DeepEM: Deep Neural Networks Model Recovery through EM Side-Channel Information Leakage

Honggang Yu*, Haocheng Ma†, Kaichen Yang*, Yiqiang Zhao† and Yier Jin*
*Department of Electrical and Computer Engineering, University of Florida
†School of Microelectronics, Tianjin University
honggang.yu@ufl.edu, hc_ma@tju.edu.cn, bojanykc@ufl.edu, yq_zhao@tju.edu.cn, yier.jin@ece.ufl.edu

***Abstract*—Neural Network (NN) accelerators are currently widely deployed in various security-crucial scenarios, including image recognition, natural language processing and autonomous vehicles. Due to economic and privacy concerns, the hardware implementations of structures and designs inside NN accelerators are usually inaccessible to the public. However, these accelerators still tend to leak crucial information through Electromagnetic (EM) side channels in addition to timing and power information. In this paper, we propose an effective and efficient model stealing attack against current popular large-scale NN accelerators deployed on hardware platforms through side-channel information. Specifically, the proposed attack approach contains two stages: 1) Inferring the underlying network architecture through EM side-channel information; 2) Estimating the parameters, especially the weights, through a margin-based, adversarial active learning method. The experimental results show that the proposed attack approach can accurately recover the large-scale NN through EM side-channel information leakages. Overall, our attack highlights the importance of masking EM traces for large-scale NN accelerators in real-world applications.**

I. Introduction

Neural Networks (NNs) have recently shown tremendous progress in various real-world applications, ranging across object recognition [1]–[3], natural language processing [4] and autonomous vehicles [5], [6]. Additionally, there has been an increasing effort to deploy large-scale NN models on dedicated hardware platforms such as GPU, FPGAs, or customized ASICs in order to improve the performance and efficiency of data processing systems. Hardware vendors including Xilinx and Intel spend great efforts collecting a data set, training these NNs models on it, and developing the NNs accelerators, and thus want to keep the trained models private and secret.

However, recent studies have demonstrated that severe vulnerabilities exist in hardware implementations of these NN accelerators. An adversary, who has no knowledge of the details of structures and designs inside these accelerators (i.e., black-box), can effectively reverse engineer the neural networks by leveraging various side-channel information, such as timing, power and electromagnetic (EM) emanations. For example, Hua *et al.* [7] infer the underlying network architecture and parameters (e.g., weights) of NNs running hardware accelerators by observing the resulting off-chip memory accesses while providing random inputs. Hong *et al.* [8] introduce a new attack method, known as *DeepRecon*, which accurately extracts the internal architecture of victim NNs by using the cache side-channel technique. In addition, Duddu *et al.* [9] also present that NNs are extremely susceptible to timing side-channel attacks. In their attack scheme, adversaries recover the layer's depth by applying timing side-channel information and exploit a reinforcement learning technique to search for the best substitute model with functionality similar to the victim networks. It is important to note that IP vendors do not always allow users to access these architectural side-channel information, such as memory and cache due to security and privacy concerns. Therefore, these attacks can not be conducted while targeting NNs protected in this way. To solve this problem, Batina *et al.* [10] propose a new model theft attack that exploits EM side-channel analysis to effectively reverse engineer the network characteristics of small-scale multilayer perception (MLP) and convolutional neural networks (CNNs). Specifically, the authors perform correlation electromagnetic analysis (CEMA) using the Hamming weight model to recover the networks weights. However, uniform weight setting makes current leakage models, i.e., Hamming weight and Hamming distance, slightly deviate actual EM leakages [11]. Considering the enormous parameters (e.g., weights) those large-scale neural network accelerators maintain, this deviation will significantly degrade the effectiveness of EM based model theft attacks.

To address these challenges, we present a new black-box attack that exploits EM side-channel information to effectively reverse engineer Binarized Neural Networks (BNNs), which are commonly used NNs for IoT/edge devices that apply binary values for activations and weights. Different from the previous attacks, in this study we assume the adversary has no access to the exact training data, network architecture, parameters, etc, but can only collect EM side-channel information under inference operations and observe the networks outputs (e.g, labels or confidence scores).

The key idea of our attack method is that we exploit EM side-channel information to reconstruct the network architecture of the victim BNN accelerators, including the depth and the types of layers (e.g., convolution layers, pooling layers and fully-connected layers). In addition to the network architecture, we also explore stealing the black-box model, i.e., identifying the parameters, such as weights, of the victim accelerators. Although model stealing attacks will be extremely challenging when targeting recent popular BNN accelerators like AlexNet [12], VGGNet [5] that contain millions of parameters, researchers have already proved that it is still

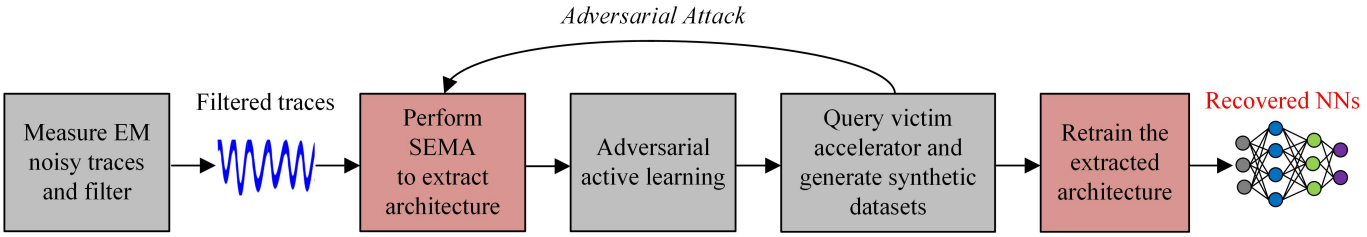

Figure 1: Overview of the Proposed Model Stealing Attack

possible to conduct accurate parameters extraction by applying a margin-based, adversarial active learning method [13]. In this study, we use a novel combination of these algorithms to accurately estimate the large-scale parameters, which are currently the bottleneck for stealing more complicated network accelerators through EM side information leaks. Figure 1 shows the overview of the proposed network theft attack targeting current popular BNN accelerators. Specifically, the margin-based, adversarial active learning algorithm proposed in [13] is used for generating malicious examples to query the victim accelerators. By using the resulting input-output pairs to retrain the substitute model with the extracted network architecture from side-channel information, an attacker can estimate the large-scale parameters (e.g., weights) of black-box victim models more efficiently. The proposed method will significantly reduce the computation overhead required to obtain the EM information leaks from NN accelerator than previous differential electromagnetic analysis (DEMA) based NN model stealing attacks [10].

The main contributions of this paper are described as follows:

- We demonstrate that attackers with no prior knowledge of the victim BNN accelerators can accurately recover the underlying model characteristics by exploiting EM side-channel information leaks from hardware implementations. By applying simple EM analysis (SEMA), attackers can develop the topology of the substitute models.
- We use a margin-based, adversarial active learning method to generate malicious examples, which can easily fool the substitute models to output incorrect classification results. These crafted malicious examples are then used to query the victim BNNs accelerators, which helps attackers to acquire more decision boundary information about the victim neural networks in the black-box setting.
- A black-box network extraction attack is designed in this paper for estimating the large-scale parameters of victim models and also accelerating the model theft process.
- We evaluate the proposed model theft attacks on a group of popular BNNs accelerators in real world. The experimental results show that our attack approach can successfully recover a substitute neural network with both classification performance and layout architecture similar to victim BNNs on the same hardware platforms.

The rest of this paper is organized as follows: Section II discusses some relevant background on our work. Section III presents the threat model of our attack. Section IV describes our scheme of the model stealing attack. Section V presents the experimental results and analysis. Section VI reviews the related work. Section VII concludes the paper.

II. Background

This section describes the details of BNNs as well as the corresponding target hardware implementation.

A. Neural Networks

Neural Networks (NNs) consist of multiple layers of neurons with complicated architectures and leverage such a deep cascaded layer structure to capture the spatial and temporal dependencies of data. The network characteristics of NNs mainly includes two crucial aspects: (1) *architecture*, including the number and types of layers, connection topology between layers, etc; (2) *parameters*, such as weights, filter size, padding, etc. Currently, the most popular NNs such as LeNet [14], AlexNet [12] and VGGNet [5] are composed of three types of hidden layers for conducting various neural processing such as convolution, pooling and fully connected. In these networks, convolution layers extract feature maps from the input data through the application of relevant filters (i.e., kernel-based filters), and then pooling layers perform global average or max pooling on the resulting feature maps for a spatial dimension reduction. Ultimately, the fully connected layers with the same principle as the traditional multi-layer perceptron (MLP) predict the class labels as well as their confidence scores by computing weighted summations, adding certain biases and applying non-linear activation functions. Moreover, the associated parameters in each layer are statically configured at the beginning and will be partly updated (e.g., weights) in the training stage of neural networks by using gradient descent algorithm or its variants such as stochastic gradient descent (SGD) algorithm.

B. BNNs

BNNs are the neural networks that use binary weights and activation values. With binary values, BNNs can reduce the memory requirement and computational complexity without excessively sacrificing accuracy, and hence are well suitable for power and resource constrained platforms. Various binarization methodologies have been developed recently for BNNs constructions [15]–[18]. Mathematically, the nonlinear

activation output y for the neuron of the fully BNNs can be denoted as follows:

$$y = f_a(\sum_{i=0}^{S} \omega_i x_i + b) \tag{1}$$

where f_a is the activation function, S is the number of synapses which are connected to each neuron in the current layer, ω_i and b are weights and bias for the neuron, respectively. Each hidden layer has a set of neurons, which are connected to the neurons of the previous hidden layer and serve as computational units to transform its input data into representations using particular activation functions such as sigmoid function, tanh function and Rectified Linear (ReLU) function. In this paper, we consider the ReLU as the activation function of victim BNNs due to the following advantages: (a) Sparse activation; (b) Efficient computation; and (c) Better gradient propagation.

In this study, the binarization of real-valued weights or activations can be done using a simple sign function as follows:

$$x_b = \text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

where x_b is the binary values. Consequently, the binary values of weights or activations can be represented either $+1$ or -1 through Equation (2) and then be encoded with a 0 for -1 or 1 for $+1$ in order to perform an XNOR logical operation on a bitwise level. In BNNs, the computations including dot production and accumulation are reduced to these simple XNOR bitwise operations, enabling faster execution times and requiring less hardware resources. Therefore, the BNNs are well-suitable for source constrained platforms such as edge devices in IoT systems [19].

The BNNs system involves two crucial phases: *training* and *inference*. During the training process, the neural network is trained on the dataset collected by the vendors or users. This process partly updates the parameters through SGD algorithm which exploits binarized weights and activations to compute the parameters gradients. The pre-trained network can be applied for the inference on the dedicated hardware platform. During the inference, the network architecture and associated parameters are statically configured and will be used to make predictions on new inputs, e.g., for classification problems.

We build the hardware implementation of the BNNs inference by using a typical neural network accelerator architecture as shown in Figure 2. Specifically, the victim BNNs accelerator receives instructions from the CPU processor, reads inputs and weights from off-chip DRAM, and finish the data transfer between off-chip DRAM and on-chip RAM through direct memory access (DMA) system. In BNNs accelerator, the processor usually supports a *popcount* instruction that counts the number of set bits in binary values for accumulation. After accelerating the neural processing through matrix multiplications and accumulations in the processing element (PE) array, the intermediate results of all layers such as output feature

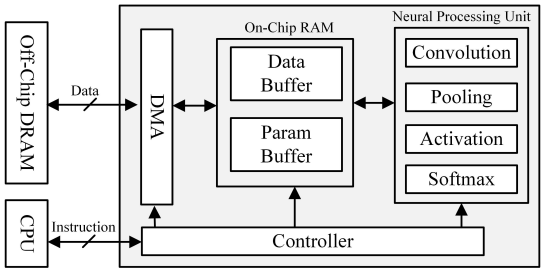

Figure 2: A typical BNNs inference accelerator

maps would be stored on the on-chip memory. Since the entire neural network is stored on the FPGA, previous model stealing attacks through memory side-channel information leaks are not applicable to on-chip BNNs [7]. The system controller is considered here as a part of accelerator to monitor various computation tasks among different components. By combining the computation results of all layers such as convolution, pooling and fully-connected layers, the BNN accelerator eventually returns the class label and/or confidence score of each image as classification outputs.

The adversary in our work targets this BNN model inference phase and wants to extract a copy of the victim neural networks by directly observing the hardware accelerator.

C. Adversarial Active Learning

Active learning (AL) aims to select a "useful" subset of unlabeled data set (i.e., image pool) for effectively training the classifier in the target domain by exploiting special sample strategies such as random strategy, certainty strategy and uncertainty strategy [20]–[24]. As a result, such a selection process would greatly reduce the label efforts of human experts while simultaneously maximizing performance of the classifier. Unlike these traditional AL methods, new adversarial attacks such as *FeatureFool* are proposed to generate the image pool [13]. The key idea of the attack is that, compared to random examples (i.e., legitimate examples), adversarial examples would extract more information about decision boundary of the victim accelerator, and hence help accelerate the network parameter theft process. Specifically, the proposed model stealing attack relies on the uncertainty strategy to choose the most "informative" examples from the image pool and then train the substitute network extracted by exploiting EM side-channel information leakages. The malicious examples annotated with black-box victim accelerator will be an ideal data set to train the substitute network in order to effectively steal a exact copy of the victim accelerator.

III. Threat Model

In this work, we consider a victim neural network $\mathcal{F}_v$ with domain input $\mathcal{X} \subseteq \mathbb{R}^n$ and output $\mathcal{Y} \subseteq \mathbb{R}^m$. The main goal of the attacker is to accurately reverse engineer the large-scale network architecture through EM side channel information.

Scenario. In this study we consider a BNN system as shown in Figure 3. Both $\mathcal{F}_v$ and $\mathcal{F}_s$ are neural networks which consist of a sequence of various neural processing units, including

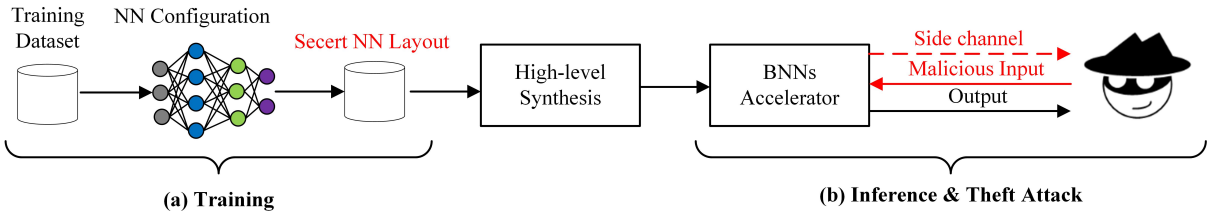

Figure 3: Binary Neural Networks (BNNs) system. The goal of an adversary is to accurately recover an exact copy of the victim network by directly observing the BNN accelerator.

convolution, pooling and fully-connected layers. Here, we consider a difficult case where the victim networks involve different types of layers and large-scale parameters, e.g., weights. We further assume that the training phase of BNNs is trusted but the BNNs accelerator is deployed on platforms that are vulnerable to side-channel attacks.

Attacker's capability. We assume that an attacker who targets the victim BNNs has no knowledge of the details of the neural networks, including the exact training set, architecture, parameters, etc. We are considering a black-box scenario. Moreover, the internal operations and behaviors of the victim accelerator cannot be directly observed and changed by inserting hardware Trojan into neural computing frameworks. We also assume that the victim accelerator under consideration does not include any defense mechanisms against EM side-channel attacks. This is a practical assumption since the threat of EM side channel threats to large-scale neural network accelerators is still not well-perceived.

Moreover, we assume that the adversary has a non-invasive and passive access to the hardware device running a victim BNNs accelerator. Specifically, the only capabilities of an adversary are to control the inputs to accelerator, and collect multiple EM side-channel measurements from a microprocessor to observe the input-output query behavior of the victim accelerator. In the proposed attack scheme, the adversary mainly focuses on exploiting the vulnerabilities in the hardware implementations of integrated BNNs accelerators to extract the networks characteristics, such as architecture and parameters for Intellectual Property (IP) theft. Generally, this would be an adversary motivated by economic incentives to legally acquire a substitute network $\mathcal{F}_s(x)$ of the secret victim network $\mathcal{F}_s(\cdot) \approx \mathcal{F}_v(\cdot)$ for future applications. As a result, attackers can arbitrarily employ the substitute BNNs extracted by the attack method on their dedicated hardware platforms, i.e., IP theft.

IV. Methodology

This section presents our model theft attack which reverse engineers the underlying BNNs accelerator's characteristics, such as network architecture and parameters, in the black-box setting.

A. Architecture Extraction

We first discuss how an adversary can reverse engineer the black-box victim accelerator through EM side-channel analysis (SCA). SCA usually exploits the vulnerabilities on the hardware implementations level to obtain crucial information about the computations occurring inside an electronic device [25], [26]. Our EM SCA attacks involve a target FPGAs-based platform running BNNs classifiers, where the EM radiation is collected by an adversary for network theft. We select ZYNQ XC7000 SoC, which is included on the Pynq-Z1 board [27], as the target FPGA chip.

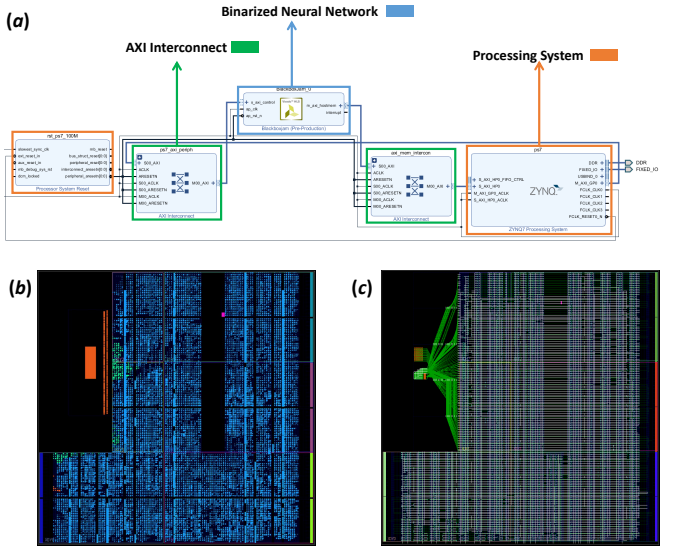

Figure 4: FPGA based implementation of BNNs system: (a) schematic; (b) cells layout; and (c) nets layout.

Figure 4(a) shows the schematic of the FPGA based BNNs system, containing programmable logic (PL), processing system (PS) and AXI interconnect. Note that the BNNs bitstream is configured on the PL component. The PS provides instructions to configure and control the behaviors of the BNNs. These instructions and data transactions are delivered by the AXI interconnect. To obtain the configured bitstream, the overall design is synthesized, mapped, placed and routed to actual resources (cells and nets) of the FPGA layout as shown in Figure 4(b). The yellow, green and blue blocks are the occupied cells of PS, AXI interconnect and BNNs parts respectively. The white lines in Figure 4(c) denote partial nets of the BNNs parts. These cells, i.e., LUT, RAM, and Registers, are configured to execute specific functions of the BNNs system. The nets are applied for power routing

and signal routing. Dynamic current flows are generated due to the charging and discharging of parasitic capacitance, as characterized by

$$I_{dyn,i}(t) = \frac{1}{2} C_i \cdot D_i(t) \cdot V_{DD} \cdot f_{clk} \tag{3}$$

where $I_{dyn,i}(t)$ denotes transient current flow, C_i is the capacitance of net i, $D_i(t)$ is its transient transition rate, V_{DD} is the voltage supply, and f_{clk} is the clock frequency [28]. Further, the EM radiations are emanated from these metal nets carrying time-varying currents. Considering the multi-layer structure of the FPGA chip shown in Figure 5, the magnetic field B_z along the z-axis is calculated at a point by Biot-Savart law:

$$B_z(t) = \frac{\mu_0}{4\pi} \sum_{i \,\epsilon\, nets} \frac{I_{dyn,i}(t) \cdot l_i}{r_i \cdot \sqrt{l_i^2/4 + r_i^2}} \tag{4}$$

where μ_0 is the magnetic permeability, l_i is the length of net i, r_i is the distance from the middle of the net i to the target point. Hence, we can conclude that the EM field is proportional to the number n, transient transition rate $D(t)$ and capacitance C of nets.

In the FPGA based BNNs system, the instructions transmitted from the PS part are generally not continuous, resulting in a discrete value of $D(t)$ in the temporal domain. Therefore, its EM field emanates in the form of discrete pulses. Nevertheless, computations are executed consecutively in the inference phase of the BNNs part. The $D(t)$ of its nets is a continuous function. So the EM emissions from this part are continuity curves in the time domain. Similarly, we can illustrate the EM patterns of each layer of BNNs.

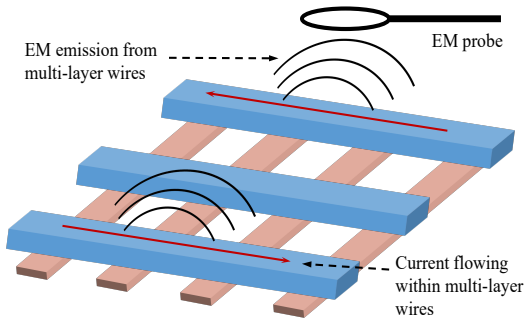

Figure 5: EM emissions from multi-layer wires of FPGAs.

As mentioned, the factor $D(t)$ is related to the characteristics of computations involved in each layer. Meanwhile, previous works have validated that the parameters of a layer directly determine the execution time of its sequential computations [29]. Hence, the temporal behavior of each layer's factor $D(t)$ is proportional to its own parameters which can be described as follows.

$$T(D_i(t)) \propto P_i \quad i \,\epsilon\, layers \tag{5}$$

Table I: Layer parameters of Neural Networks

Layer Parameter	Definition
w_{in}, w_{out}	Width of the input/output feature map
d_{in}, d_{out}	Depth of the input/output feature map
p	Indicator of pooling layer
z_{in}, z_{out}, z_f	Size of the input/output filter
f_{conv}, f_{pool}	Width of the convolution/pooling filter
s_{conv}, s_{pool}	Stride of the convolution/pooling filter
p_{conv}, p_{pool}	Padding of the convolution/pooling layer

We illustrate a set of hyper-parameters that the adversary needs to fully determine the entire network architecture in Table I. The convolution layer within the BNNs accelerator performs the non-linear operation by using the multiply-accumulators computing weighted sums of their inputs. Given the number of input channels d_{in} and output channels d_{out}, the size of the output feature map $w_{out} \times w_{out}$, the size of the filter kernel $f_{conv} \times f_{conv}$, the approximation number of parameters P_c of a convolution layer can be denoted as follows.

$$P_c = (w_{out})^2 \times d_{out} \times (f_{conv})^2 \times d_{in} \tag{6}$$

In this paper, max-pooling layers are applied to downsample the input feature map from previous hidden layers. The approximation number of parameters of a max-pooling layer depends on the size of the filter kernel, which can be denoted as follows.

$$P_p = (f_{conv})^2 \tag{7}$$

Similar to MLP, fully-connected layers connect every neuron in the neighboring layer to classify the input images. Typically, these layers perform matrix multiplications between input nodes m and output nodes n. Hence, the approximation number of parameters of fully-connected layers can be denoted as:

$$P_f = m \times n \tag{8}$$

Based on the above analysis, we can split the overall EM trace and distinguish the depth and type of individual layers, which is the basis of architecture reconstruction through EM side-channel analysis. This relation can be simply denoted as:

$$T(B_c(t)) : T(B_p(t)) : T(B_f(t)) = P_c : P_p : P_f \tag{9}$$

So far, we show that a few crucial network hyper-parameters, including layer boundaries, depth, and types, can be relatively easily revealed by an adversary through directly observing the EM emissions pattern in the time domain. However, not all network hyper-parameters can be accurately recovered through EM side-channel information leaks. For example, the layer dimensions, the width of input/output feature map, and filter kernel sizes are also crucial information for an adversary to rebuild a copy of the victim network. We illustrate the relationship between the hyper-parameters for each

layer in Equations (10)-(18). Similar to [7], [30], we follow Equations (10)-(18) to enumerate a small number of candidate architectures by reasonably guessing these hyper-parameters and then choosing the best architecture by comparing their accuracy on the same test set. Moreover, since the parameter p in Equation (13) is obtained by directly observing the timing behavior of EM traces, our method can significantly reduce the number of possible architectures.

$$z_{in} = (w_{in})^2 \times d_{in} \tag{10}$$

$$z_{out} = (w_{out})^2 \times d_{out} \tag{11}$$

$$z_f = (f_{conv})^2 \times d_{in} \times d_{out} \tag{12}$$

$$w_{out} = \frac{\frac{w_{in} - f_{conv} + p_{conv}}{s_{conv} + p_{conv}} + 1 + p\,(p_{pool} - f_{pool})}{s_{pool} \times p + \bar{p}} \tag{13}$$

$$s_{conv} \leq f_{conv} \leq \frac{w_{in}}{2} \tag{14}$$

$$s_{pool} \leq f_{pool} \leq \frac{w_{in} - f_{conv} + p_{conv}}{s_{conv}} + 1 \tag{15}$$

$$s_{pool} \leq f_{pool} \leq \frac{w_{in} - f_{conv} + p_{conv}}{s_{conv}} + 1 \tag{16}$$

$$p_{conv} < f_{conv} \tag{17}$$

$$p_{pool} < f_{pool} \tag{18}$$

The structure reverse engineering attack proposed in this section is the basis of our model theft attack. The associated parameters in each layer such as weights and biases can be effectively inferred with the knowledge of the extracted network architecture.

B. Parameter Estimation

In this section, we discuss how an adversary can reverse engineer the victim network parameters through a combination of some novel algorithms. Once the network architecture has been extracted through EM side-channel information leakages, our next step is to estimate the parameters, especially the weights by leveraging the adversarial active learning algorithm. We consider using the adversarial active strategy in [13], known as *FeatureFool*, for BNNs models. *FeatureFool* is proposed in [13] to obtain a perturbed $x' = x + \delta$ that lie approximately on the global margin (*i.e.*, decision boundary) of target BNNs classifier. Given a source image x, a target BNNs classifier $f(x)$ and the targeted label l, we use the triplet loss as a new penalty term and rewrite the initial box-constraint optimization problem in [31] as follows:

$$\begin{aligned} \min_{\delta} \quad & \|\delta\|_p + \alpha \cdot \mathrm{loss}_{f,l}\,(x+\delta) \\ s.t. \quad & x + \delta \in [0,1] \end{aligned} \tag{19}$$

where α denotes the chosen coefficient which enables to minimize all the loss terms simultaneously (similar to [13], we apply L-BFGS algorithm to solve the optimization problem in Equation (19)). The distance function $\|\cdot\|_p$ is the L_p norm which quantifies the similarity between two images in the 2-D space. Given a feature vector $\mathbf{x} = (x_1, \ldots, x_n)$, the p-norm $\|\mathbf{x}\|_p$ can be described as follows:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{20}$$

Moreover, the triplet loss function $\mathrm{loss}_{f,l}\,(x)$ in Equation (19) can be denoted as:

$$\begin{aligned} \mathrm{loss}_{f,l}\,(x) = & \max(\|\phi_K(x), \phi_K(x_t)\|_p - \\ & \|\phi_K(x), \phi_K(x_s)\|_p + M, 0) \end{aligned} \tag{21}$$

where $\phi_K(.)$ is the feature vector at k_{th} layer of BNNs classifier, M is the constant margin of the triplet loss. By adjusting the parameter M, an adversary can control the misclassification confidence score and thus effectively craft those adversarial examples that lie on the global margin of the target classifier.

For reference, we also consider other two representative strategies for crafting useful data set to train the substitute model, including *Ramdom* and *FeatureAdversary* [13]. The *Ramdom Strategy* exploited in this paper can be viewed as an extreme case where an adversary randomly samples a subset of a few examples x from unlabeled data set $\mathcal{D}_u$ and queries a black-box victim BNNs accelerator $\mathcal{F}_v$. For the *FeatureAdversary Strategy*, we leverage the algorithm proposed in [32] to minimize the L_p norms distance between the internal feature presentation of image pairs (source image x_s, target image x_t) of the victim classifier $\mathcal{F}_v$ and craft adversarial examples. Once these special examples are generated using *Ramdom Strategy* and *FeatureAdversary Strategy*, our next step is to query the victim BNNs accelerator and acquire the predictions results, e.g., labels or confidence score. The resulting image-prediction pairs will be applied to train the substitute model whose network architecture is extracted through EM side channel information leakages.

C. Evaluation Metric

Given the input x, the victim classifier $\mathcal{F}_v$ and the substitute classifier $\mathcal{F}_s$, the Average Test Error (ATE) over the test set $\mathcal{D}_t$ can be calculated by:

$$ATE = \sum_{(x,y) \in D_t} \frac{d(\mathcal{F}_v(x), \mathcal{F}_s(x))}{|D_t|} \tag{22}$$

where d is the p-norm distance function, $\mathcal{F}_v(x)$ and $\mathcal{F}_s(x)$ are ground-truth labels and estimated labels on the same test set, respectively. In our experiment, a lower ATE implies that an adversary can extract a copy $\mathcal{F}_s$ (i.e., substitute classifier) that closely matches the victim classifier $\mathcal{F}_v$ running in the hardware accelerator.

V. Evaluation

A. Experimental Setup

To evaluate the proposed model theft attack, we implement two large-scale BNN classifiers on a low-cost Pynq-Z1 development board (see Figure 6), including a 12-layer ConvNet

[33] and a more complex 23-layer VGGNet [5]. We also perform case studies on other small-scale neural networks such as LeNet [14] and AlexNet [12][1]. We use Xilinx Vivado High-level synthesis (HLS) to implement BNNs accelerators and collect EM traces under inference operations to obtain crucial information about the victim networks. The positioning system comprising of three integrated stepper motors is exploited to place the EM probe at the surface of the targeted FPGA chip running BNN classifiers. In order to improve the signal-to-noise ratio (SNR) of the EM signals, we select a set of near-field probes (RF2 from LANGER) to measure EM traces and then amplify these traces using a amplifier (PA303) up to 30dB magnification. Additionally, band pass frequency domain filtering is used for the entire EM traces to isolate the signal from noise [34].

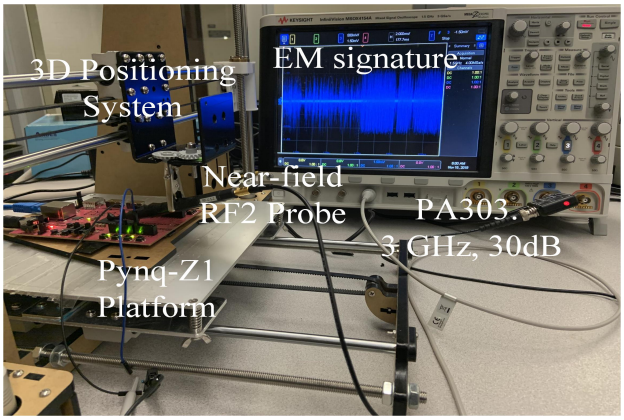

Figure 6: Pynq-Z1 side-channel characterization platform.

B. Side-channel based BNNs Recovery

1) Reverse Engineering Networks Architecture: The layer topology, such as depth and types, are necessary to achieve better classification accuracy. As such, they are crucial elements for any neural network architecture reconstruction process. In this work, we explore how an non-invasive attacker reverses engineer the layer topology by exploiting EM side-channel information and further recovers the whole network architecture. Take the ConvNet as an example, we collect 10000 EM traces from the FPGA chip and measure the timing behavior of the hidden layers. Table II lists the average execution time for each layer in ConvNet. We can see that different layers parameters configurations result in different processing times. For instance, the pooling layer requires shorter execution time than the convolution layer due to its computation simplicity. From Table II we also observe that the first fully-connected layer (FC1) requires the longest execution time since it performs most of the sequential XNOR computations. The relationship between parameters configurations and average execution time is the basis of architecture extraction through side-channel analysis. Specifically, an adversary can reverse engineer layer's depth and types by directly measuring the timing behavior of the EM traces during the inference.

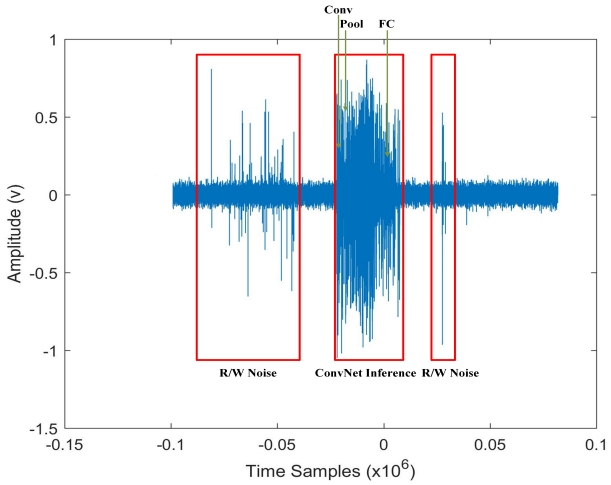

Figure 7: Observing pattern and timing of ConvNet inference - R/W indicates that BNNs system reads or writes memory operations.

So far, we have shown that the distinct EM signatures of accelerators can help an adversary to effectively extract the depth and types of layers in the large-scale victim BNN model. Table IV lists possible architecture configurations for different networks. To fully reverse engineer the entire network architecture, an adversary needs to determine other structural parameters such as the size of convolution or pooling filters, stride, etc. As shown in Figure 7, since the OS running on the SoC induces vast amounts of noise while reading/writing feature maps to the on-chip RAM, it is difficult for an adversary to infer these parameters through directly observing these noisy and imperfect EM traces. In order to address this challenge, we try to "guess" these structural parameters based on the following observations: (1) For the commonly used NNs [5], [12], [14], [35]–[37], the filter size of convolution/pooling layers are limited in a small set as shown in Table III. Larger NNs are typically constructed based on these common architecture using various learning algorithms such as transfer learning [38]–[40], and hence have the same candidate set of filter size. (2) The same convolution layers, pooling layers or FC layers are usually reused for reducing the computation complexity. In the proposed attack scheme, we assume that the types of the hidden layers are identical and the size of convolution/pooling filters are chosen from Table III. The remaining parameters associated with networks architecture, including stride and padding size, can be correspondingly calculated by Equations (10)-(18). From Table IV, we can see that the possible number of substitute model can be reduced to 11 for ConvNet. In order to obtain the best architecture configurations for ConvNet, we train substitute models with different structures, which are extracted through EM side-channel information. Figure 8 depicts the classification accuracy of possible 11 candidates structure of ConvNet. The

[1]Note that we did not present detailed experimental results of LeNet and AlexNet since their results are consistent with more complex neural network model stealing.

Table II: Layers parameters configurations and average execution time for ConvNet - N/A indicates that there is no parameter.

Layer	Conv1	Conv2	Pool	Conv3	Conv4	Pool	Conv5	Conv6	Pool	FC1	FC2	FC3
Number of parameters (Bits)	2367	135K	N/A	270K	540K	N/A	1M	2.1M	N/A	7.3M	910K	9K
Average Execution Time (ms)	1.81	2.32	**0.01**	3.71	5.60	**0.09**	10.14	15.61	**1.03**	31.14	0.12	0.08

black-box ConvNet achieves 81.25% performance on the test set. From Figure 8, we can see that the best structure (Conv11) achieves 75.15% performance, which is 29.95% higher than the worst one (Conv1 with 45.20%). This indicates that the architecture selection is extremely important for an adversary to reverse engineer the victim accelerator. In addition to

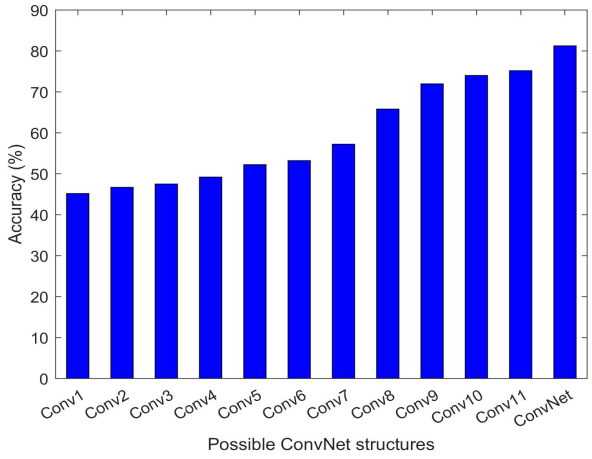

Figure 8: Possible ConvNet Structures.

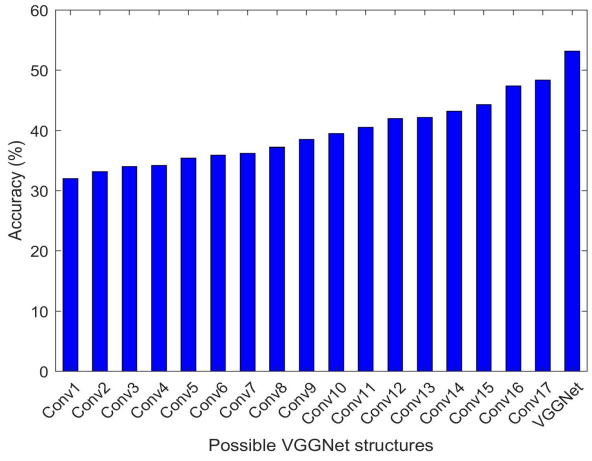

Figure 9: Possible VGGNet Structures.

ConvNet, we also perform the proposed attack on a more complex VGGNet model. Compared to ConvNet, VGGNet increases the depth of the neural network up to 23 layers by adding more convolution and pooling layers. We exploit the EM side-channel information to infer the networks' depth and layers' types and then reasonably guess the possible 17 networks structures. The classification accuracy of 17 possible structures is shown in Figure 9. The black-box VGGNet achieves 53.18% performance on the test set. From Figure 9, we can see that the substitute network with the structure Conv17 achieves the best accuracy (48.35%) on the test set.

Table III: Possible convolution and max-pooling filter configurations

Layers	Filter Size
Convolution	1x1, 3x3, 5x5, 7x7
Max-Pooling	2x2, 3x3

Table IV: Possible architecture configurations for different networks

Networks	LeNet	AlexNet	ConvNet	VGGNet
# of layers	6	8	12	23
# of possible structures	5	7	11	17

2) Reverse Engineering Network Parameters: After the recovery of the architecture, now we apply a combination of few algorithms to estimate the parameters. With the extracted network architectures (Conv11 and Conv17), we start to train the substitute models on a small amount of dataset. Then we conduct experiments with three kinds of synthetic dataset generated by *Random* (RM), *FeatureAdversary* (FA) and *FeatureFool* (FF). In each training epoch, the transfer learning technique was used to retrain original substitute models on the synthetic datasets. As shown in Table V, we can see that the substitute model (ConvNet3) with Conv11 architecture achieves 69.20% accuracy with RM samples, 75.25% accuracy with FA samples and 80.40% accuracy with FF samples, which is similar to the 81.25% accuracy achieved by the black-box ConvNet. For the VGGNet, our substitute model (VGGNet3) uses Con17 as the network architecture and obtains 96.90% performance of the black-box VGGNet by using the FF training set. From Table V we can also observe that a local substitute model trained by adversarial examples always achieves higher accuracy than the model trained by random samples which is consistent with the results in [13]. Specifically, using the synthetic dataset generated by the FF algorithm achieves the best classification performance on the same test set, illustrating that the FF algorithm can help an adversary more effectively reverse engineer the victim accelerators parameters such as weights and biases.

VI. Related Work

Model extraction attacks aim to effectively recover a exact copy of a victim network $\mathcal{F}_v$. To steal the functionality of the

Table V: Accuracy on test sets. RM = *Random*, *FeatureAdversary* = FA, *FeatureFool* = FF.

Dataset	Networks	Absolute Accuracy	Relative to Black-box
CIFAR	ConvNet	81.25%	100%
	ConvNet1 (Conv11, RM)	69.20%	85.17%
	ConvNet2 (Conv11, FA)	75.25%	92.62%
	ConvNet3 (Conv11, FF)	**80.40**%	**98.95**%
GTSRB	VGGNet	53.18%	100%
	VGGNet1 (Conv17, RM)	39.26%	73.82%
	VGGNet2 (Conv17, FA)	45.01%	84.64%
	VGGNet3 (Conv17, FF)	**51.53**%	**96.90**%

black-box victim network $\mathcal{F}_v$, an adversary retrains a substitute network $\mathcal{F}_s$ on the synthetic dataset S (i.e., $\mathcal{F}_s(x) \approx \mathcal{F}_v(x)$),

$$\mathcal{F}_s \approx \arg\min_{\mathcal{F}'} \mathcal{L}_{CE}\left(\mathcal{F}', \{(x, \mathcal{F}_v(x)) : x \in S\}\right) \tag{23}$$

where $\mathcal{L}_{CE}$ denotes the loss function that can be optimized using stochastic gradient descent (SGD) algorithm. Some existing studies demonstrate that an adversary can extract the neural networks' structure and design even in a black-box setting. For example, Papernot *et al*. [41] proposes to use the synthetic datasets generated by an adversary to train a local substitute model for the victim neural network. Since then, several studies [13], [24], [42]–[47] extend their works to design more effective synthetic datasets generation algorithms in order to improve the effectiveness of training set. In these attacks, a malicious entity aims to select the basic architecture from the candidate's model zoo as a substitute network, and retrain that network on various synthetic datasets to steal the model. Although these attacks have proved significant success, they remain impractical for stealing more complicated neural networks, especially in the case that the network architecture of the victim model can not be previously included in the candidates model zoo.

Recently, several works on network theft attacks demonstrated that side-channel analysis can be exploited by an adversary to disclose the details of structures and designs inside the victim networks [7]–[10]. In these attacks, they demonstrate that a certain hardware platform releases various side-channel information, such as timing, power and electromagnetic (EM) emanations, while performing computations. The adversary uses this side-channel information to reverse engineer the model characteristics, such as networks architecture and associated parameters. However, previous SCA based works either require physical access to CNN accelerators to collect side-channel information such as memory and power, or they suffer from extreme computation overhead while targeting complicated CNN accelerators which contain vast majority of hidden layers and parameters. To address these challenges, this paper proposes a more efficient black-box attack method that can accurately recover the victim network characteristics through a special combination of a few novel algorithms including: EM side-channel analysis and adversarial active learning. Specifically, a non-invasive and passive attacker accurately extracts the victim network's architecture by directly measuring the EM side-channel information from hardware accelerators.

VII. Conclusions

Neural networks (NNs) have been widely applied in real-world situations. These NNs, however, tend to be vulnerable to model theft attacks launched by an adversary even in black-box scenarios. In this paper, we introduce a novel model theft attack that extracts the network structure and designs inside BNN accelerators through EM side-channel information. The proposed attacks are borrow ideas mainly from the side-channel security domain and query theory, and can effectively disclose the details of the victim model, including network architecture and associated parameters. In order to properly evaluate the effectiveness of the proposed attacks, we conduct experiments on multiple commonly used neural networks, including LeNet, AlexNet, ConvNet and VGGNet. Experimental results demonstrate that the proposed attacks are more effective at stealing the large-scale NNs than previous works. Since NNs architectures are similar, our attack can be easily extended to other types of NNs such as recurrent neural networks. In the future, we will mainly focus on designing effective defense mechanisms against model stealing attacks, and therefore enhance the robustness of NN accelerators.

Acknowledgments

This work was partially supported by Office of Naval Research (ONR) Young Investigator Program (YIP) and AFRL CYAN Center of Excellence.

References

[1] K. Lee, K. Lee, K. Min, Y. Zhang, J. Shin, and H. Lee, "Hierarchical novelty detection for visual object recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] M. Feng, H. Lu, and E. Ding, "Attentive feedback network for boundary-aware salient object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3d object detection leveraging accurate proposals and shape reconstruction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] P. Teufl, U. Payer, and G. Lackner, "From nlp (natural language processing) to mlp (machine language processing)," in *Computer Network Security*, I. Kotenko and V. Skormin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 256–269.

[5] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very deep convolutional networks for natural language processing," *CoRR*, vol. abs/1606.01781, 2016. [Online]. Available: http://arxiv.org/abs/1606.01781

[6] S. M. Silva and C. R. Jung, "License plate detection and recognition in unconstrained scenarios," in *2018 European Conference on Computer Vision (ECCV)*, Sep 2018, pp. 580–596.

[7] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: ACM, 2018, pp. 4:1–4:6.

[8] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitras, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *CoRR*, vol. abs/1810.03487, 2018. [Online]. Available: http://arxiv.org/abs/1810.03487

[9] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *CoRR*, vol. abs/1812.11720, 2018. [Online]. Available: http://arxiv.org/abs/1812.11720

[10] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, Aug. 2019, pp. 515–532.

[11] S. Guilley, P. Hoogvorst, and R. Pacalet, "Differential power analysis model and some results," in *Smart Card Research and Advanced Applications Vi*. Springer, 2004, pp. 127–142.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: http://doi.acm.org/10.1145/3065386

[13] H. Yu, K. Yang, T. Zhang, Y.-Y. Tsai, T.-Y. Ho, and Y. Jin, "Cloudleak: Large-scale deep learning models stealing through adversarial examples," in *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, 2020.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*, 2016.

[16] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, and K. A. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *FPGA*, 2017.

[17] H. Yonekawa and H. Nakahara, "On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an fpga," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2017, pp. 98–105.

[18] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 4107–4115.

[19] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[20] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Advances in Neural Information Processing Systems 20*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., 2008, pp. 1289–1296.

[21] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 129–145, 1996.

[22] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08, 2008, pp. 1070–1079.

[23] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, 2002.

[24] S. Pal, Y. Gupta, A. Shukla, A. Kanade, S. K. Shevade, and V. Ganapathy, "A framework for the extraction of deep neural networks by leveraging public data," *CoRR*, vol. abs/1905.09165, 2019.

[25] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 251–261.

[26] J.-J. Quisquater and D. Samyde, "Electromagnetic analysis (ema): Measures and counter-measures for smart cards," in *Proceedings of the International Conference on Research in Smart Cards: Smart Card Programming and Security*, ser. E-SMART '01. London, UK, UK: Springer-Verlag, 2001, pp. 200–210.

[27] B. Janßen, P. Zimprich, and M. Hübner, "A dynamic partial reconfigurable overlay concept for pynq," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 2017, pp. 1–4.

[28] J. H. Anderson and F. N. Najm, "Power estimation techniques for fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 10, pp. 1015–1027, 2004.

[29] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *arXiv preprint arXiv:1812.11720*, 2018.

[30] Y. Liu, D. Dachman-Soled, and A. Srivastava, "Mitigating reverse engineering attacks on deep neural networks," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 657–662.

[31] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *arXiv preprint arXiv:1312.6199*, 2013.

[32] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[33] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs," *Int'l Symp. on Field-Programmable Gate Arrays (FPGA)*, Feb 2017.

[34] N. Chawla, A. Singh, N. M. Rahman, M. Kar, and S. Mukhopadhyay, "Extracting side-channel leakage from round unrolled implementations of lightweight ciphers," in *HOST*, 2019.

[35] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016.

[36] 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," *CoRR*, vol. abs/1704.04861, 2017.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[38] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2015, pp. 36–45.

[39] W. Ge and Y. Yu, "Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[40] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[41] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *AsiaCCS*, 2017.

[42] V. Chandrasekaran, K. Chaudhuri, I. Giacomelli, S. Jha, and S. Yan, "Model extraction and active learning," *ArXiv*, vol. abs/1811.02054, 2018.

[43] J. R. C. da Silva, R. F. Berriel, C. Badue, A. F. de Souza, and T. Oliveira-Santos, "Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data," *CoRR*, vol. abs/1806.05476, 2018.

[44] M. Juuti, S. Szyller, A. Dmitrenko, S. Marchal, and N. Asokan, "PRADA: protecting against DNN model stealing attacks," *CoRR*, vol. abs/1805.02628, 2018.

[45] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *CVPR*, 2019.

[46] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High-fidelity extraction of neural network models," *arXiv preprint arXiv:1909.01838*, 2019.

[47] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 345–353. [Online]. Available: http://papers.nips.cc/paper/6638-towards-accurate-binary-convolutional-neural-network.pdf