STEALING THE INVISIBLE: UNVEILING PRE-TRAINED CNN MODELS THROUGH ADVERSARIAL EXAMPLES AND TIMING SIDE-CHANNELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning, with its myriad applications, has become an integral component of numerous technological systems. A common practice in this domain is the use of transfer learning, where a pre-trained model's architecture, readily available to the public, is fine-tuned to suit specific tasks. As Machine Learning as a Service (MLaaS) platforms increasingly use pre-trained models in their backends, it's crucial to safeguard these architectures and understand their vulnerabilities. In this work, we present an approach based on the observation that the classification patterns of *adversarial images* can be used as a means to steal the models. Furthermore, the adversarial image classifications in conjunction with *timing side channels* can lead to a model stealing method. Our approach, designed for typical user-level access in remote MLaaS environments exploits varying misclassifications of adversarial images across different models to fingerprint several renowned Convolutional Neural Network (CNN) and Vision Transformer (ViT) architectures. We utilize the profiling of remote model inference times to reduce the necessary adversarial images, subsequently decreasing the number of queries required. We have presented our results over 27 pre-trained models of different CNN and ViT architectures using CIFAR-10 dataset and demonstrate a high accuracy of 88.8% while keeping the query budget under 20.

1 INTRODUCTION

The rapid growth of Machine Learning (ML) has transformed various industries. However, the complexity and resource intensity of developing in-house models have paved the way for Machine Learning as a Service (MLaaS) (Ribeiro et al., 2015). Companies like Google and Amazon provide businesses access to advanced, pre-trained ML/DL models via cloud services, eliminating the overhead of internal development and maintenance. However, the widespread use of MLaaS has amplified concerns around model privacy and security. These models, loaded with proprietary data and unique algorithms, are vital intellectual properties that offer competitive edge. In such a highly competitive environment, the security of these models is at risk due to the rise of techniques for reverse-engineering or "stealing" (Oliynyk et al., 2022). Increased research in model stealing poses a significant threat to the proprietary rights and market position of MLaaS providers.

A query-based attack is a common method for model stealing, where adversaries use a model's prediction Application Programming Interface (API)to recreate or "steal" (Oliynyk et al., 2022) it without direct access to its parameters or training data. Attackers generate a set of synthetic input samples or may already have access to data of similar distribution as that of the training data and send these to the model's prediction API. Further, by analyzing the predictions, thet attempt to reverse-engineer the model, often referred to as a black-box attack because the attacker has no knowledge of the model's internal workings, but can only access its input/output interface. Numerous studies have proposed attacks on diverse ML and DL models across various modalities, including text(Krishna et al., 2020; Li et al., 2023; Pal et al., 2020), images, and graphs(Wu et al., 2022; He et al., 2021; DeFazio & Ramesh, 2019; Shen et al., 2022). In particular for attacks on image modality which is the focus of this paper, there are certain works which try to steal the target model's complete architecture and parameters (Kariyappa et al., 2021; Rolnick & Kording, 2020; Roberts et al., 2019). On the other hand there are many works which create a substitute model by replicating the performance of the original target model (da Silva et al., 2018; Kariyappa et al., 2021; Li et al., 2018; Mosafi et al., 2019; Orekondy et al., 2019; Papernot et al., 2017; Yuan et al., 2022). Notably, the success and practicality of query-based attacks hinges on the *query budget*, which limits the number of queries one can make to an ML model in a set period to manage resources and enhance security. Hence for an attacker, reducing query numbers is vital, as excessive queries can raise alarms, leading to service suspension and a thwarted attack.

In addition to query-based attacks, there exists another category of model stealing attacks that leverage side-channel or microarchitectural leakages to extract details about the model's architecture and parameters. A substantial amount of research has focused on using side-channel information in remote settings to reverse-engineer the architecture and parameters of proprietary Deep Neural Networks (DNNs). Various studies have leveraged cache-based side-channels to recreate essential architectural secrets of the target DNN during the inference phase, using the Generalized Matrix Multiply (GEMM) operation in the DNN's implementation (Hong et al., 2018; Yan et al., 2020). Cache memory access patterns have also been exploited to gain layer sequence information of CNN models and thereafter the complete architecture by utilizing LSTM-CTC model and GANs as well. (Hu et al., 2020; Liu & Srivastava, 2020). Other investigations have exploited shared GPU resources, hardware performance counters, and GPU context-switch side-channels to extract the internal DNN architecture of the target (Naghibijouybari et al., 2018; Wei et al., 2020). Additionally, some researchers have shown that it is possible to steal critical parameters of the target DNN by exploiting rowhammer fault injections on DRAM modules (Rakin et al., 2021). Timing side-channels have been utilized to both build an optimal substitute architecture for the victim and extract DL models on high-performance edge deep learning processing units (Duddu et al., 2018; Batina et al., 2019; Won et al., 2021). Few works have also leveraged side-channel information like power (Wei et al., 2018; Yoshida et al., 2020), electromagnetic emanation (Batina et al., 2019; Yu et al., 2020; Chmielewski & Weissbart, 2021), and off-chip memory access (Hua et al., 2018) to reverse engineer architectural secrets of DNNs, which require physical access to the model.

Companies that offer APIs for specialized applications, often use models fine-tuned from standard pre-trained models like AlexNet or ResNet that are available publicly. Some research works aim to identify these underlying pre-trained/teacher models in the backend of MLaaS platforms. One such work has proposed a query based model stealing attack that relies on analyzing the classification outputs of customized synthetic input images introduced to the model with a minimum requirement of atleast 100 queries for good attack accuracy (Chen et al., 2022). From a side-channel perspective, a recent study employed GPU side-channels to identify pre-trained model architectures, but this approach used `nvprof` GPU profiler, which is mostly disabled on cloud platforms providing the MLaaS services (Weiss et al., 2023). In other work, user accessible CPU and GPU memory side-channel information were exploited to perform DNN fingerprinting on CPU-GPU based edge devices, but these won't work in cloud settings where the client only gets an API response from the server and cannot access any other information about the system (Patwari et al., 2022). *Our research is the first to combine both query based as well side-channel model stealing attack methodologies to determine the pre-trained models deployed in the backend of an MLaaS platform while only possessing client or user privileges and limiting the query requirements to less than* 20 *queries.*

With high influx of works on model stealing attacks, defending machine learning models against theft has also become of paramount importance. Various techniques, such as rate limiting and incorporating noise into the output predictions, have been devised to prevent these attacks. However, these strategies have their limitations and can impact the service utility for legitimate users. An emerging technique in the field of model IP protection is watermarking or fingerprinting models (Regazzoni et al., 2021; Lederer et al., 2023). Recently many works have also utilized adversarial examples for the same (Xue et al., 2021; Szyller et al., 2021; Zhao et al., 2020). This method works by embedding unique perturbations into the model during its training phase, which can be used as identifiers. These adversarial examples - inputs that are intentionally designed to induce model errors - act as the model's unique fingerprints and can be used as markers of authenticity. However, the fact that adversarial examples can be used to fingerprint the ML models also poses the looming danger of being used as a means of model stealing.

Adversarial example show an intriguing property of transferability (Liu et al., 2017), observed in machine learning models, particularly deep neural networks (DNNs). This property means that an adversarial example, originally designed for a specific machine learning model, can also affect other models, leading to successful misclassifications. In this work, we demonstrate and emphasize on the fact that, although adversarial examples can transfer between models, they may not necessarily be classified into the same class as the initial model due to difference in the decision boundaries of various models. We exploited these divergent misclassifications of adversarial images from different models to fingerprint several renowned pre-trained CNN architectures.

We work with assumption that the adversary does not have any knowledge about target model's architecture as well as weight parameters. For this, we utilize a window of top classifications from the MLaaS server. For each architecture we profile with multiple models of varying weight parameters to better classify the target model. *It is to be noted, our work is the first one to exploit adversarial image classification pattern among various CNN architectures to reverse-engineer CNN models.* Our work shows that an effective combination of adversarial image selection and timing based side-channels can be used to discern the target CNN models, with as little as 15 observations, thus reducing the query budget requirement for the attack.

This strategy significantly helps in reducing our query requirements, allowing us to maintain it below ten queries for a successful attack. We have shown results for our attack with the standard CIFAR10 (Krizhevsky et al., 2009) dataset. Furthermore, we have worked with 27 pre-trained models of different CNN and ViT architectures provided by PyTorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2019) respectively. Next, we summarize the contribution of this work:

- We observe that while transfer learning of CNN architectures allow adversarial attacks the target classes are distinct and can be used for fingerprinting.
- We present a 2-staged model stealing attack, exploiting the remote inference timing side-channels in the first stage to shortlist potential architectures and prediction pattern of adversarial images in the second stage for final prediction.
- We show through extensive experiments on 27 pretrained models available on PyTorch and HuggingFace using CIFAR10 dataset that our model stealing attack works accurately even in situations where the weights of the target model vary significantly compared to state-of-the-art.

2 Recognizing Adversarial Images as Architecture Identifiers

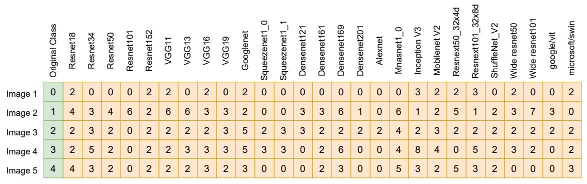
We are aware of the transfer-ability property inherent in adversarial examples, whereby an adversarial image generated to induce misclassification in a particular Machine Learning (ML) model may also succeed in causing misclassification when presented as input to other ML models. We emphasize that while transfer-ability implies that the misclassification extends across models the resulting misclassified class is not the same. In this section, we delve into these misclassification patterns observed among various pre-trained, or teacher models and evaluate their ability of fingerprinting these models and subsequently exploit it to orchestrate a model extraction attack on Machine Learning as a Service (MLaaS) servers. A comprehensive exploration of these adversarial examples application is discussed in Sections 4.

Experimental Setup: We perform our experiments using a total 27 pre-trained image classification models including both CNN and ViT architectures. We show the list of models in Table 1, where we group the models under different architecture types. While the experiments were performed using PyTorch, it should be noted that they are not limited to this particular Deep Learning (DL) framework and can be replicated using alternative frameworks such as TensorFlow and Caffe. For the adversarial example generation we use the three well-known algorithms namely, Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD) and Basic Iterative Method (BIM). We use CIFAR-10 dataset for model training and adversarial examples generation. The models are finetuned for these datasets over the pre-trained models provided by PyTorch, which are originally trained on Imagenet-1k dataset.

Consider a scenario where we have a set of models $M = \{M_1, M_2, \ldots, M_Z\}$. From this set, we select a single model, say M_i, $1 \leq i \leq Z$ as the base model to generate N adversarial examples employing any of the recognized adversarial attack strategies. We then give these N adversarial images as input to all the remaining Z models and observe the classification pattern for each image. The primary objective is to determine whether these classifications are uniform across all models or whether they show variation. For our initial experiment, we have $Z = 27$, M_i is `Resnet-18` model and we generate $N = 1000$ adversarial images of CIFAR-10 dataset using FGSM, BIM and PGD adversarial attacks. In Figure 1 we show classification for a sample of 5 adversarial images

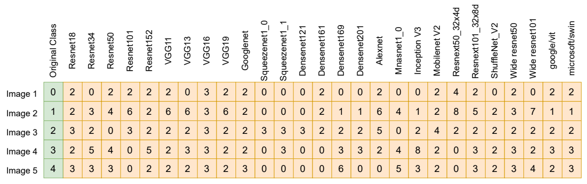
Table 1: List of Models and their Groups

Group Name	Models
AlexNet (Krizhevsky et al., 2012)	AlexNet
VGG (Simonyan & Zisserman, 2015)	VGG-11, VGG-13, VGG-16, VGG-19
ResNet (He et al., 2016)	Resnet-18, Resnet-34, Resnet-50, Resnet-101, Resnet-152
Squeezenet (Iandola et al., 2016)	Squeezenet1.0, Squeezenet1.1
Densenet (Huang et al., 2017)	Densenet-121, Densenet-161, Densenet-169, Densenet-201
Inception (Szegedy et al., 2016)	Inception v3
GoogleNet (Szegedy et al., 2015)	GoogleNet
ShuffleNet (Ma et al., 2018)	ShuffleNet V2
MobileNet (Sandler et al., 2018)	MobileNet V2
ResNeXt (Xie et al., 2017)	ResNeXt-50-32x4d, ResNeXt-101-32x8d
Wide ResNet (Zagoruyko & Komodakis, 2016)	Wide ResNet-50-2, Wide ResNet-101-2
MNASNet (Tan et al., 2019)	MNASNet 1.0
Google ViT (Wu et al., 2020)	google/vit-base-patch16-224-in21k
Microsoft Swin (Liu et al., 2021)	microsoft/swin-base-patch4-window7-224



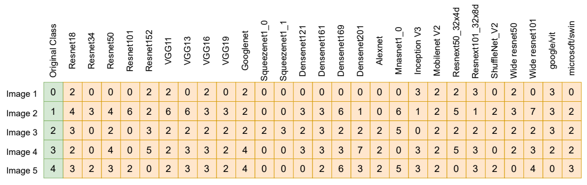
	Original Class	Resnet18	Resnet34	Resnet50	Resnet101	Resnet152	VGG11	VGG13	VGG16	VGG19	Googlenet	Squeezenet1_0	Squeezenet1_1	Densenet121	Densenet161	Densenet169	Densenet201	Alexnet	Mnasnet1_0	Inception V3	Mobilenet V2	Resnext50_32x4d	Resnext101_32x8d	ShuffleNet_V2	Wide resnet50	Wide resnet101	google/vit	microsoft/swin
Image 1	0	2	0	0	0	2	2	0	2	0	2	0	0	0	0	0	0	0	0	3	2	2	3	0	2	0	0	2
Image 2	1	4	3	4	6	2	6	6	3	3	2	0	0	3	3	6	1	0	6	1	2	5	1	2	3	7	3	0
Image 3	2	2	3	2	0	2	2	2	2	3	5	2	3	3	2	2	2	2	4	2	3	2	2	2	2	2	2	2
Image 4	3	2	5	2	0	2	2	3	3	3	5	3	3	0	2	6	0	0	4	8	4	0	5	2	3	2	0	2
Image 5	4	4	3	2	0	2	2	2	3	2	3	0	0	0	2	3	0	0	5	3	2	5	3	2	0	0	0	3

(a) FGSM



	Original Class	Resnet18	Resnet34	Resnet50	Resnet101	Resnet152	VGG11	VGG13	VGG16	VGG19	Googlenet	Squeezenet1_0	Squeezenet1_1	Densenet121	Densenet161	Densenet169	Densenet201	Alexnet	Mnasnet1_0	Inception V3	Mobilenet V2	Resnext50_32x4d	Resnext101_32x8d	ShuffleNet_V2	Wide resnet50	Wide resnet101	google/vit	microsoft/swin
Image 1	0	2	0	2	2	2	2	0	3	2	2	0	0	0	2	0	0	2	0	0	2	4	2	0	2	0	2	2
Image 2	1	2	3	4	6	2	6	6	3	6	2	0	0	0	2	1	1	6	4	1	2	8	5	2	3	7	1	1
Image 3	2	3	2	0	3	2	2	2	3	2	2	3	3	3	2	2	2	5	0	2	4	2	2	2	2	2	2	2
Image 4	3	2	5	4	0	5	2	3	3	2	2	0	3	0	0	3	3	2	4	8	2	0	3	2	3	2	3	3
Image 5	4	3	3	3	0	2	2	2	3	2	3	0	0	0	0	6	0	0	5	3	2	0	3	2	3	4	2	3

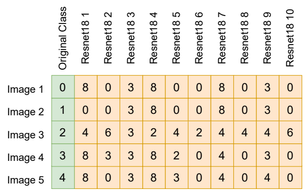
(b) PGD



	Original Class	Resnet18	Resnet34	Resnet50	Resnet101	Resnet152	VGG11	VGG13	VGG16	VGG19	Googlenet	Squeezenet1_0	Squeezenet1_1	Densenet121	Densenet161	Densenet169	Densenet201	Alexnet	Mnasnet1_0	Inception V3	Mobilenet V2	Resnext50_32x4d	Resnext101_32x8d	ShuffleNet_V2	Wide resnet50	Wide resnet101	google/vit	microsoft/swin
Image 1	0	2	0	0	0	2	2	0	2	0	2	0	0	0	0	0	0	0	0	3	2	2	3	0	2	0	3	0
Image 2	1	4	3	4	6	2	6	6	3	3	2	0	0	3	3	6	1	0	6	1	2	5	1	2	3	7	3	2
Image 3	2	3	0	2	0	3	2	2	2	2	2	2	3	2	3	2	2	2	5	0	2	2	2	2	2	2	3	2
Image 4	3	2	0	4	0	5	2	3	3	2	4	0	0	3	3	3	7	2	0	3	2	5	3	0	2	3	2	2
Image 5	4	3	2	3	2	0	2	3	3	2	4	0	0	0	2	6	3	2	5	3	2	0	3	2	0	4	0	3

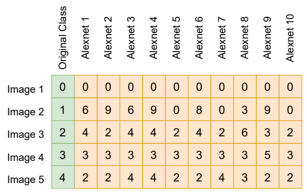
(c) BIM

Figure 1: Varying classification for 5 adversarial images generated using FGSM, PGD, and BIM attacks, belonging to 5 different classes of CIFAR-10 dataset for 27 pre-trained models



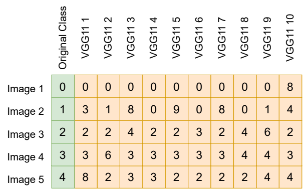
	Original Class	Resnet18 1	Resnet18 2	Resnet18 3	Resnet18 4	Resnet18 5	Resnet18 6	Resnet18 7	Resnet18 8	Resnet18 9	Resnet18 10
Image 1	0	8	0	3	8	0	0	8	0	3	0
Image 2	1	0	0	3	8	0	0	8	0	3	0
Image 3	2	4	6	3	2	4	2	4	4	4	6
Image 4	3	8	3	3	8	2	0	4	0	3	0
Image 5	4	8	0	3	8	3	0	4	0	4	0

(a) Resnet-18



	Original Class	Alexnet 1	Alexnet 2	Alexnet 3	Alexnet 4	Alexnet 5	Alexnet 6	Alexnet 7	Alexnet 8	Alexnet 9	Alexnet 10
Image 1	0	0	0	0	0	0	0	0	0	0	0
Image 2	1	6	9	6	9	0	8	0	3	9	0
Image 3	2	4	2	4	4	2	4	2	6	3	2
Image 4	3	3	3	3	3	3	3	3	3	5	3
Image 5	4	2	2	4	4	2	2	4	3	2	2

(b) Alexnet



	Original Class	VGG11 1	VGG11 2	VGG11 3	VGG11 4	VGG11 5	VGG11 6	VGG11 7	VGG11 8	VGG11 9	VGG11 10
Image 1	0	0	0	0	0	0	0	0	0	0	8
Image 2	1	3	1	8	0	9	0	8	0	1	4
Image 3	2	2	2	4	2	2	3	2	4	6	2
Image 4	3	3	6	3	3	3	3	3	4	4	3
Image 5	4	8	2	3	3	2	2	2	2	4	4

(c) VGG11

Figure 2: Classification of adversarial images generated using PGD with models of the same architectures but different weight parameters

(out of total 1000 images) from 5 different classes of CIFAR-10 dataset on 27 pre-trained CNN and ViT models. The classification for adversarial images generated using FGSM, PGD and BIM are shown in Figure 1a, Figure 1b, and Figure 1c respectively. For each adversarial image, we observe that the misclassified label by each model is not same and it varies for all the models. This trend is consistent for not only the five images depicted in Figure 1 but also for the entire set of 1000 adversarial images. Based on our observation, in Section 4 we demonstrate how we can leverage the unique misclassification trend among different pre-trained models to fingerprint them and then execute a successful model extraction attack. However, prior to that, we furnish the threat model in the following section.

3 Threat Model

In this section we define the threat model for the proposed model extraction attack. We consider an MLaaS scenario, where a ML service provider provides API access to one of the trained ML model which has been deployed on the cloud server, to all the authorized clients. The adversary is also an authorized client of the service. The clients have no knowledge about the ML model's architecture running on the server. The adversary does not have knowledge about the target model's architecture, and further it does not have information about the weights as well.

Adversary's capabilities: Unlike other works (Weiss et al., 2023; Patwari et al., 2022) the adversary only has API access to the MLaaS model, through which it's impossible to use any CPU and GPU profiling tools on the server. Additionally, the adversary has only client-level privileges, and can get the execution time of each image's inference query to the model. The adversary has access to publicly available pre-trained models which he can fine-tune for particular datasets. The adversary has access to the dataset belonging to the same distribution as target model's training data.

Adversary's Objective: The primary objective of the adversary is to discern the pre-trained model utilized in training the target model that operates on the MLaaS server. The adversary seeks to extract the model by analyzing the classifications of the input image and its corresponding inference time, and ensuring minimum possible queries to the MLaaS.

4 Model Fingerprinting using Adversarial Examples and Timing Side-channel

In this section, we extend the discussion from Section 2, where we highlighted the non-uniform classifications by pre-trained models on adversarial images. We illustrate how to identify a minimal subset of adversarial images that can effectively profile all the pre-trained models. Additionally, we demonstrate that by leveraging timing side-channels, we can further reduce the size of this minimal adversarial set. This is achieved by focusing on models whose inference time aligns closely with that of the target model operating on MLaaS server.

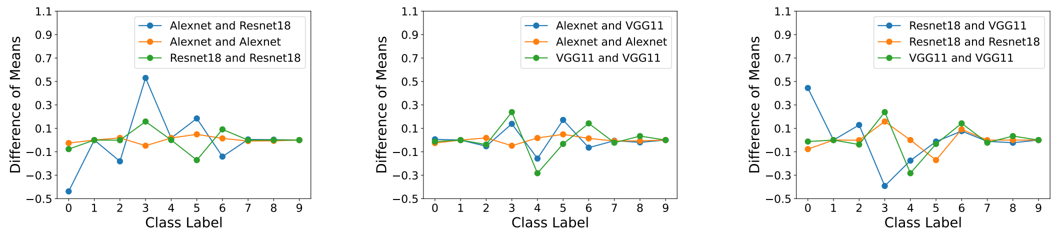
4.1 Model Profiling with Adversarial Images

In Section 2 we showed varying classification patterns for different model architectures, but model for each architecture had a specifc set of weights which in realistic scenario won't be same even though the architecture remains same. This is because various possible initial parameters which are set before training may vary for different models. Thus, we work under the assumption of a weight-oblivious adversary, implying that we are unaware of the model architecture and its weights. For such an adversary, it is essential to create profiles for numerous models with the same architecture but with differing weight parameters.

We have total of Z different model architectures. We train k models of each architecture with varying weights. We generate a set of N adversarial images and generate classification for all k models for each architecture. For our experiment, without loss of generality, we took $Z = 27$, $k = 10$ and $N = 1000$. In Figure 2, we show classification for 5 (out of 1000) adversarial images of CIFAR-10 dataset with 3 models, namely Alexnet, Resnet-18 and VGG11. It is visible from the figure that the the classification for each image is not consistent across all models of the same architecture. Furthermore, where classifications appear consistent—for example, for the class 0 image—the results are comparable across all three architectures, making it difficult to distinguish between different architectures using such images. Consequently, we chose to evaluate the top-5 classifications of the model, rather than just the top-1. This approach provided us with deeper insights into the varied classification patterns of the different models.

We show this by again taking the example of three architectures, Alexnet, Resnet-18, and VGG11. We trained 10 models of each architecture with different weight parameters. Next, for a particular adversarial image generated using a CIFAR-10 image with PGD, we collected top-5[1] classifications for all the 30 models. We then calculated class-wise probability means for each architecture. As a result, for every architecture, we had 10 mean values corresponding to the 10 CIFAR-10 classes. The final step is to calculate class-wise *difference of means (DoMs)* for each architecture pair, for

[1]We choose top-5 classifications as it is a common practice in MLaaS environments.



(a) (b) (c)

Figure 3: Comparison of Class-wise Difference of Means (DoMs) of classification probabilities between (a) Alexnet and Resnet18, (b) Alexnet and VGG11, and (c) Resnet18 and VGG11 with intra-architecture DoMs

which we have provided three plots in Figure 3. The blue line plots are for inter-architecture DoMs. Subsequently, we also show plots for intra-architecture DoMs, for which we compare first 5 models of a given architecture with the other 5 models of the same architecture. From Figure 3, it is evident that this specific adversarial image serves well in distinguishing between Alexnet and Resnet-18 models, as well as Resnet-18 and VGG11 models. This is observable through the high DoMs for classes 0 and 3 in both the Alexnet/Resnet-18 pair and the Resnet-18/VGG11 pair. However, for the Alexnet and VGG11 pair, there isn't a significant difference in DoMs when comparing inter and intra-architecture models.

We now formulate our methodology to discern the target model's architecture by utilizing top-5 classification information for some adversarial images. We have total of Z architectures and k models for all architectures with different weight parameters which were trained earlier. Let I_x be an image from the adversarial image set $\{I_1, I_2, \ldots, I_N\}$. We first get classification for all $Z \times k$ models for the image I_x. For each model we get a vector of 5 probabilities for top-5 class labels. Next, we transform these vectors into vectors of size $|L|$, where L is the set of class-labels for any chosen dataset. In each of this vector, we place the probabilities of top-5 classes at their index values, and all other values are set to zero. With these steps our template data for all models for a particular adversarial image I_x is ready. Now, for an unknown target model we pass the image I_x and get the top-5 classification. We convert the result to a vector of size $|L|$ as before. The next step is to discern the architecture of the target model, by comparing the target model's generated classification vector with the prior created template. This prediction will be based on template created for one adversarial image. We select W adversarial images and then create template for them. We perform majority voting on the results from template of each image. The next question that arises is how we do we decide on which images to choose for template creation and we delve into it in the next subsection.

4.1.1 Adversarial Image set selection for Template creation

So far, we have explored how to identify the target model by utilizing its classification of an adversarial image. The next question we address is how to determine the most effective adversarial image from a given set for this specific task. Furthermore, we need to decide the number of such images to be selected to ensure the best results with majority voting, while also optimizing this quantity to maintain the query budget.

We have a set of Z potential target architectures. For each of these Z architectures k models have been trained with different weight parameters. Furthermore, we have top-5 classification vectors transformed into vectors of size $|L|$ for these models on a set of N adversarial images. Our objective is to identify the top d images that shows the highest distinguishing ability among the Z architectures. To achieve this, we first compute the element-wise mean of the classification vectors from the k models for each adversarial image across all architectures. This results in Z vectors of size L for each adversarial image. We then calculate the Euclidean distance between each pair of architecture vectors for each adversarial image and sum these distances across all pairs. The adversarial images are ranked in descending order based on this sum.

We have established a method to select the the adversarial images which can be used to distinguish between various architectures, but the number of images which we require will depend on the total number or architectures Z. Finally, we select the top d images, those that have the highest summed Euclidean distances, as these are the images that are most effective in distinguishing the different CNN architectures. We have elaborated our methodology in a systematic manner in Algo-

Algorithm 1 Adversarial Image Selection for Model profiling in Black-box setup

```
1: $N \leftarrow$ number of adversarial images
2: $Z \leftarrow$ number of architectures
3: $k \leftarrow$ number of models per architecture
4: $L \leftarrow$ size of classification vector
5: $d \leftarrow$ number of images to select
6: $ED()$: Euclidean Distance calculation
7: Initialize $V_{i,j,p}$
8: for $i = 1$ to $N$ do
9:     for $j = 1$ to $Z$ do
10:        for $p = 1$ to $k$ do
11:            $V_{i,j,p} \leftarrow$ Classify$(i, A_j, p)$
12:        end for
13:    end for
14: end for
15: Initialize $D_i$
16: for $i = 1$ to $N$ do
17:     for $j = 1$ to $Z$ do
18:         $V_{i,j} \leftarrow \frac{1}{k}\sum_{p=1}^{k} V_{i,j,p}$
19:     end for
20:     $D_i \leftarrow \sum_{j=1}^{Z-1}\sum_{m=j+1}^{Z} ED(V_{i,j}, V_{i,m})$
21: end for
22: Sort $D_i$ in descending order
23: return top $d$ indices from $D_i$
```

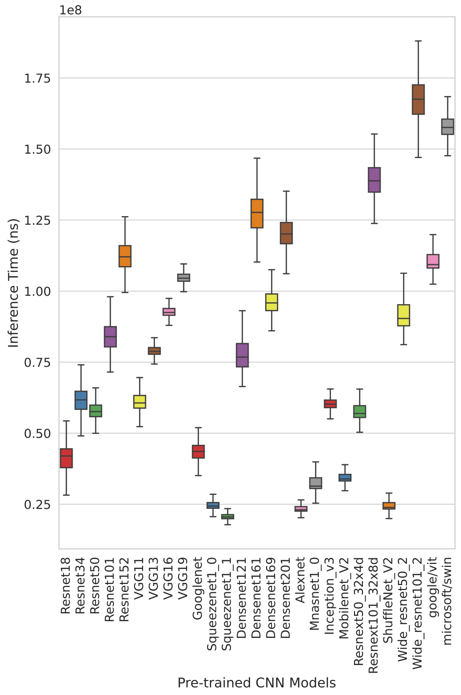
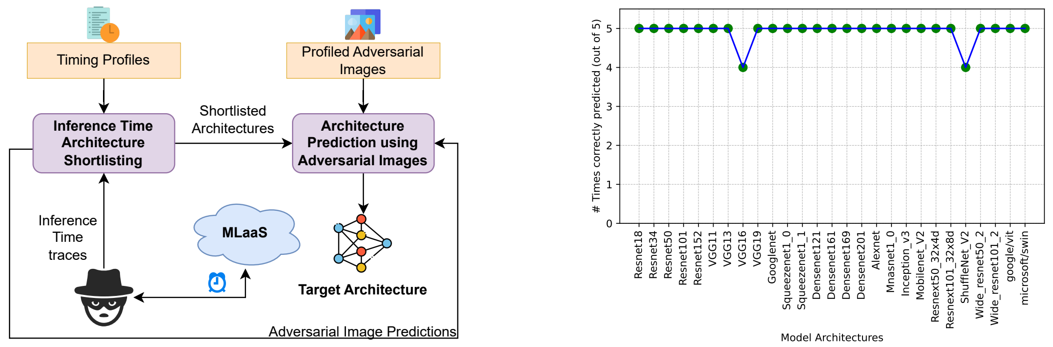


Figure 4: Box-plot for inference time distributions for 27 Pre-trained models on CIFAR-10

rithm 1. This is because to distinguish between a larger number of architectures, the requirement for the number of distinguishing images will also increase which in turn means higher query budget requirement. Also this will hamper the architecture predictability performance of our proposed approach. To address this we came up with a methodology which uses model inference times to first shortlist the potential target models, making Z smaller and then applying our adversarial image selection algorithm. We discuss this in detail in the next sub-section.

4.2 TIMING PROFILES

The architecture of all vision models varies in terms of the number of layers, layer types, and other parameters. Primarily, the inference time of any CNN model depends on the network's depth. This is equally applicable to publicly available pre-trained CNN and ViT models. In Figure 4, we display the box-plots representing timing distributions of 27 pre-trained models, including 25 CNN from PyTorch and 2 transformer models from HuggingFace, fine-tuned on the CIFAR-10 dataset. We use `perf_counter_ns` function from the `time` Python package to collect these timings. Each timing distribution is obtained by measuring the inference time of 100 images of differing classes. Each image is processed through the model 100 times, resulting in a total of $10,000$ timing values for each distribution. The plot clearly shows that the inference times for all the 27 models vary significantly. We notice that some models have timing ranges that partially intersect with those of others. Thus, inference time alone cannot serve as a distinctive factor between the pre-trained models. However, we can use it as a criterion to filter the potential target models whose timing aligns closely with the target model's timing. Subsequently, we can utilize Algorithm 1 from Section 4.1.1 to identify the minimum set of adversarial images, which can help recognize the target model among the shortlisted ones based on inference time. It is crucial to note that by trimming down the models to create a smaller pool of possible target models, which further helps in reducing the number of adversarial images required to distinguish between shortlisted architectures. This transition further aids in lowering the query budget for the model extraction attack. In Figure 4 we observe the maximum intersection in box-plots for 6 models namely, *Resnet34, Resnet50, VGG11, VGG13, Inception-V3* and *Resnext101-32-4d* is still less than the total number of class labels in CIFAR-10 dataset.



(a) (b)

Figure 5: (a) Attack Methodology (b) Shortlisting correctness based on inference time for 5 test CNN target models of each architecture with varying weight parameters trained using CIFAR10.

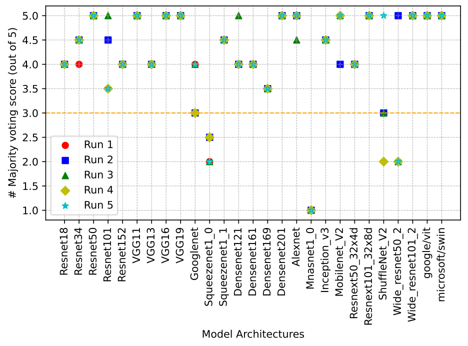
4.2.1 Shortlisting models given unknown target model's inference time

To begin with, we will gather timing traces for all the available pre-trained models and store their maximum and minimum values range as the timing profile for each model. For every model, we will collect the inference time for different class images from the dataset, then jointly calculate their maximum and minimum range, providing a complete timing range for all image types. Let us denote the number of models as Z. For any model i, the min-max timing range is defined as (MIN_i, MAX_i). We now outline the procedure for narrowing down potential target models, given the inference time of the original unknown model. Consider an unknown target model, denoted as X. The inference time for this model is represented as T_X. We use T_X as a basis to select models whose min-max range encompasses T_X. This procedure is formally outlined in Algorithm 2 (Appendix A).

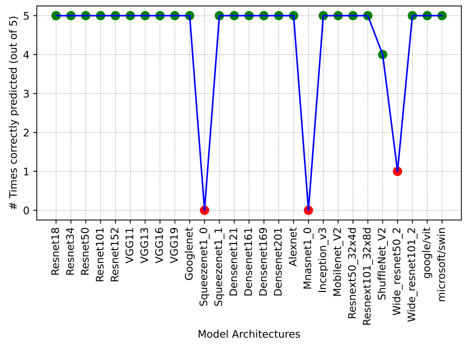
Once we have the set of potential target models, we can employ Algorithm 1 to get the minimum set of adversarial examples for the particular set of models. Then we pass the final selected images to the target model and discern it s architecture based on the model's classification outputs. The final prediction of the target model is determined through a process of majority voting. In this step, adversarial images chosen for a pre-selected group of architectures are inputted into the target model, which then yields the top-n class predictions. For every prediction made from an adversarial image, we measure the Euclidean distance to all profiled model architectures and pinpoint the one with the closest proximity. In the final phase, a comprehensive majority voting is conducted, taking into account all the models predicted by the selected adversarial images. The outcome of this collective voting determines the final prediction. The final methodology is shown in Figure 5a.

5 Experimental Results and Discussion

In this section, we discuss the results for the model extraction methodologies explained in the previous section. We have performed our experiments using 27 pre-trained models including 25 CNN models provided by PyTorch via the `torchvision.models` package and 2 Vision Transformer models from HuggingFace. We further fine-tuned these models for CIFAR-10 dataset. All experiments throughout the paper have been performed on an Intel Xeon Silver 4214R CPU system with 128 GB RAM. We discuss our results with assumption of an adversary, who has no knowledge about the architecture as well as weight parameters of the target model. In this scenario, we would require multiple models to profile any particular architecture, and hence for each architecture we trained total of 15 models with varying weights for each architecture using the CIFAR-10 dataset. Among these 15 models we used 10 models for fingerprinting each architecture whereas we kept the 5 other models for testing purpose. It is to be noted that all the models have been fine-tuned on the pre-trained models, which means that the weights have been modified in all the layers of the model and not just the last layers as assumed in the prior work (Chen et al., 2022). The initial step involves profiling the inference time for each model across all architectures. We access all the model architectures remotely over an API call using the FLASK API setup. We discuss the results in detail in the next subsection.



(a)



(b)

Figure 6: (a) Majority voting score for different architectures across 5 prediction runs. (b) Number of correctly predicted test models of different CNN architectures using CIFAR10 dataset

5.1 Mapping Inference Time to Timing Profiles

For every architecture, we record the inference times remotely through the API calls for all the trained models within that architecture and consolidate them. For our experiments, we have collected 10 timing traces for each model, thus accumulating a total of 100 timing values per architecture. For the target model, we collect 10 timing traces and then use Algorithm 2 to narrow down potential target models. To check the reliability of our approach we collected the timing traces for each of the 5 models of each architecture set apart earlier. In Figure 5b we show the number of times the actual target architecture got shortlisted for all the test models. We observe that out of total 27 architectures, 25 architectures are shortlisted with 100% accuracy, whereas 2 architectures, VGG-16 and Shufflenet_V2 are correctly clasified 4 out of 5 times. Overall 98.5% models are correctly shortlisted based on inference time. Now we move on to the next subsection, where we discern the target model from the shortlisted models using specifically chosen adversarial images.

5.2 Adversarial Image Set Selection

We'll now employ the approach defined in Section 4.1.1 to select the images which best help in distinguishing the group of shortlisted architectures for each of the 27 target architectures. We select these images utilizing adversarial image classifications from 10 models of varying weights from each architecture. Subsequently, we check the reliability of this approach using the test models set apart earlier for each architecture. We first select top 5 adversarial images suitable for distinguishing the shortlisted architectures using Algorithm 1. Then we apply the majority voting to get the final target model prediction. In Figure 6a, we show the majority voting scores for 5 test models of each architecture over different runs. Finally in Figure 6b we show the number of correctly classified models out of the 5 test models for each target architecture after the majority voting. We observe that out of total 27 architectures 24 of them show 100% correct prediction for the 5 test models, whereas for Wide_resnet101_2 there are 4 correct predictions. Overall, we tested for $135 = 27 * 5$ models of different architecture, and we get an average accuracy of 88.8% and maximum accuracy of 92.59% for correct predictions across different runs.

6 Conclusion

In this study, we delved into the intriguing property of adversarial examples in machine learning models, with a focus on CNNs. We discovered that adversarial examples could influence the classification of various models, but don't always trigger the same misclassification due to differing decision boundaries. Utilizing this, we developed a unique fingerprinting method for renowned pre-trained CNN and ViT architectures. Furthermore, we employed timing side-channels to minimize the number of adversarial image queries required for identifying the target model. This approach greatly reduced the queries needed, typically to fewer than 20. Moreover, we demonstrate that, despite fine-tuning all layers of the pre-trained model, we successfully beat the state-of-the-art work on model fingerprinting by correctly classifying 88.8% of models from varying architectures correctly.

REFERENCES

Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. CSI NN: reverse engineering of neural network architectures through electromagnetic side channel. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pp. 515–532. USENIX Association, 2019. URL `https://www.usenix.org/conference/usenixsecurity19/presentation/batina`.

Yufei Chen, Chao Shen, Cong Wang, and Yang Zhang. Teacher model fingerprinting attacks against transfer learning. In *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pp. 3593–3610. USENIX Association, 2022. URL `https://www.usenix.org/conference/usenixsecurity22/presentation/chen-yufei`.

Lukasz Chmielewski and Leo Weissbart. On reverse engineering neural network implementation on GPU. In *Applied Cryptography and Network Security Workshops - ACNS 2021 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, and SiMLA, Kamakura, Japan, June 21-24, 2021, Proceedings*, volume 12809 of *Lecture Notes in Computer Science*, pp. 96–113. Springer, 2021. doi: 10.1007/978-3-030-81645-2_7. URL `https://doi.org/10.1007/978-3-030-81645-2_7`.

Jacson Rodrigues Correia da Silva, Rodrigo Ferreira Berriel, Claudine Badue, Alberto Ferreira de Souza, and Thiago Oliveira-Santos. Copycat CNN: stealing knowledge by persuading confession with random non-labeled data. In *2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8-13, 2018*, pp. 1–8. IEEE, 2018. doi: 10.1109/IJCNN.2018.8489592. URL `https://doi.org/10.1109/IJCNN.2018.8489592`.

David DeFazio and Arti Ramesh. Adversarial model extraction on graph neural networks. *CoRR*, abs/1912.07721, 2019. URL `http://arxiv.org/abs/1912.07721`.

Vasisht Duddu, Debasis Samanta, D. Vijay Rao, and Valentina Emilia Balas. Stealing neural networks via timing side channels. *CoRR*, abs/1812.11720, 2018. URL `http://arxiv.org/abs/1812.11720`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL `https://doi.org/10.1109/CVPR.2016.90`.

Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In Michael Bailey and Rachel Greenstadt (eds.), *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pp. 2669–2686. USENIX Association, 2021. URL `https://www.usenix.org/conference/usenixsecurity21/presentation/he-xinlei`.

Sanghyun Hong, Michael Davinroy, Yigitcan Kaya, Stuart Nevans Locke, Ian Rackow, Kevin Kulda, Dana Dachman-Soled, and Tudor Dumitras. Security analysis of deep neural networks operating in the presence of cache side-channel attacks. *CoRR*, abs/1810.03487, 2018. URL `http://arxiv.org/abs/1810.03487`.

Xing Hu, Ling Liang, Shuangchen Li, Lei Deng, Pengfei Zuo, Yu Ji, Xinfeng Xie, Yufei Ding, Chang Liu, Timothy Sherwood, and Yuan Xie. Deepsniffer: A DNN model extraction framework based on learning architectural hints. In *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, pp. 385–399. ACM, 2020. doi: 10.1145/3373376.3378460. URL `https://doi.org/10.1145/3373376.3378460`.

Weizhe Hua, Zhiru Zhang, and G. Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*, pp. 4:1–4:6. ACM, 2018. doi: 10.1145/3195970.3196105. URL `https://doi.org/10.1145/3195970.3196105`.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261–2269. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.243. URL `https://doi.org/10.1109/CVPR.2017.243`.

Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016. URL `http://arxiv.org/abs/1602.07360`.

Sanjay Kariyappa, Atul Prakash, and Moinuddin K. Qureshi. MAZE: data-free model stealing attack using zeroth-order gradient estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 13814–13823. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01360. URL `https://openaccess.thecvf.com/content/CVPR2021/html/Kariyappa_MAZE_Data-Free_Model_Stealing_Attack_Using_Zeroth-Order_Gradient_Estimation_CVPR_2021_paper.html`.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=Byl5NREFDr`.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106–1114, 2012. URL `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

Isabell Lederer, Rudolf Mayer, and Andreas Rauber. Identifying appropriate intellectual property protection mechanisms for machine learning models: A systematization of watermarking, fingerprinting, model access, and attacks. *CoRR*, abs/2304.11285, 2023. doi: 10.48550/arXiv.2304.11285. URL `https://doi.org/10.48550/arXiv.2304.11285`.

Pengcheng Li, Jinfeng Yi, and Lijun Zhang. Query-efficient black-box attack by active learning. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*, pp. 1200–1205. IEEE Computer Society, 2018. doi: 10.1109/ICDM.2018.00159. URL `https://doi.org/10.1109/ICDM.2018.00159`.

Zongjie Li, Chaozheng Wang, Pingchuan Ma, Chaowei Liu, Shuai Wang, Daoyuan Wu, and Cuiyun Gao. On the feasibility of specialized ability stealing for large language code models. *arXiv preprint arXiv:2303.03012*, 2023.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=Sys6GJqxl`.

Yuntao Liu and Ankur Srivastava. GANRED: gan-based reverse engineering of dnns via cache side-channel. In *CCSW'20, Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, Virtual Event, USA, November 9, 2020*, pp. 41–52. ACM, 2020. doi: 10.1145/3411495.3421356. URL `https://doi.org/10.1145/3411495.3421356`.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer V2: scaling up capacity and resolution. *CoRR*, abs/2111.09883, 2021. URL `https://arxiv.org/abs/2111.09883`.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, volume 11218 of *Lecture Notes in Computer Science*, pp. 122–138. Springer, 2018. doi: 10.1007/978-3-030-01264-9_8. URL https://doi.org/10.1007/978-3-030-01264-9_8.

Itay Mosafi, Eli (Omid) David, and Nathan S. Netanyahu. Stealing knowledge from protected deep neural networks using composite unlabeled data. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pp. 1–8. IEEE, 2019. doi: 10.1109/IJCNN.2019.8851798. URL https://doi.org/10.1109/IJCNN.2019.8851798.

Hoda Naghibijouybari, Ajaya Neupane, Zhiyun Qian, and Nael B. Abu-Ghazaleh. Rendered insecure: GPU side channel attacks are practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pp. 2139–2153. ACM, 2018. doi: 10.1145/3243734.3243831. URL https://doi.org/10.1145/3243734.3243831.

Daryna Oliynyk, Rudolf Mayer, and Andreas Rauber. I know what you trained last summer: A survey on stealing machine learning models and defences. *CoRR*, abs/2206.08451, 2022. doi: 10.48550/arXiv.2206.08451. URL https://doi.org/10.48550/arXiv.2206.08451.

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 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 4954–4963. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00509. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Orekondy_Knockoff_Nets_Stealing_Functionality_of_Black-Box_Models_CVPR_2019_paper.html.

Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish K. Shevade, and Vinod Ganapathy. Activethief: Model extraction using active learning and unannotated public data. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 865–872. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5432.

Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi (eds.), *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pp. 506–519. ACM, 2017. doi: 10.1145/3052973.3053009. URL https://doi.org/10.1145/3052973.3053009.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Kartik Patwari, Syed Mahbub Hafiz, Han Wang, Houman Homayoun, Zubair Shafiq, and Chen-Nee Chuah. DNN model architecture fingerprinting attack on CPU-GPU edge devices. In *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*, pp. 337–355. IEEE, 2022. doi: 10.1109/EuroSP53844.2022.00029. URL https://doi.org/10.1109/EuroSP53844.2022.00029.

Adnan Siraj Rakin, Md Hafizul Islam Chowdhuryy, Fan Yao, and Deliang Fan. Deepsteal: Advanced model extractions leveraging efficient weight stealing in memories. *CoRR*, abs/2111.04625, 2021. URL https://arxiv.org/abs/2111.04625.

Francesco Regazzoni, Paolo Palmieri, Fethulah Smailbegovic, Rosario Cammarota, and Ilia Polian. Protecting artificial intelligence ips: a survey of watermarking and fingerprinting for machine learning. *CAAI Trans. Intell. Technol.*, 6(2):180–191, 2021. doi: 10.1049/cit2.12029. URL `https://doi.org/10.1049/cit2.12029`.

Mauro Ribeiro, Katarina Grolinger, and Miriam A. M. Capretz. Mlaas: Machine learning as a service. In *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, pp. 896–902. IEEE, 2015. doi: 10.1109/ICMLA.2015.152. URL `https://doi.org/10.1109/ICMLA.2015.152`.

Nicholas Roberts, Vinay Uday Prabhu, and Matthew McAteer. Model weight theft with just noise inputs: The curious case of the petulant attacker. *CoRR*, abs/1912.08987, 2019. URL `http://arxiv.org/abs/1912.08987`.

David Rolnick and Konrad P. Kording. Reverse-engineering deep relu networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8178–8187. PMLR, 2020. URL `http://proceedings.mlr.press/v119/rolnick20a.html`.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00474. URL `http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html`.

Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. Model stealing attacks against inductive graph neural networks. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pp. 1175–1192. IEEE, 2022. doi: 10.1109/SP46214.2022.9833607. URL `https://doi.org/10.1109/SP46214.2022.9833607`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 1–9. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298594. URL `https://doi.org/10.1109/CVPR.2015.7298594`.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.308. URL `https://doi.org/10.1109/CVPR.2016.308`.

Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N. Asokan. DAWN: dynamic adversarial watermarking of neural networks. In *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, pp. 4417–4425. ACM, 2021. doi: 10.1145/3474085.3475591. URL `https://doi.org/10.1145/3474085.3475591`.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 2820–2828. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00293. URL `http://openaccess.thecvf.com/content_CVPR_2019/html/Tan_MnasNet_Platform-Aware_Neural_Architecture_Search_for_Mobile_CVPR_2019_paper.html`.

Junyi Wei, Yicheng Zhang, Zhe Zhou, Zhou Li, and Mohammad Abdullah Al Faruque. Leaky DNN: stealing deep-learning model secret with GPU context-switching side-channel. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020*, pp. 125–137. IEEE, 2020. doi: 10.1109/DSN48063.2020.00031. URL `https://doi.org/10.1109/DSN48063.2020.00031`.

Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. I know what you see: Power side-channel attack on convolutional neural network accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference, ACSAC 2018, San Juan, PR, USA, December 03-07, 2018*, pp. 393–406. ACM, 2018. doi: 10.1145/3274694.3274696. URL `https://doi.org/10.1145/3274694.3274696`.

Jonah O'Brien Weiss, Tiago A. O. Alves, and Sandip Kundu. Ezclone: Improving DNN model extraction attack via shape distillation from GPU execution profiles. *CoRR*, abs/2304.03388, 2023. doi: 10.48550/arXiv.2304.03388. URL `https://doi.org/10.48550/arXiv.2304.03388`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019. URL `http://arxiv.org/abs/1910.03771`.

Yoo-Seung Won, Soham Chatterjee, Dirmanto Jap, Shivam Bhasin, and Arindam Basu. Time to leak: Cross-device timing attack on edge deep learning accelerator. In *2021 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–4. IEEE, 2021.

Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Model extraction attacks on graph neural networks: Taxonomy and realisation. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako (eds.), *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, pp. 337–350. ACM, 2022. doi: 10.1145/3488932.3497753. URL `https://doi.org/10.1145/3488932.3497753`.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision, 2020.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5987–5995. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.634. URL `https://doi.org/10.1109/CVPR.2017.634`.

Mingfu Xue, Shichang Sun, Can He, Yushu Zhang, Jian Wang, and Weiqiang Liu. Activeguard: An active DNN IP protection technique via adversarial examples. *CoRR*, abs/2103.01527, 2021. URL `https://arxiv.org/abs/2103.01527`.

Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn DNN architectures. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pp. 2003–2020. USENIX Association, 2020. URL `https://www.usenix.org/conference/usenixsecurity20/presentation/yan`.

Kota Yoshida, Takaya Kubota, Shunsuke Okura, Mitsuru Shiozaki, and Takeshi Fujino. Model reverse-engineering attack using correlation power analysis against systolic array based neural network accelerator. In *IEEE International Symposium on Circuits and Systems, ISCAS 2020, Sevilla, Spain, October 10-21, 2020*, pp. 1–5. IEEE, 2020. doi: 10.1109/ISCAS45731.2020.9180580. URL `https://doi.org/10.1109/ISCAS45731.2020.9180580`.

Honggang Yu, Haocheng Ma, Kaichen Yang, Yiqiang Zhao, and Yier Jin. Deepem: Deep neural networks model recovery through EM side-channel information leakage. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020, San Jose, CA, USA, December 7-11, 2020*, pp. 209–218. IEEE, 2020. doi: 10.1109/HOST45689.2020.9300274. URL `https://doi.org/10.1109/HOST45689.2020.9300274`.

Xiaoyong Yuan, Leah Ding, Lan Zhang, Xiaolin Li, and Dapeng Oliver Wu. ES attack: Model stealing against deep neural networks without data hurdles. *IEEE Trans. Emerg. Top. Comput. Intell.*, 6(5):1258–1270, 2022. doi: 10.1109/TETCI.2022.3147508. URL `https://doi.org/10.1109/TETCI.2022.3147508`.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL `http://www.bmva.org/bmvc/2016/papers/paper087/index.html`.

Jingjing Zhao, Qingyue Hu, Gaoyang Liu, Xiaoqiang Ma, Fei Chen, and Mohammad Mehedi Hassan. AFA: adversarial fingerprinting authentication for deep neural networks. *Comput. Commun.*, 150:488–497, 2020. doi: 10.1016/j.comcom.2019.12.016. URL `https://doi.org/10.1016/j.comcom.2019.12.016`.

A Algorithms

Algorithm 2 Model Shortlisting Based on Inference Time

```
1: Set of models $\mathcal{M} = \{1, 2, \ldots, Z\}$
2: for $i = 1$ to $Z$ do
3:     Define $(MIN_i, MAX_i)$ for each model $i$
4: end for
5: Given a target model $X$ with inference time $T_X$
6: Initialize an empty set of selected models $S = \{\}$
7: for $i = 1$ to $Z$ do
8:     if $MIN_i \leq T_X \leq MAX_i$ then
9:         Add model $i$ to the set of selected models $S$
10:    end if
11: end for
12: return Set of selected models $S$
```