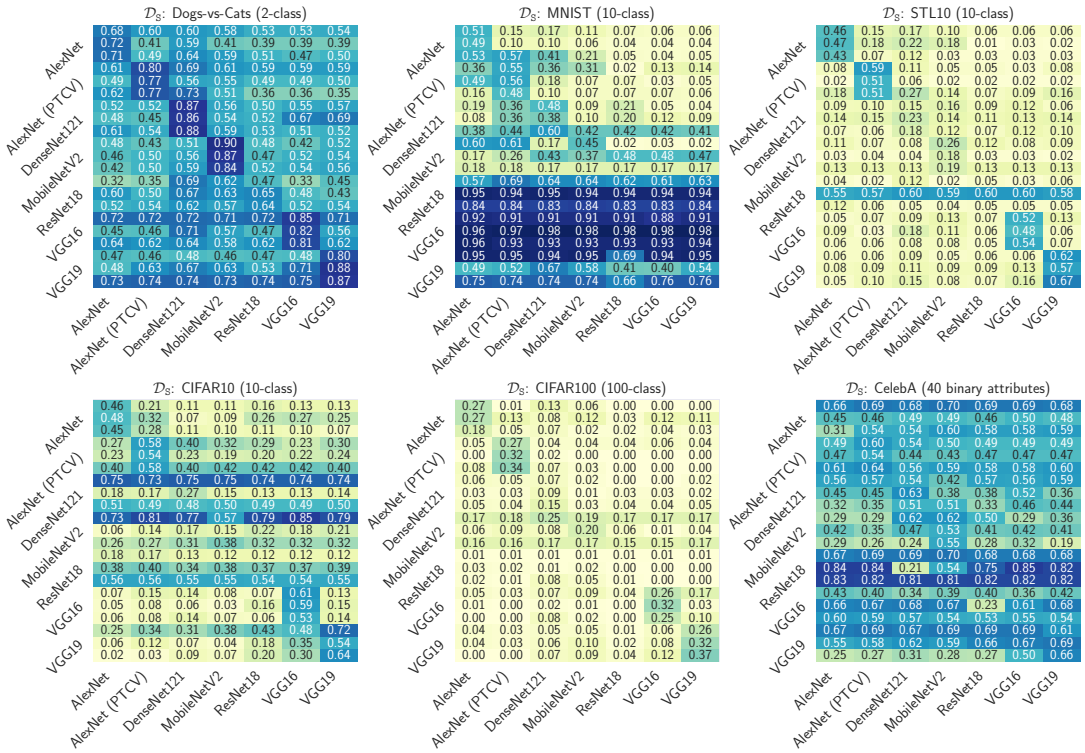


Figure 16: Teacher model fingerprinting vectors w.r.t. different classification tasks (100 fingerprinting pairs for each teacher model candidate, when the last two blocks get fine-tuned).