

Precise Extraction of Deep Learning Models via Side-Channel Attacks on Edge/Endpoint Devices

Younghan Lee¹, Sohee Jun¹, Yungi Cho¹, Woorim Han¹,
Hyungon Moon^{2*}, and Yunheung Paek^{1*}

¹ Seoul National University, Seoul, Republic of Korea

² UNIST, Ulsan, Republic of Korea

{201younghanlee, soheejun12, rimwoo98, ypaek}@snu.ac.kr,
ygcho@sor.snu.ac.kr, hyungon@unist.ac.kr

*: Correspondence should be addressed to H. Moon and Y. Paek.

Abstract. With growing popularity, deep learning (DL) models are becoming larger-scale, and only the companies with vast training datasets and immense computing power can manage their business serving such large models. Most of those DL models are proprietary to the companies who thus strive to keep their private models safe from the model extraction attack (MEA), whose aim is to steal the model by training surrogate models. Nowadays, companies are inclined to offload the models from central servers to edge/endpoint devices. As revealed in the latest studies, adversaries exploit this opportunity as new attack vectors to launch side-channel attack (SCA) on the device running victim model and obtain various pieces of the model information, such as the model architecture (MA) and image dimension (ID). Our work provides a comprehensive understanding of such a relationship for the first time and would benefit future MEA studies in both offensive and defensive sides in that they may learn which pieces of information exposed by SCA are more important than the others. Our analysis additionally reveals that by grasping the victim model information from SCA, MEA can get highly effective and successful even without any prior knowledge of the model. Finally, to evince the practicality of our analysis results, we empirically apply SCA, and subsequently, carry out MEA under realistic threat assumptions. The results show up to 5.8 times better performance than when the adversary has no model information about the victim model.

Keywords: Privacy in Deep Learning Models · Model Extraction Attack · Side-channel Attack

1 Introduction

Deep learning (DL) models empower many commercial applications and are potentially worth millions of dollars [12,3,19]. Until now, most model architectures and topology have been publicly available, but as models become larger-scale, the increased training cost and difficulty drive companies to prohibit the competitors from creating a copy and taking the market share. The cost of training a DL

model comes from both the computational resources and training datasets. Recent studies have shown that the *model extraction attack* (MEA), aiming to train a *surrogate model* of similar performance with much less training cost, is a real threat to such efforts of protecting valuable DL models [22,16,4,14,2,15,25]. Unfortunately, black-box MEAs require tremendous computational resources and time overhead [8]. To mitigate the amount of labor and increase the chances of success, they usually make certain unrealistic assumptions that give them pre-knowledge about the victim model. For instance, a typical assumption is that the surrogate model has the same or more complex model architecture and the same image dimension as the victim [22,14,15].

The growing demand for on-device ML services is fulfilled by offloading their models to edge/endpoint devices [11], which the adversary may access physically or via network connections, to improve response times and save bandwidth. All in all, this recent development in ML computing opened up a new opportunity that the adversary may exploit as attack vectors to wage *side-channel attacks* (SCAs) on the machine running the victim model. For example, when the victim model and the adversary's application run on the same device, the cache memory may be shared between them, which renders the model vulnerable to the cache SCA [24]. After gathering run-time information leaked via the device hardware, SCA can provide the adversary essential information about the victim that includes *model architecture* (MA) and *image dimension* (ID) of a DL model. Such information is essentially identical to the assumed prior knowledge necessitated for boosting black-box MEAs. This means that with the aid of SCA, MEA can still build a surrogate model better resembling the target even without unrealistic initial assumptions on the victim side. However, there is a stumbling block to the full utilization of SCA for MEA; SCA does not come for free but requires a great deal of cost and effort to obtain sufficient model information accurately [24,8]. To improve their work, SCAs usually need to make strong assumptions on their target systems, which could often be unrealistic or broken just by simple obfuscation techniques [27]. Therefore, a practical and efficient way to use SCA for MEA should be to extract only the essential information required to gain enough knowledge rather than prying into the target device to collect all sorts of information ignorantly. In order to pinpoint such essential information, we must understand how each piece of information affects the performance of MEA. Unfortunately, to the best of our knowledge, no studies have examined extensively the effects of different pieces in the collectable information on MEA. There are some preliminary studies demonstrating that MEA is robust to the difference in certain model features, such as MA, as long as the surrogate model's complexity is high enough [14,15]. Nonetheless, there has been no analytical report on the significance of other types of model information like ID in influencing the effectiveness of MEA.

Our work is the first to present an empirical analysis of the effects of SCA on MEA by evaluating the relationship between the performance of MEA and the model information supplied by SCA. We endeavor to empirically verify the relationship with various settings such as datasets, attack query budget, and

attack strategy. We delve into linking a particular type of model information with the outcomes of MEAs by investigating the correlation. We believe that our analytical report will give a glimpse of what types of model information are of more value to boost the performance of MEA relatively. Thus, it will enhance the efficacy of SCAs in their assistance to MEA by letting them concentrate on such valuable ones. In addition, we demonstrate the practicality of utilizing ID obtained from SCA to boost MEA by carrying out the experiment under realistic assumptions. The results achieve up to 5.8x much higher accuracy and fidelity than the adversary without any prior knowledge. Consequently, we argue that our work paves the way for future (offensive and defensive) research in DL model privacy by providing organized knowledge of correlation between existing MEAs with the SCA-supplied knowledge about the victim model. The following summarizes our contributions:

- We analyze the effect of model information exposed by SCA on MEA with different settings: datasets, query budget, and attack strategy.
- We demonstrate how accurately ID of DL models can be estimated by SCA and perform subsequent MEA with estimated ID under realistic threat assumptions to evince the practicality of MEA with SCA.
- We provide an informative insight into improving the defense against MEA allied with SCA by identifying what parts of model information are to be obfuscated from SCA for maximum defense with minimum effort.

2 Background

While MEA and SCA ultimately share the same goal of extracting the victim model of high value, their interpretations of a successful attack are different. MEA aims to obtain a replica of its victim model by copying the functionality. In contrast, SCA intends to extract the structural or architectural model information, including dimensions of layers and their topology.

MEA strives to create at low cost a surrogate of a high-performance victim DL model trained with a dataset of both high quality and large quantity. The most popular method relies on only querying and collecting the inference results from the victim model. The surrogate model is trained with the data used to query the victim with the classification result as its label. As the cost of extracting the model increases with querying more samples, recent studies focus mainly on selecting more valuable samples to reduce the query budget. Ideally, MEA must be carried out with a pure black-box setting where no model information are initially exposed to the adversary. However, in reality, to reduce the amount of labor and increase the chances of their success, many MEA techniques are evaluated under certain assumptions that they already have prior knowledge (ID and MA) about the victim model. We find that such assumptions are often unrealistic in practice as some knowledge can not be available to adversaries in a black-box setting.

SCA has long been studied by cyber security researchers for many decades. In recent years, the use of SCA has been broadened to extract the architectural

information of valuable DL models. Although SCA requires the assumption that adversaries gain access to hardware resources in the machine running the victim models, this assumption is deemed plausible these days, as discussed earlier. SCA attempts to collect model information by exploiting vulnerabilities in the underlying hardware. Previous studies have shown that SCA can collect mainly two types of model information, ID and MA. We note here that these are the same pieces of information somehow given to the adversary by assumptions. Further details of SCA is explained in Section 5.

3 Related Work

3.1 Model Extraction Attack

KnockoffNets [14] is one of the early MEA technique which is designed under black-box setting. The query samples are selected randomly from out-of-distribution attack dataset as the adversaries are unaware of the training dataset used by the victim model. Finally, the output label from the victim model is paired with the query data (i.e., *re-labeled image*) and used as training dataset for the surrogate model which will exhibit a similar functionality as the victim.

ActiveThief [15] proposed another method called *uncertainty* which is based on the confidence vector of the query samples. The intuition behind this approach is that to extract the classification functionality of the victim model, it is beneficial to concentrate on the query samples that will lie near the decision boundary. By doing so, the surrogate can be trained much more quickly with fewer query samples to reach the victim’s classification accuracy.

KnockoffNets and *ActiveThief* discuss briefly how the knowledge of MA affects their performance, and conclude that MEAs are relatively robust to (or regardless of) the choice of MA for their surrogates if the MA complexity is high enough. In other words, MEA can achieve good performance as long as the complexity of a surrogate is sufficiently high (usually, higher than the victim model). This implies that if SCA reveals the exact complexity of MA, the adversary can set up the surrogate with MA of optimal complexity to maximize the effectiveness of MEA. Regarding ID, *KnockoffNets* and *ActiveThief* are evaluated under a strong assumption that the adversary is aware of ID of the victim and thus can set up the surrogate with the same ID. They ignorantly assume that this essential pre-knowledge could be offered to attackers by convention, and none of them show how the attackers obtain such information. In this paper, we argue that this assumption can be fulfilled by employing SCA to decide the victim model ID in a deterministic manner and will empirically prove that it is indeed indispensable to the success of MEA.

3.2 Side-channel Attack

Cache Telepathy [24] utilized cache side-channel attack in estimating the architectures in MLaaS (Amazon SageMaker [1], Google ML Engine [5]) platforms.

By inferring the size of each layer’s input matrix, they deduce the ID of the model. Also, they can infer the MA by identifying the topology of layers. To evaluate the accuracy of their method, the extracted structure is compared to the model’s original structure. However, they do not show how useful their extracted model information is to boost MEA, which is necessary to understand the individual effect of each piece of information on black-box MEA.

4 Analysis of the Effects of Model Information

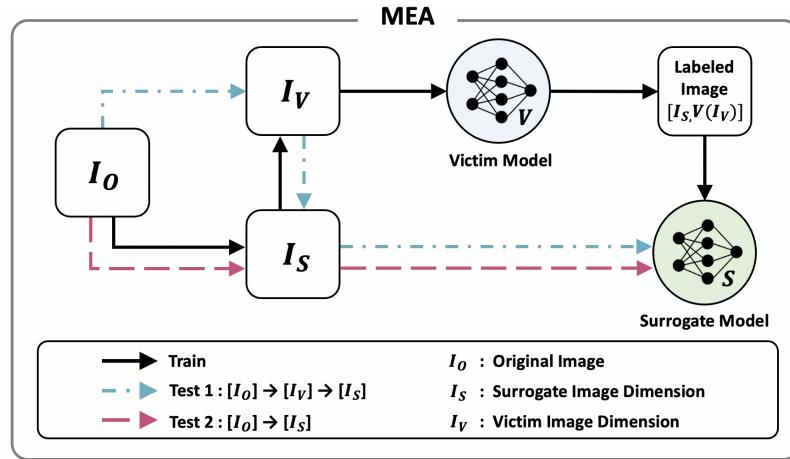


Fig. 1: Flowchart of Training and Evaluating Surrogate Models through MEA

As described just above, it is evident that an attacker can acquire by SCA the model information (i.e., ID, MA) pivotal to MEA which otherwise would have to rely on somewhat ungrounded assumptions to attain its goal. In this work, we conduct a comprehensive analysis to understand the effects of SCA on MEA. For this, we evaluate the performance results of MEA for various configurations of ID and MA, and find the following relationships. **R1:** Effectiveness of MEA vs. model information (ID and MA) of the victim. **R2:** **R1** vs. analysis settings (a : datasets, b : attack query budget, c : attack strategy).

4.1 Training & Evaluation

In this subsection, we elaborate Fig. 1 in further detail, in which the process of MEA is illustrated by training and evaluating surrogate models.

Surrogate Model Training. By employing MEA, we trained the surrogate model following the flow depicted as the solid black line in Fig. 1. We first resize the original image I_O (ID of original image) to I_S (ID of the surrogate model)

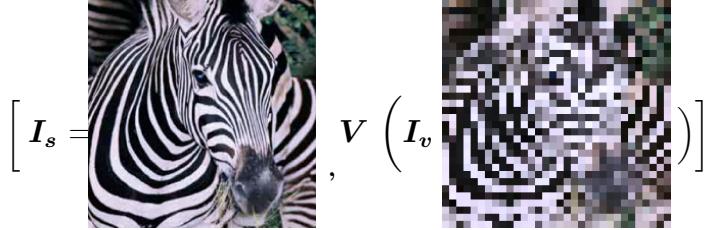


Fig. 2: Example of Re-labeled Image Example (Surrogate [224], Victim [32])

and resize once more to I_V (ID of the victim model). This is a realistic attack scenario as the adversary is unaware of the victim’s image dimension and is likely to query the victim with I_S . The confidence score that the victim returns is $V(I_V)$ and the surrogate model is trained with the *re-labeled image* with the new label, $[I_S, V(I_V)]$. A discrepancy in I_S and I_V causes the *label mismatch*, which is a difference in the classification score of I_S and I_V given by the victim model. Fig. 2 visually illustrates the *re-labeled image* where the new label of I_S is given by the classification score of the victim with lower ID (I_V).

Surrogate Model Evaluation. To measure the effectiveness of MEA, we evaluated the surrogate model with two separate tests as shown in Fig. 1. Test 1 resizes I_O to I_V before eventually arriving at I_S , and test 2 resizes I_O directly to I_S . Test 1 is similar to the current method of evaluation where model information including ID (i.e., I_V) is known to adversaries. Test 2 can be considered as a more realistic way of measuring the accuracy of the surrogate model as ID of the victim is unknown to the adversary in reality. We note that when ID of the surrogate is the same as that of the victim, there is no difference between test 1 and test 2 as $I_S = I_V$. The result tables (Table 2, Table 3, Table 4, Table 5, Table 7) include both tests, and the reported surrogate accuracy denotes the best accuracy among the ones measured every five epochs during the training.

4.2 Analysis settings

Image Dimension (ID). To understand the relationship between IDs and the MEA effectiveness, we optimize the models with various IDs to achieve the best accuracy for the victim models. We choose ResNet-50 [7] (Additional result for VGG16 [21] in Appendix A.2) as the architectures of both the victim and surrogate models. The ID is represented as a subscript (i.e., $RN50_{[64]}$ represents ResNet-50 model optimized for 64x64x3 images).

Datasets. For ID analysis, the victim models are trained with three widely used datasets, as shown in Table 1. To achieve a realistic and high-accuracy model, they are trained by a transfer learning method with a pre-trained model by ISLVRC-2012 (ImageNet) dataset. The accuracy of the trained victim models is shown in the second column of Table 2. For the attack query dataset, we follow the common assumption that an adversary is unaware of the dataset used for training the victim model. The analysis is performed with *out-of-distribution*

Table 1: Dataset Configuration

	Dataset	Classes	Train Samples	Test Samples	Original Image (I_O)	Analysis
Victim	Indoor[18]	67	14,280	1,340	224x224x3	ID& MA
	Caltech-256[6]	256	23,380	6,400	224x224x3	ID& MA
	CUB-200[23]	200	5,994	5,794	224x224x3	ID
	CIFAR-100[9]	100	50,000	10,000	32x32x3	MA
Attack	ImageNet[20]	1,000	1.2M	150,000	224x224x3	ID& MA
	OpenImages[10]	600	1.74M	125,436	224x224x3	ID

datasets, ImageNet and OpenImages, with enough samples for a large query budget analysis. We note that OpenImages is an unbalanced dataset where there are uneven number of samples for each image class. For MA analysis, along with three forementioned datasets, we added CIFAR-100 as shown in Table 1 and only ImageNet is used as an attack query dataset. The accuracy of the victim models is shown in the second column of Table 4 and Table 5.

Attack Query Budget. Various attack query budgets are used for ID analysis (i.e., 30k, 60k, and 90k). It is designed to verify if the relationship between ID of the victim and the effectiveness of MEA changes over different query budgets.

Attack Strategy. For ID analysis, two attack strategies are implemented. We use a random sampling strategy based on KnockoffNets and *uncertainty* method from ActiveThief because they exhibit arguably the best performance in MEA. For MA analysis, KnockoffNets was implemented.

Model Architecture (MA). to understand the relationship between MA and the effectiveness of MEA, various MAs of different complexities are used to train the victim and surrogate models: WideResNet-28-k [26] with different k values, VGG16, VGG19, ResNet-50 and ResNet-101. (Details in Appendix A.1).

4.3 Effect of Image Dimension (ID)

Model Extraction Attack Result. Table 2 depicts the result of MEA with the relative accuracy (i.e., accuracy of surrogate model relative to that of the victim which is 1x) as the effectiveness metric. The grey colored boxes denote that IDs of the victim and the surrogate are the same. The bold type represents the best accuracy among surrogate models of different IDs. The result confirms that matching the ID of the victim and surrogate is vital to maximize the efficacy of MEA. Among the total of 48 cases in the grey colored boxes per metric, 92% (44/48) achieve the best accuracy. We find only few cases in the upper right triangular matrix where the surrogate with a higher ID achieved the same or better performance which is at most 3% higher. However, when ID is different, the accuracy of surrogate trained with higher ID ($I_S > I_V$) drop by 0.72x at worst case (average drop of 0.24x). The decrease is more significant when the surrogate is trained with smaller ID ($I_S < I_V$), showing 0.73x in accuracy at worst case (average drop of 0.43x). Between the effectiveness measured by test 1 and test 2 (excluding the cases where $I_s = I_V$), the result from test 1 is

higher for 71% (51/72) of the total cases. This instance can be explained by *label mismatch* which is caused by the fact that the new label of I_S is given by I_V . When the relative accuracy is measured by test 1, the influence of *label mismatch* is diminished as I_S is transformed from I_V unlike test 2 in which I_S is transformed from I_O directly. In short, the effectiveness of MEA is maximized when the surrogate model's ID matches the victim model's. (Ablation study results in Appendix A.2)

Datasets. We examine if the trend described above is consistent throughout various datasets. Three different datasets are used for victim models and two attack query datasets for training the surrogate model. Table 2 shows MEA is most effective when the surrogate's ID is identical for all three victim model datasets which have different number of classes. The effectiveness is generally higher with ImageNet which achieve higher relative accuracy in 68% (65/96) of total cases with the average rise of 1.4%. This phenomenon is due to the fact that the victim models are pre-trained with ImageNet dataset. However, it is important to note that the trend continues for both attack query datasets (i.e., Imagenet and OpenImages).

Attack Query Budget. Fig. 3 illustrates changes in the effectiveness of MEA over various query budgets (30k, 60k, and 90k). The surrogate model that matches the victim's ID is marked with a red star marker at each query budget. The result shows that matching the surrogate's ID to the victim model is always beneficial through various query budgets. Also, we note that even with a much less query budget, a higher accuracy can be attained when ID is matched. In some cases, query-budget-30k with the same ID can achieve a better accuracy than query-budget-90k with a different ID. Moreover, as the cost of MEA increases as the query budget increases, the adversary can save a huge amount of cost just by training the surrogate with the same ID as the victim.

Attack Strategy. We implement ActiveThief to verify if using a different attack strategy consorts with the phenomena observed in the previous analysis. The attack is carried out with 2k initial seed samples and by sampling new 2k samples for 9 additional rounds (i.e., query-budget-20k). Table 3 shows the similar result that the surrogates trained with the identical ID achieve the best accuracy.

4.4 Effect of Model Architecture (MA)

Model Extraction Attack Result. In order to investigate how MA information of the victim model affects the effectiveness of MEA, we design the analysis with various MA of different complexities. We note that in order to eliminate any effects of ID, all models are set to have the same ID. Therefore, the results shown in Table 4 and Table 5 are from both test 1 and 2. Unlike ID analysis, Table 4 shows that MA of higher complexity achieves better accuracy than MA of the same complexity. Also, Table 5 reveals a similar results as previous studies mentioned in Section 3.1. While the adversary can benefit from knowing the same model architecture, the effect is relatively less significant for most

Table 2: ID Analysis (Datasets). Effectiveness (Relative Accuracy) of MEA (KnockoffNets) with query-budget-60k

Dataset	a Accuracy	Model	Attack Query	Surrogate Model					
				RN50 _[32]	RN50 _[64]	RN50 _[128]	RN50 _[224]		
				Test 1	Test 2	Test 1	Test 2	Test 1	Test 2
Indoor67	64.78% (1x)	RN50 _[32]	ImageNet	0.88x 0.88x	0.63x 0.91x	0.59x 0.50x	0.43x 0.16x		
			OpenImages	0.91x 0.91x	0.69x 0.91x	0.62x 0.44x	0.46x 0.17x		
Caltech-256	66.56% (1x)		ImageNet	0.96x 0.96x	0.78x 0.97x	0.75x 0.61x	0.59x 0.28x		
			OpenImages	0.94x 0.94x	0.75x 0.95x	0.66x 0.53x	0.47x 0.23x		
CUB-200	67.02% (1x)	RN50 _[32]	ImageNet	0.86x 0.86x	0.62x 0.80x	0.51x 0.40x	0.35x 0.15x		
			OpenImages	0.83x 0.83x	0.56x 0.73x	0.48x 0.35x	0.31x 0.14x		
Indoor67	72.99% (1x)	RN50 _[64]	ImageNet	0.33x 0.28x	0.94x 0.94x	0.77x 0.87x	0.69x 0.49x		
			OpenImages	0.35x 0.29x	0.96x 0.96x	0.85x 0.91x	0.71x 0.53x		
Caltech-256	76.81% (1x)		ImageNet	0.51x 0.48x	0.99x 0.99x	0.90x 0.96x	0.85x 0.72x		
			OpenImages	0.48x 0.45x	0.97x 0.97x	0.87x 0.94x	0.78x 0.69x		
CUB-200	77.89% (1x)	RN50 _[64]	ImageNet	0.15x 0.13x	0.88x 0.88x	0.66x 0.79x	0.58x 0.40x		
			OpenImages	0.13x 0.11x	0.82x 0.82x	0.65x 0.76x	0.55x 0.37x		
Indoor67	67.24% (1x)	RN50 _[128]	ImageNet	0.33x 0.22x	0.82x 0.78x	0.97x 0.97x	0.95x 0.94x		
			OpenImages	0.33x 0.22x	0.84x 0.80x	1.00x 1.00x	0.96x 0.96x		
Caltech-256	76.75% (1x)		ImageNet	0.44x 0.43x	0.78x 0.75x	0.99x 0.99x	0.97x 0.97x		
			OpenImages	0.43x 0.42x	0.76x 0.73x	0.97x 0.97x	0.95x 0.98x		
CUB-200	77.44% (1x)	RN50 _[128]	ImageNet	0.21x 0.15x	0.64x 0.59x	0.91x 0.91x	0.86x 0.87x		
			OpenImages	0.18x 0.13x	0.60x 0.56x	0.88x 0.88x	0.83x 0.84x		
Indoor67	73.51% (1x)	RN50 _[224]	ImageNet	0.26x 0.25x	0.66x 0.67x	0.90x 0.87x	0.92x 0.92x		
			OpenImages	0.26x 0.23x	0.69x 0.69x	0.92x 0.90x	0.97x 0.97x		
Caltech-256	78.11% (1x)		ImageNet	0.36x 0.39x	0.78x 0.75x	0.95x 0.92x	1.00x 1.00x		
			OpenImages	0.34x 0.38x	0.74x 0.73x	0.92x 0.90x	0.99x 0.99x		
CUB-200	78.17% (1x)	RN50 _[224]	ImageNet	0.17x 0.16x	0.53x 0.52x	0.78x 0.78x	0.89x 0.89x		
			OpenImages	0.15x 0.14x	0.48x 0.45x	0.74x 0.71x	0.85x 0.85x		

cases. Attacking with $RN101_{[224]}$ achieves equally high or higher relative accuracy compared to attacking with the same model architecture. We conclude that the effect of MA on MEA becomes insignificant as long as the surrogate model's MA occupies a high complexity.

5 Experiments

Our analysis, as explained in the previous section (Section 4), suggests that an adversary knowing model information (i.e., ID) can boost the effectiveness of MEA. To estimate such information, an adversary can exploit SCA on local devices on which the victim model is running. In this section, we demonstrate end-to-end MEA with SCA *without any prior knowledge* and confirm that our attack mechanism is realistic and highly effective to achieve virtually the same ideal performance of MEA exhibited by the previous work assuming that they somehow manage to obtain model information before actual attacks.

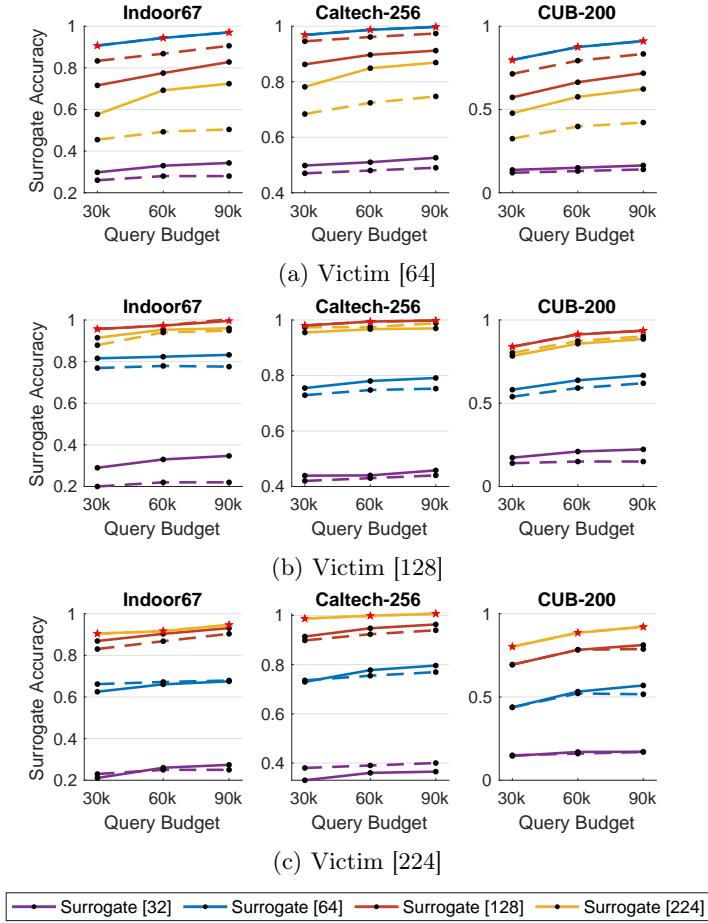


Fig. 3: ID Analysis (Attack Query Budget). Effectiveness (Relative Accuracy) of MEA (KnockoffNets with ImageNet) for test 1 (solid line) & test 2 (dotted line)

5.1 Experimental Setups for MEA with SCA

Overview Our SCA infers ID from the Generalized Matrix Multiply (GEMM), a commonly used building block of DL model implementation, operating in repeated loops for the matrix multiplication. If our SCA infers the number of iterations executed in each layer, it can compute the size of the layer’s input matrix by multiplying the number with known constants. The target of our SCA, ID, can be inferred in this way because ID is equivalent to the first layer’s input matrix size. Our implementation of SCA is based on *Cache Telepathy* [24] with modifications tailored for our purposes. Fig. 4 illustrates the process of MEA with SCA. Firstly, we generate a dynamic call graph (DCG) that reflects the execution flow is generated by monitoring GEMM library with noise filtering

Table 3: ID Analysis (Attack Strategy). The Effectiveness (Relative Accuracy) of MEA (ActiveThief with ImageNet) with query-budget-20k

Victim Model			Surrogate Model									
Dataset	Accuracy	Model	RN50 _[32]		RN50 _[64]		RN50 _[128]		RN50 _[224]			
			Test 1	Test 2	Test 1	Test 2	Test 1	Test 2	Test 1	Test 2		
Indoor67	64.78% (1x)	RN50 _[32]	0.82x	0.82x	0.34x	0.30x	0.48x	0.44x	0.46x	0.16x		
	72.99% (1x)	RN50 _[64]	0.31x	0.27x	0.90x	0.90x	0.70x	0.86x	0.65x	0.50x		
	67.24% (1x)	RN50 _[128]	0.28x	0.21x	0.78x	0.75x	0.95x	0.95x	0.90x	0.91x		
	73.51% (1x)	RN50 _[224]	0.17x	0.23x	0.60x	0.65x	0.85x	0.84x	0.88x	0.88x		

Table 4: MA Analysis 1. The Effectiveness (Relative Accuracy) of MEA (Knock-offNets with ImageNet) with query-budget-20k

Victim Model			Surrogate Model		
Dataset	Accuracy	Model	WRN28-1 _[32]	WRN28-5 _[32]	WRN28-10 _[32]
CIFAR-100	68.36% (1x)	WRN28-1 _[32]	0.43x	0.56x	0.57x
	77.95% (1x)	WRN28-5 _[32]	0.26x	0.36x	0.39x
	79.44% (1x)	WRN28-10 _[32]	0.26x	0.37x	0.39x

Table 5: MA Analysis 2. The Effectiveness (Relative Accuracy) of MEA (Knock-offNets with ImageNet) with query-budget-20k

Victim Model			Surrogate Model			
Dataset	Accuracy	Model	VGG16 _[224]	VGG19 _[224]	RN50 _[224]	RN101 _[224]
Indoor67	78.20% (1x)	VGG16 _[224]	0.88x	0.86x	0.86x	0.88x
Caltech-256	83.06% (1x)	VGG16 _[224]	0.94x	0.92x	0.93x	0.94x
Indoor67	78.13% (1x)	VGG19 _[224]	0.83x	0.90x	0.87x	0.90x
Caltech-256	85.77% (1x)	VGG19 _[224]	0.86x	0.93x	0.92x	0.94x

mechanism. From the DCG, we can infer the number of iterations of loops executed in each layer. Secondly, ID of DL model is estimated through the inverse calculation from the properties of the loops. Finally, estimated ID is used for subsequent MEA. Each step is described in more detail below.

Threat Assumptions We performed SCA and MEA assuming that DL model is a black-box and running on an edge/endpoint device. The adversary is not given direct access to the victim model, but only the prediction result is available. For cache-timing attack as a part of SCA, the tracing process by the adversary is running on the same processor as the victim model’s process to capture the addresses which are managed by a process running in the back-

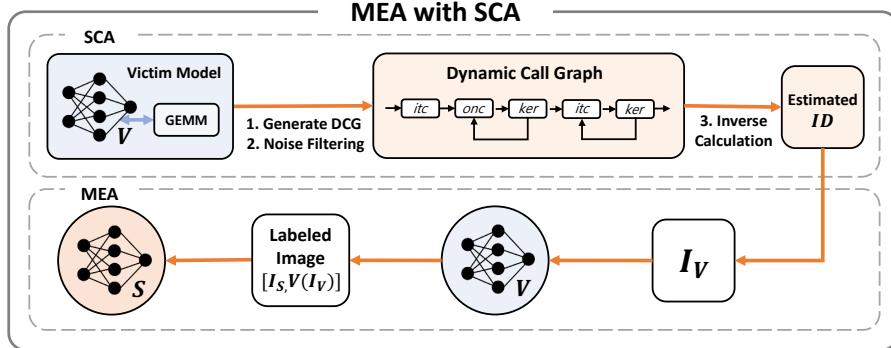


Fig. 4: Flowchart of MEA with SCA. 1) DCG Generation, 2) Noise Filtering for DCG, 3) Inverse Calculation to estimate ID, 4) MEA with estimated ID

ground. Such scenario is realistic as the DL model is off-loaded to the local device. Also, the adversary is capable of analyzing linear algebra library used in the DL model such as OpenBLAS [28] which is open-source. We assume that stride and padding are known to the adversary. For SCA experiment, we used OpenBLAS for GEMM library with a Linux machine running on a i7-9700 processor that has 8 cores, 64KB of L1 cache, 256KB of L2 cache, and 12MB of shared last-level cache. The system runs with 64GB of main memory.

Dynamic Call Graph Generation For the purpose of inferring the number of loop iterations, we used cache-timing attack as a part of SCA to infer the DCG that is composed of calls to three key functions, *itcopy*, *oncopy* and *kernel*. We chose these as key functions because each loop of our interest for generating DCG is composed of a specific sequence of such functions as shown below.

- L1: *itcopy*-*oncopy*-*kernel*-*itcopy*-*kernel*
- L2: *oncopy*-*kernel*
- L3: *itcopy*-*kernel*

While the victim model is running, the adversary monitors the calls to these three key functions using *Flush+Reload*, a common technique used for such purpose. By constantly monitoring the addresses of the key functions, we determine if one of the addresses has recently been accessed by measuring the access delay (i.e., cache hit or miss). Three properties of each loop can be deduced from DCG for the following procedure of MEA with SCA. 1) the number of iterations of loops (N), 2) short execution time (ST), 3) average execution time (AT). Short execution time is measured by the last function call and the average execution time is measured with all other calls. Further detail about the algorithm of DCG generation is described in Appendix A.3.

Noise Filtering Mechanism We found that the DCG generation suffers from the inherent noise that SCA is prone to. We devised a noise filtering mechanism that is tailored for inferring a precise enough DCG exploiting the characteristics of GEMM computation in the target DL architecture. While running,

the execution time of each loop iteration is similar to each other except for the last one because each iteration processes inputs of similar size. For this reason, the interval between the function calls that we are monitoring is supposed to be similar to each other. Based on this, we filtered out duplicate observations of one function calls in two ways. First, we filtered out the function calls observed shortly after (< 10 time intervals) the previous one, considering that the two adjacent observation is from a single function call. Second, we used the average interval between the function calls as a threshold and considered any calls to `itcopy` within the threshold as noise.

Inverse Calculation Algorithm In GEMM matrix multiplication (i.e., m by k and k by n), m , k , n represent input matrix, weight matrix, and output depth respectively and they are divided into loops by constant Q , P , and UNROLL which are pre-defined by the GEMM library. Therefore, after analyzing DCG to deduce three properties of each loop described above, we estimate ID of DL models by the inverse calculation. Firstly, m value was calculated from properties of $L2$. As m is divided by P to form $L2$ except for the last two iterations, which is operated by an unit of $(P + m \bmod P)/2$, m can be obtained by the multiplication of P with the number of iterations of $L2$. Secondly, n is divided by $3 \cdot \text{UNROLL}$ to form $L3$ and depending on the AT of $L3$, the last iteration is operated with either $3 \cdot \text{UNROLL}$ or UNROLL . Therefore, we inversely calculate n by the multiplication of UNROLL with the number of iterations of $L3$. Thirdly, if the number of $L1$ is less than two, we need to estimate the average execution time of $L1$ by matrix multiplication. The algorithm can be found in Appendix A.3. Once we obtained all properties of $L1$, the value of k is estimated in a similar fashion as above. Finally, with all estimated values, ID of DL model is estimated as shown in Algorithm 1.

Algorithm 1: Estimate Input Dimension

Input: P , Q , UNROLL , stride , padding , N (Number of iterations of loop),
 ST (Short execution time of loop), AT (Average execution time of loop)

Output: ID : Input Dimension

```

1  $m \leftarrow ((N_{L2}-2)+2*(ST_{L2}/AT_{L2}))*P$                                 ▷ Find  $m$ 
2 if  $ST_{L3} < (AT_{L3}/2)$  then                                                 ▷ Find  $n$ 
3    $n \leftarrow (N_{L3}-1)*3\text{UNROLL}+\text{UNROLL}$ 
4 else
5    $n \leftarrow N_{L3}*3\text{UNROLL}$ 
6 if  $N_{L1} < 2$  then                                                       ▷ Find  $k$ 
7    $AT_{L1} \leftarrow \text{EstimateL1}(m, n, Q)$ 
8    $k \leftarrow (ST_{L1}/AT_{L1})*Q$ 
9 else
10   $k \leftarrow ((N_{L1}-2)+2*(ST_{L1}/AT_{L1}))*Q$ 
11  $kernel \leftarrow \text{sqrt}(k/3)$ 
12  $ID \leftarrow (\text{sqrt}(m)+(kernel-1)-2*\text{padding})*\text{stride}$ 
13 return  $ID$ 
```

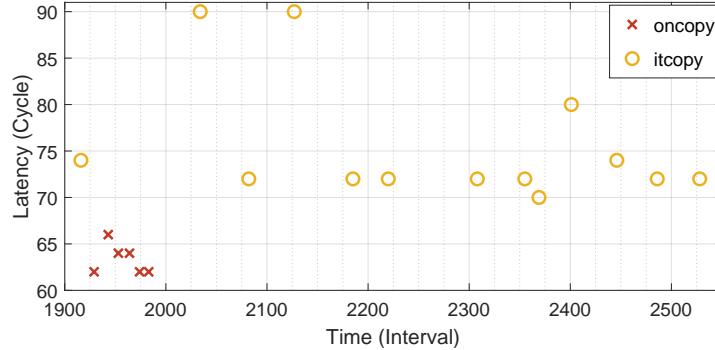
Fig. 5: DCG Generation Result for $RN50_{[128]}$ Victim Model

Table 6: Image Dimension Estimation Result. Values in SCA and target columns represent the estimated and actual values respectively

Victim Model	m		n		k		$kernel$		ID	
	SCA	Target	SCA	Target	SCA	Target	SCA	Target	SCA	Target
$RN50_{[128]}$	4118.5	4096	72	64	35.7	27	3.5	3	129.3	128

5.2 SCA Results

Dynamic Call Graph Generation Fig. 5 shows the result of DCG generation after the noise filtering mechanism which only took less than 0.016 seconds to construct. From this result, three properties described above are calculated. We note that the observations about *kernel* is omitted for brevity. Firstly, the number of each loop iteration (N_L) is calculated by counting the number of *oncopy* and *itcopy* function calls for $L2$ and $L3$ respectively. There are six *oncopy* calls between 1,900 and 2,000 time intervals and 12 *itcopy* calls after 2,000 intervals and 1 *itcopy* at the beginning. From these, N_{L2} and N_{L3} are estimated as 13 and 6 respectively. N_{L1} is estimated to be 1 as no calls to *oncopy* has been observed after the last *itcopy*. Also, the average and short execution time of each function call can also be estimated from the result by comparing the time intervals as described in Section 5.1. All properties collected from DCG for different victim models can be found in Appendix A.3.

Image Dimension Estimation In our demonstration, pre-defined constant values of P , Q , and UNROLL are 320, 320, and 104,512 respectively. Each of m , n , and k values are inversely calculated according to Algorithm 1 and *kernel* is calculated by taking the square root of $(k/3)$ (i.e., RGB channels). Finally, we round the calculated output to the nearest whole number to estimate ID of the victim model. Image dimension estimation result of different victim models are illustrated in Table 6. We see that even with some discrepancies in estimated values of m , n , and k , ID of the victim models was retrieved accurately with only ID of $RN50_{[128]}$ being off only by 1.

Table 7: Subsequent MEA (with ImageNet) result. $RN50_{[129]}$ (with estimated ID) shows slightly worse performance than $RN50_{[128]}$

Victim Model			Surrogate Model				
Dataset	Accuracy	Model	$RN50_{[32]}$	$RN50_{[64]}$	$RN50_{[128]}$	$RN50_{[129]}$	$RN50_{[224]}$
Indoor67	67.24% (1x)		0.22x	0.78x	0.97x	0.99x	0.94x
Caltech-256	76.75% (1x)	$RN50_{[128]}$	0.43x	0.75x	0.99x	0.96x	0.97x
CUB-200	77.44% (1x)		0.15x	0.59x	0.91x	0.87x	0.87x

5.3 MEA with Model Information from SCA

We performed subsequent MEA with estimated ID from SCA to demonstrate the benefit of using the estimated ID in MEA. We only carried out MEA only for $RN50_{[128]}$ as the victim with $RN50_{[129]}$ as a surrogate model for the worst case experiment (More results in Appendix A.3). The result in Table 7 illustrates that $RN50_{[129]}$ (with estimated ID) achieves a slightly worse performance as ID is not the exact match. However, such performance is still better than most of MEAs with different IDs as shown in Section 4.3 because estimated ID was relatively closer to I_V . At most, the adversary can achieve up to 5.8 times better relative accuracy than the worst case $RN50_{[32]}$. This result backs the hypothesis that employing SCA to collect model information such as ID of the model helps boost the performance of MEA.

6 Discussion and Limitations

Our work targets the present and future computing environments where DL models are not only running on central servers or public cloud, but also off-loaded to diverse classes of edge/endpoint devices. In these environments occur *device fragmentation* referring to when users are running many different versions of software and hardware platforms. Clearly, DL models running on such different platforms may also be diversified (i.e., various ID) for efficiency or portability. Our proposed method is applicable regardless of MA given environments where adversaries can pry on edge/endpoint devices. Moreover, with a recent development in computing power, ResNet-50 can be deployed in edge devices [17].

Offensive Side. Traditional MEAs armed with prior model knowledge based purely on postulations may find more challenging time to steal information from such diversified DL models. Through extensive empirical analysis, we prove that a key to the success of MEA is to have the exact information about ID and MA of the victim model, and also demonstrated that SCAs have enough capability of extracting ID and MA accurately. Consequently, we suggest that for better probability of success, future researchers of MEA first should attempt to obtain the model information from SCA, instead of relying on a prior assumption.

Defensive Side. In our work, we quantitatively show that among model information gathered by SCA, ID is the most essential to MEAs. This implies

to defenders against MEA as well as SCA that they do not have to exhaust themselves to protect every information about their DL models, but should focus mainly on concealing or obfuscating the ID value from the adversary. For instance, to fight against SCA, they may obfuscate GEMM operations to hide actual cache access patterns. For example, dummy matrix operations may be added to the original model by inserting dummy columns and rows in the first layer of a DL model to increase the number of loop iterations. Such obfuscated operations would misinform the adversary of the ID value, which in turn eventually hampers the performance of MEA.

Limitations. Our study can be further improved by expanding the training datasets and architectures of victim models and utilizing newly published MEAs can revamp the analysis. In regards to SCA, due to the nature of cache-timing attack, the outcome of SCA can be obscure with noise from CPU. Such limitation may lead to repetition or failure of the attack. Therefore, devising and applying a noise filtering mechanism appropriate for the target execution environment is required in MEA with SCA to maximize the performance of MEA without prior information.

7 Conclusion

Our systematic analysis has shown that ID and MA are two crucial pieces of model information as the initial knowledge about the victim for MEA. Our demonstration confirmed that MEA achieves the best performance when training the surrogate model with ID identical to that of the victim model, and MA more complex than the victim's. This result was consistent across various analysis settings. Our findings account for the reasoning behind the common design decision of existing MEA techniques that prefers their surrogates to have the identical IDs as the victims and as complex MAs as possible. We note that this assumption will become unrealistic for MEAs aiming at future DL models diversified to run on varied classes of computing devices because the models will have many different IDs and MAs depending on their devices. However, an adversary may use advanced SCA techniques by exploiting vulnerabilities in hardware to provide fairly accurate ID and MA of the victim model and achieve satisfying outcomes even without such an unrealistic assumption, as demonstrated in this paper. According to our empirical study, SCA can provide the estimated values for ID of DL models that are extremely close to the target value, thereby helping the subsequent MEA to achieve the idealistic performance. From our result, defenders fighting against MEA allied with SCA may learn a lesson that they can most effectively thwart MEA by obfuscating the ID values of their DL models.

8 Acknowledgements

This work was supported by the BK21 FOUR program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2022 and

the National Research Foundation of Korea (NRF) grant funded by Korea government (MSIT) (NRF-2020R1A2B5B03095204) & (NRF-2022R1F1A1076100) and by Inter-University Semiconductor Research Center (ISRC). Also, it was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-01840, Analysis on technique of accessing and acquiring user data in smartphone) & (No.2021-0-01817, Development of Next-Generation Computing Techniques for Hyper-Composable Datacenters) & (No.2021-0-00724, RISC-V based Secure CPU Architecture Design for Embedded System Malware Detection and Response)

A Appendix

A.1 Model architectures

Table 8 summarises the details of model architectures used for experiments.

Table 8: Details of model architecture used for victim and surrogate models

	RN50 ₁₂₁	RN50 ₁₄₅	RN50 ₁₂₆	RN50 ₂₂₄	RN101 ₂₂₄	WRN28-4 ₂₁₀	VGG16 ₁₂₁	VGG16 ₉₄	VGG16 ₁₂₆	VGG16 ₂₂₄	VGG19 ₂₂₄
Parameter size	91.65MB	91.65MB	91.65MB	91.68MB	164.18MB	1.43MB (k=1) 34.95MB (k=5) 139.38MB (k=10)	400.85MB	516.16MB	516.16MB	516.16MB	536.46MB
conv1	[3 × 3, 64], stride 1	[4 × 4, 64], stride 1	[3 × 3, 64], stride 2		(7 × 7, 64), stride 2		(3 × 3, 16), stride 1			[3 × 3, 64] × 2	
maxpool	x				(3 × 3), stride 2		x	x	x	[2 × 2], stride 1	(2 × 2), stride 2
conv2x		[1 × 1, 64](3 × 3, 64)(1 × 1, 256) × 3				[3 × 3, 16] × k [3 × 3, 16] × k	× 4			[3 × 3, 128] × 2	
maxpool	x					x		x			(2 × 2), stride 2
conv3x		[1 × 1, 128](3 × 3, 128)(1 × 1, 512) × 4				[3 × 3, 32] × k [3 × 3, 32] × k	× 4	[3 × 3, 128] × 3	[3 × 3, 256] × 3		[3 × 3, 256] × 4
maxpool	x					x		(2 × 2), stride 1		(2 × 2), stride 2	
conv4x		[1 × 1, 256](3 × 3, 256)(1 × 1, 1024) × 6		[1 × 1, 256](3 × 3, 256)(1 × 1, 1024) × 23		[3 × 3, 64] × k [3 × 3, 64] × k	× 4	[3 × 3, 256] × 3	[3 × 3, 512] × 3		[3 × 3, 512] × 4
maxpool	x					x		(2 × 2), stride 2	(2 × 2), stride 1		(2 × 2), stride 2
conv5x		[1 × 1, 512](3 × 3, 512)(1 × 1, 2048) × 3				x			[3 × 3, 512] × 3		[3 × 3, 512] × 4
maxpool	x					x			(2 × 2), stride 2		
					averagpool, FC, Softmax			maxpool, FC, FC, FC, Softmax			

A.2 Ablation Study

Eigen-CAM Analysis To further examine if the surrogate copies the victim by inheriting inner representations from convolutional layers, we carry out Eigen-CAM analysis [13] as shown in Fig. 6. Even though all models predicted correctly (i.e., as sunflower), only the surrogate with the same ID has a visual explanation similar to the victim. **ID analysis of VGG16** An additional ID analysis with different model architecture VGG16 is shown in Table 9. ID is represented as a subscript. The result shows the same trend with the analysis on RN50 based model.

A.3 SCA algorithms and DCG generation result

Algorithm 2 and Algorithm 3 are shown below and Table 10 shows the values of properties obtained from Algorithm 2. (N is the number of iterations of loop. ST

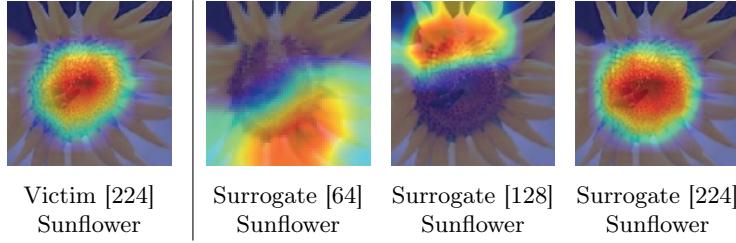


Fig. 6: Eigen-CAM Results. Only the surrogate with identical ID is similar

Table 9: ID Analysis (Datasets) with VGG16. Effectiveness (Relative Accuracy) of MEA (KnockoffNets with ImagenNet) with query-budget-60k

Dataset	Victim Model		Surrogate Model							
	Accuracy	Model	VGG16 _[32]		VGG16 _[64]		VGG16 _[128]		VGG16 _[224]	
			Test 1	Test 2	Test 1	Test 2	Test 1	Test 2	Test 1	Test 2
Indoor67	68.35(1x)		0.89x 0.89x	0.54x 0.88x	0.50x 0.48x	0.31x 0.20x				
Caltech-256	73.55(1x)	VGG16 _[32]	0.94x 0.94x	0.66x 0.96x	0.62x 0.66x	0.45x 0.31x				
CUB-200	63.82(1x)		0.91x 0.91x	0.52x 0.83x	0.49x 0.40x	0.21x 0.15x				
Indoor67	75.00(1x)		0.38x 0.35x	0.93x 0.93x	0.73x 0.86x	0.63x 0.63x				
Caltech-256	80.55(1x)	VGG16 _[64]	0.56x 0.51x	0.95x 0.95x	0.81x 0.93x	0.78x 0.80x				
CUB-200	72.56(1x)		0.24x 0.22x	0.90x 0.90x	0.64x 0.80x	0.53x 0.43x				
Indoor67	77.91(1x)		0.32x 0.30x	0.70x 0.68x	0.91x 0.91x	0.78x 0.81x				
Caltech-256	82.39(1x)	VGG16 _[128]	0.49x 0.42x	0.81x 0.78x	0.95x 0.95x	0.90x 0.92x				
CUB-200	77.30(1x)		0.16x 0.13x	0.56x 0.52x	0.91x 0.91x	0.74x 0.76x				
Indoor67	78.20(1x)		0.23x 0.27x	0.60x 0.62x	0.84x 0.82x	0.92x 0.92x				
Caltech-256	83.06(1x)	VGG16 _[224]	0.33x 0.40x	0.76x 0.75x	0.92x 0.91x	0.95x 0.95x				
CUB-200	77.11(1x)		0.10x 0.09x	0.38x 0.35x	0.76x 0.71x	0.90x 0.90x				

Algorithm 2: CreateDCG

Input: $addresses(it, on, ker), threshold$
Output: DCG_a, DCG_d : Dynamic Call Graph

```

1 for  $addr \in addresses$  do
2    $delay \leftarrow probe(addr)$                                 ▷ Time taken to access addr
3    $flush(addr)$ 
4   if  $delay < threshold$  then                                ▷ cache hit
5      $DCG_a.append(addr), DCG_d.append(delay)$ 
6 return  $DCG_a, DCG_d$ 

```

Algorithm 3: EstimateL1

Input: $m, n, Q, threshold$
Output: AT_{L1} : L1 Average Execution Time

```

1  $k' \leftarrow 4Q, A \in \mathbb{R}^{(m, k')}, B \in \mathbb{R}^{(k', n)}$ 
2 while  $GEMM(A, B)$  do  $DCG_a, DCG_d \leftarrow CreateDCG()$ 
3  $List idx \leftarrow FindIndex(['itc, onc, ker, itc, ker'], DCG_a)$ 
4  $AT_{L1} \leftarrow Avg(DCG_d[idx][0 : (idx.size() - 1)])$ 
5 return  $AT_{L1}$ 

```

Table 10: Properties of Loops obtained from DCG Generation Result

Victim Model	Loop1			Loop2			Loop3		
	N	AT	ST	N	AT	ST	N	AT	ST
$RN50_{[32]}$	1	1527	163	4	49	29	6	11.5	10
$RN50_{[64]}$	1	5774	704	13	69.1	50	6	18.3	3.0
$RN50_{[128]}$	1	5212	582	13	44.9	42	6	11.3	9
$RN50_{[224]}$	1	17665.5	8163	40	208.3	124	6	38.25	13

Table 11: Image Dimension Estimation Result. Values in SCA and target columns represent the estimated and actual values respectively

Victim Model	m		n		k		kernel		ID	
	SCA	Target	SCA	Target	SCA	Target	SCA	Target	SCA	Target
$RN50_{[32]}$	1018.8	1024	72	64	34.2	27	3.4	3	32.3	32
$RN50_{[64]}$	3983.1	3969	64	64	39	48	3.6	4	63.7	64
$RN50_{[224]}$	12541	12544	64	64	147.9	147	7	7	224	224

and AT are the short and average execution time of loop respectively.) Table 11 shows the result of ID estimation results for $RN50_{[32]}$, $RN50_{[64]}$ and $RN50_{[224]}$.

References

1. Amazon: Amazon AmazonSageMaker. <https://docs.aws.amazon.com/sagemaker/index.html> (2021), [Online; accessed 15-Nov-2021]
2. Barbalau, A., Cosma, A., Ionescu, R.T., Popescu, M.: Black-box ripper: Copying black-box models using generative evolutionary algorithms. arXiv preprint arXiv:2010.11158 (2020)
3. Beatrice, A.: Top companies using machine learning in a profitable way. <https://www.analyticsinsight.net/top-companies-using-machine-learning-in-a-profitable-way/> (Aug 2021), (Accessed on 08/23/2021)
4. Correia-Silva, J.R., Berriel, R.F., Badue, C., de Souza, A.F., Oliveira-Santos, T.: Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2018)
5. Google: Google ML Engine. <https://cloud.google.com> (2021), [Online; accessed 15-Nov-2021]
6. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. Hu, X., Liang, L., Li, S., Deng, L., Zuo, P., Ji, Y., Xie, X., Ding, Y., Liu, C., Sherwood, T., et al.: Deepsniffer: A dnn model extraction framework based on learning architectural hints. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. pp. 385–399 (2020)
9. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

10. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloji, M., Kolesnikov, A., et al.: The open images dataset v4. *International Journal of Computer Vision* **128**(7), 1956–1981 (2020)
11. Li, D., Wang, X., Kong, D.: Deeprebirth: Accelerating deep neural network execution on mobile devices. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
12. Lorica, B., Paco, N.: The state of machine learning adoption in the enterprise. O'Reilly Media (2018)
13. Muhammad, M.B., Yeasin, M.: Eigen-cam: Class activation map using principal components. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2020)
14. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4954–4963 (2019)
15. Pal, S., Gupta, Y., Shukla, A., Kanade, A., Shevade, S., Ganapathy, V.: Activethief: Model extraction using active learning and unannotated public data. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 865–872 (2020)
16. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia conference on computer and communications security. pp. 506–519 (2017)
17. Q-engineering: Deep learning with raspberry pi and alternatives in 2022. <https://qengineering.eu/deep-learning-with-raspberry-pi-and-alternatives.html> (2022), [Online; accessed 11-Apr-2022]
18. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 413–420. IEEE (2009)
19. Ribeiro, M., Grolinger, K., Capretz, M.A.: Mlaas: Machine learning as a service. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). pp. 896–902. IEEE (2015)
20. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* **115**(3), 211–252 (2015). <https://doi.org/10.1007/s11263-015-0816-y>
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
22. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: 25th {USENIX} Security Symposium ({USENIX} Security 16). pp. 601–618 (2016)
23. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
24. Yan, M., Fletcher, C.W., Torrellas, J.: Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures. In: 29th {USENIX} Security Symposium ({USENIX} Security 20). pp. 2003–2020 (2020)
25. Yu, H., Yang, K., Zhang, T., Tsai, Y.Y., Ho, T.Y., Jin, Y.: Cloudleak: Large-scale deep learning models stealing through adversarial examples. In: NDSS (2020)
26. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
27. Zhang, Y., Reiter, M.K.: Düppel: Retrofitting commodity operating systems to mitigate cache side channels in the cloud. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. pp. 827–838 (2013)
28. Zhang Xianyi, Wang Qian, and Zaheer Chothia: OpenBLAS. <http://www.openblas.net/> (2019)