Model Extraction Attack against On-device Deep Learning with Power Side Channel

***Abstract*—The proliferation of on-device deep learning models in resource-constrained environments has led to significant advancements in privacy-preserving machine learning. However, the deployment of these models also introduces new security challenges, one of which is the vulnerability to model extraction attacks. In this paper, we investigate a novel attack with power side channel to extract on-device deep learning model deployed, which poses a substantial threat to on-device deep learning systems. By carefully monitoring power consumption during inference, an adversary can gain insights into the model's internal behavior, potentially compromising the model's intellectual property and sensitive data. Through experiments on a real-world embedded device (Jetson Nano) and various types of deep learning models, we demonstrate that the proposed attack can extract models with high fidelity. Based on experiments, we find that the power side channel-assisted model extraction attack can achive high attacking success rate, up to 96.7% and 87.5% under close world and open world settings. This research sheds light on the evolving landscape of security threats in the context of on-device deep learning and provides valuable insights into safeguarding these models from potential adversaries.**

***Index Terms*—Model extraction attack, on-device deep learning, power side channel**

I. Introduction

Deep learning (DL) has seen remarkable progress over the past decade and has gained widespread popularity for performing various tasks such as image classification, natural language processing, and security enhancement, among others. According to a Statista report [1], the global market for artificial intelligence (AI) is projected to reach 126 billion US dollars by 2025. Similarly, the Wall Street Journal [2] predicts that advancements in AI and ML could drive GDP growth by 14% from 2019 to 2030. On-device AI is an emerging trend in the field of artificial intelligence, driven by the need for more efficient and secure processing of data locally. With the increasing ubiquity of intelligent devices, from smartphones, smart watches to home appliances, the demand for on-device AI is rising. The deployment of on-device deep learning models avoids sharing sensitive data with the cloud and reduces network traffic, but it also poses new attack surfaces.

However, exposing deep learning on users' side also brings new attacking surfaces, such as model extraction [3], membership inference [4], etc. In response, recent research has proposed encrypting models and hiding prediction vectors to prevent leakage of models and training datasets in on-device AI at the software level. However, they ignore the information attackers gained from hardware-level [5]–[10]. Recent works show that the emerging side-channel attacks (SCAs) can observe the low-level behaviors from shared hardware components and deduce users' sensitive information such as secret keys [11], [12]. The side channels arise from various vulnerable hardware components including cache, memory, branch, floating point units and etc. For example, cache-based SCAs exploit the last-level cache, which is shared among all cores, to monitor the execution of victim applications, which can include deep learning models.

Most prior studies have focused on analyzing information leakage resulting from timing-based and access-based Side-Channel Attacks (SCAs). However, these attacks typically require the insertion of malicious code into victim devices to collect side-channel traces, a method that can be mitigated by restricting third-party programs. Additionally, some other studies, such as Patwari et al. (2022) [13], adopt a native code attack model. In this model, the CPU, memory, or storage information can be monitored using tegrastats on Nvidia Jetson Nano. Nevertheless, device vendors can still defend against these attacks by reducing the sampling rate, which results in a significantly lower attack success rate.

"In this paper, we will present a model extraction attack using the power side channel against on-device deep learning models. To achieve this, we first leverage open-source on-device deep learning models and profile them with a power monitor to build a database of deep learning models and their power consumption traces. Then, we explore a wide range of machine learning classifiers to select the optimal one for predicting the victim model's architecture. Lastly, we consider two attacking scenarios, namely, close world and open world, to evaluate the effectiveness of the presented attack.

The rest of the paper is organized as follows: Section II introduces the background of SCAs, on-device deep learning, and related works. Section III presents threat model of the proposed attack. The details of the proposed model extraction attack with power side channel are introduced in Section IV. Section V comprehensively evaluates the presented attack with three real-world datasets. Section VI discusses defense approaches and future works. Finally, Section VII presents the conclusion of this study.

II. Background and Related Work

A. Side-Channel Attacks

Side-channel attacks (SCAs) refer to a set of attacks that exploit the side effects of a victim's execution and compromise users' privacy, such as stealing passwords, secret keys, and so on [11]. There are three major categories of SCAs: timing-based, access-based, and trace-based [14]. For both

timing-based and access-based SCAs, attackers have to place malicious code alongside victim deep learning models before launching the attacks. However, edge vendors like Android OS can forbid the installation of third-party apps or enable detection tools against such attacks. Hence, this work focuses on trace-based SCAs, which exploit the measurement of a victim program's execution and deduce secrets, such as power [9], electromagnetic radiation [15], and hardware performance counters [16], among others. Such trace-based side channels do not need to intrude into victim systems [9], [16].

B. On-device Deep Learning

The considerable progress made in deep learning has motivated its adoption across various domains, including education, health, finance, and home appliances. Compared to relying on cloud servers for DL computations, on-device DL enables intelligent apps with less bandwidth, reduced latency, and enhanced confidentiality. According to recent Samsung research [17], the number of apps with DL models increased from 165 to 342 between 2020 and 2021, highlighting their increasing popularity. These DL models are widely adopted in diverse apps, ranging from communication, finance, and medical to social networks. Most major vendors have launched mobile DL frameworks, such as TensorFlow Lite from Google, PyTorch Live from Facebook, Core ML from Apple, and NCNN from Tencent, to fast-track the development process for on-mobile DL models. Although obfuscation and encryption approaches have been proposed to protect on-device DL models against leakage from installation files, attackers can still observe the computations of DL models at the hardware level and recover secrets, including DL architectures, weights, and labels.

C. Related Work

Previous studies have utilized various side-channel data to target different aspects such as model architecture, inputs, or parameters [5]–[10]. Some of these studies have aimed to decipher or reconstruct model network architectures. This is crucial for two reasons: first, network architectures are often proprietary; second, understanding these architectures can enhance the effectiveness of adversarial and membership inference attacks, as evidenced in several studies. Common sources of side-channel information include cache, memory access, electromagnetic (EM) emissions, power, timing, and GPU statistics. While the focus has primarily been on network architectures, efforts have also been made to extract model parameters and inputs. For example, Wei et al. [9] demonstrated the feasibility of using power traces to retrieve input images from an FPGA-based CNN accelerator with known parameters. This work involves inserting a power monitor circuit into FPGA-based accelerators, which is not applicable to most edge devices, and vendors can also implement insertion behaviors. Additionally, Chmielewski et al. [18] analyzed EM emanations and timing from neural network layers running on a GPU, successfully identifying the number of layers, neurons per layer, and types of activation functions, despite investigating a different side channel with a much lower sampling rate.

Besides deep learning architectures, inputs and parameters are also under threat. Hua et al. [6] investigated the leakage from the memory side channel on hardware accelerators and found that memory access patterns can reverse-engineer both the structures and weights of CNN models. In addition to machine learning designs, the inputs and labels are also threatened. Xiang et al. [9] utilized a power side channel to reconstruct input images on FPGA-based accelerators. The capability of this attack was evaluated using one of the most prominent image datasets, the MNIST dataset [19], achieving an accuracy of 89%. Liu et al. [10] presented that using software-based side-channel telemetries can help infer the output class of input images.

III. Threat Model

The target of the proposed attack is the running applications on edge devices, which have embedded deep learning models to empower intelligent functionalities such as health monitoring, heart attack detection, etc. The goal is to deduce the architectures of the on-device deep learning models that are running. We assume that the attacker has no access to the information about the model architecture and no ability to insert malicious code into the system to collect system-level information about the running models.

A. Attacking Settings

To better represent the real-world attacking scenario, we consider two attack settings: closed world and open world, as detailed below.

Close world: In the close world setting, each model is sensitive and exists in the training dataset. The attacker's objective is to identify the model architecture used in each on-device deep learning application.

Open world: In the open world setting, only one model is sensitive and exists in the training dataset, while others in the training dataset will be categorized as "others" (meaning not sensitive). The attacker will also need to deal with some unknown models and classify them as "other."

B. Assumption

Our attack strategy is based on three key assumptions. First, to deduce the architecture of the target deep learning (DL) model, the attacker needs to identify the type of edge device used by the victim. The attacker can acquire a similar device, referred to as the attacker's device, which is used to run DL models and gather system traces, thereby emulating the victim's device. Second, it is assumed that only a single DL model operates at any one time on the victim's device, without any concurrent background processes. This reflects the actual running application's global statistics, a plausible assumption considering the targeted edge device's limited resources and typical deployment for specific tasks. Lastly, the attacker presumes that the victim's DL model is part of a recognized family of network architectures, specifically including but not limited to MobileNet, VGG, ResNet, and DenseNet. Although we consider known deep learning models

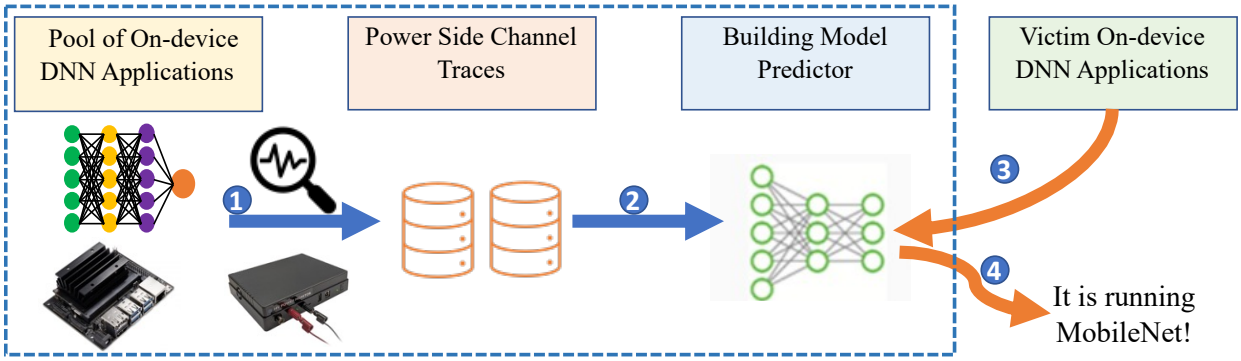

Fig. 1. The overview of the proposed model extraction attack with power side channel.

as victims, we believe this is practical since most on-device deep learning models are trained with customized datasets based on pre-trained models.

IV. Proposed Attack

This work demonstrates a mode extraction attack that aims to deduce the on-device deep learning model used in victim applications through a power side channel. As depicted in Figure 1, the entire attack consists of two parts: *offline preparation* and *online deployment*. In offline preparation, known deep learning models from an open-sourced repository are profiled using a power monitor. This process involves collecting power consumption traces and using the collected datasets to build a predictor to infer the running on-device deep learning models. Once trained, the power monitor can be launched to profile unknown on-device deep learning models and send them to the attacker's classifier for deducing the running victim model. The details of the attack are introduced in the following sections.

Fig. 2. High voltage power monitor from Msoon Solution [20]

A. Profile On-Device Deep Learning Models via Power Monitor

1) Power Monitor Setup: We leverage the high voltage power monitor from Msoon Solution [20] and install the provided software on Window to transfer the measured power power consumption of Jetson Nano to our desktop. As seen in Figure 2, there are two alligator clips (in black and red color) which can be used to connect and power edge devices. The voltage output of the power monitor ranges from 0.8 $\sim$ 13.5V, and up to 6A, which fits most edge devices. The sampling rate is 5KHz.

In this work, we chose the Jetson Nano 4GB from the Jetson series of devices as a case study while we believe the approach can also apply to other platforms. Jetson Nano can be powered on via USB-C connector or power pins where the latter one fits our demand. Hence we choose to connect the two alligator of power monitor to Jetson Nano pin header. This device features a Quad-core ARM A57 CPU operating at 1.43 GHz, a 128-core Maxwell GPU, and 4 GB of 64-bit LP DDR4 memory with a bandwidth of 25.6 GB/s. It runs on the Linux4Tegra OS, which is based on Ubuntu 18.04, and we utilized JetPack SDK version 4.5 for development. The Jetson Nano offers two power modes: a standard 10W mode and a more energy-efficient 5W mode. Our data collection was conducted in the 10W mode. Additionally, we operated the Jetson Nano in performance mode. This mode deactivates the DVFS governor and maintains the processor speeds at their maximum, making it an optimal setting for deep learning applications that leverage the GPU. In terms of AI capabilities, the Jetson Nano is capable of delivering up to 472 GFLOPs.

2) Victim On-device Deep Models: Since we assume that attackers have access to models in open repositories, all victim on-device deep learning models are from open-sourced repository [21], including MobileNetV1 [22], MobileNetV2 [23], VGG16/19 [24], ResNet18/50 [25], DenseNet121/169 [26]. All the models are implemented in PyTorch. The pretrained models encompass the three stages of inference: data loading, model (including weights) loading, and execution of model inference. As for the input, we assume the victim on-device deep learning models take in images with size of 224x224 pixels and conduct a standard normalization process. Our experiments involve running inference on 100 images from the ImageNet ILSVRC Test set. We employ PyTorch Dataloaders for random shuffling and sampling of these images. The images are then normalized and transferred to the GPU for pro-

cessing in single-batch inference. This data loading process, using images from the ImageNet Test set, remains consistent across all applications. We select models for these applications from a pool maintained by the attackers, load the chosen model and its pretrained weights onto the GPU, and then carry out single-batch inference to determine the predicted class. The overall structure is common to all applications, differing only in the choice of model.

B. Power Side Channel Prepossessing

The execution time of on-device deep learning models ranges from a few milliseconds to a few tens of milliseconds, which causes various lengths of samples in each power trace. In response, we will choose the longest length among all collected power traces as the fixed one and pad with zeros if the length of a power traces is less. For the investigated victim models, the longest execution time is eleven milliseconds. This equates to 55 samples in the power traces. Hence we choose 60 as the length of all power traces and conduct padding with zeros for all traces that have fewer samples.

C. Building On-device Deep Learning Model Predictor

In the building step, 100 traces from each victim on-device deep learning model are collected and labeled. The labeled dataset is used to train various types of machine learning-based model architecture predictor including classical machine learning, time-series, or deep learning models. The rationale for choosing these machine learning models is that they are from different branches of ML including classical model: RandomForest (RF), support vector machine (SVM); deep learning models: fully connected network (FCN), long short-term memory (LSTM); time-series models: dynamic time warping (DTW), Shapelet, covering a diverse range of learning algorithms that support our comprehensive analysis and experiments. For the attacking phase, the proposed attack model receives unlabeled power traces in which each of the trace is corresponding to a victim model inference and the trained model predictor outputs the prediction results. Each victim model has 100 traces in total, with 70 of them used for training the model architecture predictor and the remaining 30 for testing it.

D. On-line Attack Procedure

As presented in Figure 1, we setup the power monitor for unknown incoming victim applications with deep learning model inside.

- **Step 1:** Initiate power monitor on the edge devices and start the victim applications with required input images.
- **Step 2:** Preprocess the collected power traces with the same preprocessing approach (IV-B) as the one in preparation.
- **Step 3:** Send processed data to the attacker's model predictor for deducing the deep learning models used in the victim applications. The deduced label information can be further leveraged by other malicious activities.

TABLE I
TRAINING AND TESTING DATASET FOR CLOSE WORLD.

Family	Victim Model	Size in Training	Size in Testing
MobileNet	MobileNetV1	70	30
MobileNet	MobileNetV2	70	30
VGG	VGG16	70	30
VGG	VGG19	70	30
ResNet	ResNet18	70	30
ResNet	ResNet50	70	30
DenseNet	DenseNet121	70	30
DenseNet	DenseNet169	70	30

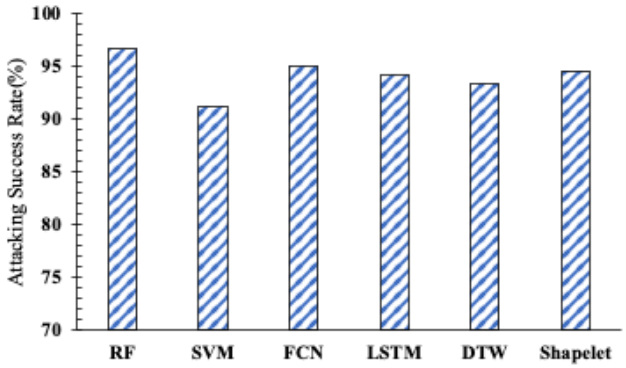

Fig. 3. Close world attacking success rate with various machine learning classifiers.

V. EVALUATION RESULTS

In this section, we evaluate the attacking success rate, i.e., model predictor accuracy on victim deep learning models, under close world and open world settings.

A. Close World Extraction Results

In this setting, we consider attackers have access to all victim on-device models and have corresponding power profiling results. As shown in Table I, there are total of 560 records in training dataset and 240 records in testing dataset. As presented in Figure 3, we observe all attacking success rate of the six classifiers obtain above 90% accuracy Among all six classifiers, we find that RF performs the best and achieves 96.7%, indicating a high risks of leaking model architecture information to malicious party.

TABLE II
EXAMPLE OF TRAINING AND TESTING DATASET FOR OPEN WORLD.

Family	Victim Model	Size in Training	Size in Testing	Label
MobileNet	MobileNetV1	70	30	Sensitive
MobileNet	MobileNetV2	70	30	Sensitive
VGG	VGG16	70	30	Others
VGG	VGG19	70	30	Others
ResNet	ResNet18	70	30	Others
ResNet	ResNet50	70	30	Others
DenseNet	DenseNet121	0	30	Others
DenseNet	DenseNet169	0	30	Others

B. Open World Extraction Results

In the real world, an attacker might not have access to all victim models and may be interested in only one sensitive

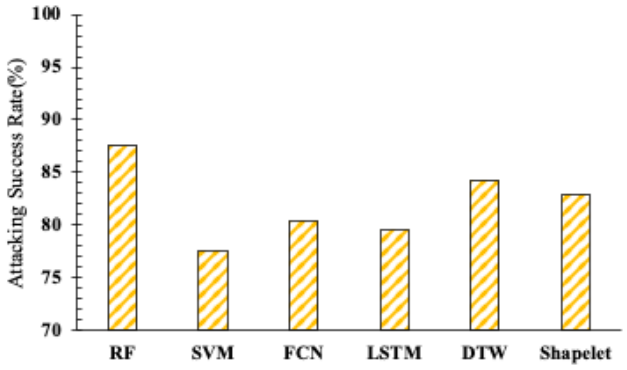

Fig. 4. Open world attacking success rate with various machine learning classifiers.

model. Hence, we also consider an open-world attacking setting. Regarding the dataset, we designate one model family from the four as sensitive, two of the four as others, and the remaining one as unknown. We rotate the sensitive and others and average the attacking success rate of the four rotations. In each rotation, we include a data record as exemplified in Table II, and there are 240 records for testing in each rotation. As seen in Figure 4, we observe a decrease in attacking success rate by around 10% compared to the closed world. Therefore, it is more important for the attacker to choose the most optimal classifiers under the open-world setting. However, the proposed model extraction attack can still achieve a 87.5% prediction accuracy of victim deep learning models with RF. This provides insights that the attacker can still have model architecture extraction ability even with unknown datasets, highlighting the importance of defending on-device deep learning models against the presented attack with power side channel.

VI. Defense Discussion and Future Works

A. Potential Defense

Since the external measurement of power consumption cannot be controlled by the vendor, it is important to mask the power side channel during the execution of on-device deep learning models. Therefore, we propose adding random fluctuations to power traces as a countermeasure against the presented model extraction attack. To achieve this, the victim application can create a dummy thread that executes some lightweight instructions, causing noise in power consumption. However, this might result in extra performance and energy costs, which some edge devices are less likely to afford. A more fine-grained method can leverage adversarial learning to generate decoy instructions that can mask the victim's deep learning models while also minimizing power costs.

B. Future Works

Although promising results were obtained in this work, there are still two avenues for future exploration to comprehensively assess the threats of exposing on-device deep learning model architectures. Firstly, the influence of image size on our model predictor can be investigated in the future. Since different applications have their own settings for inputs, we will extend the experiment to evaluate whether the presented attack can also be generalized to inputs of various sizes. Secondly, an anomaly detection method can be used to build a robust machine learning classifier in an open-world setting. In this work, we leveraged only supervised classification algorithms, whose main drawback is dealing with unknown datasets. Anomaly detection algorithms, in contrast, focus on defining normal behaviors and identifying outliers.

VII. Conclusion

The progress made in deep learning (DL) boosts its application in edge devices, including mobile phones, wearable watches, and home appliances. Protecting the on-device architecture becomes a critical part of securing intellectual property (IP). This paper investigates the potential of the power side channel to reveal on-device deep learning models. To achieve this, we profile on-device DL models downloaded from an open-source repository via a power monitor and then train a model architecture predictor using various machine learning classification techniques. We find that Random Forest (RF) performs the best for both closed and open world settings, achieving 96.7% and 87.5% attacking success rates, respectively. This highlights the importance of masking hardware-level information for on-device models.

References

[1] "Statista," in *https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/*.

[2] "Wall street journal," in *https://www.wsj.com/articles/the-economic-value-of-artificial-intelligence-1540568499*.

[3] X. Gong, Y. Chen, W. Yang, G. Mei, and Q. Wang, "Inversenet: Augmenting model extraction attacks with training data inversion.," in *IJCAI*, pp. 2439–2447, 2021.

[4] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18, IEEE, 2017.

[5] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitraş, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *arXiv preprint arXiv:1810.03487*, 2018.

[6] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2018.

[7] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in *29th USENIX Security Symposium (USENIX Security 20)*, pp. 2003–2020, 2020.

[8] M. Luo, A. C. Myers, and G. E. Suh, "Stealthy tracking of autonomous vehicles with cache side channels," in *29th USENIX Security Symposium (USENIX Security 20)*, pp. 859–876, 2020.

[9] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. 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*, pp. 393–406, 2018.

[10] C. Liu, M. Kar, X. Wang, N. Chawla, N. Roggel, B. Yuce, and J. M. Fung, "Methodology of assessing information leakage through software-accessible telemetries," in *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 259–269, IEEE, 2021.

[11] D. Gruss, C. Maurice, K. Wagner, and S. Mangard, "Flush+ flush: a fast and stealthy cache attack," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 279–299, Springer, 2016.

[12] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. B. Lee, "Last-level cache side-channel attacks are practical," in *2015 IEEE symposium on S&P*, 2015.

[13] K. Patwari, S. M. Hafiz, H. Wang, H. Homayoun, Z. Shafiq, and C.-N. Chuah, "Dnn model architecture fingerprinting attack on cpu-gpu edge devices," in *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pp. 337–355, IEEE, 2022.

[14] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses," *Journal of Hardware and Systems Security*, vol. 3, no. 3, pp. 219–234, 2019.

[15] R. Wang, H. Wang, and E. Dubrova, "Far field em side-channel attack on aes using deep learning," in *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, pp. 35–44, 2020.

[16] B. A. D. Kumar, S. C. Teja R, S. Mittal, B. Panda, and C. K. Mohan, "Inferring dnn layer-types through a hardware performance counters based side channel attack," in *The First International Conference on AI-ML-Systems*, pp. 1–7, 2021.

[17] M. Almeida, S. Laskaridis, A. Mehrotra, L. Dudziak, I. Leontiadis, and N. D. Lane, "Smart at what cost? characterising mobile deep neural networks in the wild," in *Proceedings of the 21st ACM Internet Measurement Conference*, IMC '21, (New York, NY, USA), p. 658–672, Association for Computing Machinery, 2021.

[18] Ł. Chmielewski and L. 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*, pp. 96–113, Springer, 2021.

[19] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*.

[20] "Mobile device power monitor manual," in *https://msoon.github.io/powermonitor/PowerTool/doc/Power*

[21] "Torchvision models," in https://pytorch.org/vision/stable/models.html.

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861*, 2017.*

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.*

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556*, 2014.*

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.*

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.*