Remote Power Attacks on the Versatile Tensor Accelerator in Multi-Tenant FPGAs

Shanquan Tian*, Shayan Moini†, Adam Wolnikowski*, Daniel Holcomb†, Russell Tessier† and Jakub Szefer*
*Yale University, New Haven, CT, USA Email: {shanquan.tian, adam.wolnikowski, jakub.szefer}@yale.edu
†University of Massachusetts, Amherst, MA, USA Email: {smoini, dholcomb, tessier}@umass.edu

***Abstract*—Architectural details of machine learning models are crucial pieces of intellectual property in many applications. Revealing the structure or types of layers in a model can result in a leak of confidential or proprietary information. This issue becomes especially concerning when the machine learning models are executed on accelerators in multi-tenant FPGAs where attackers can easily co-locate sensing circuitry next to the victim's machine learning accelerator. To evaluate such threats, we present the first remote power attack that can extract details of machine learning models executed on an off-the-shelf domain-specific instruction set architecture (ISA) based neural network accelerator implemented in an FPGA. By leveraging a time-to-digital converter (TDC), an attacker can deduce the composition of instruction groups executing on the victim accelerator, and recover parameters of General Matrix Multiplication (`GEMM`) instructions within a group, all without requiring physical access to the FPGA. With this information, an attacker can then reverse-engineer the structure and layers of machine learning models executing on the accelerator, leading to potential theft of proprietary information.**

***Index Terms*—Machine Learning Security, FPGA Security, Hardware Accelerators, Hardware Security**

I. Introduction

Due to the high value of machine learning intellectual property and the Machine Learning as a Service (MLaaS) market, it is important to understand potential security attacks that could extract details about a machine learning model's architecture. While attacks on machine learning algorithms have been explored in CPU [1] and GPU [2] settings, they have been much less explored for FPGAs. Of the existing FPGA-related work on machine learning algorithm attacks, much of it requires physical access to the FPGA [3], [4]. Meanwhile, a number of proposals advocate for FPGA use in data centers to accelerate machine learning algorithms [5], [6], where physical access is not possible for attackers. Further, a variety of multi-tenant FPGA proposals have emerged, e.g., [7], [8], which advocate for FPGA sharing among different users to improve FPGA utilization in cloud computing data centers.

For the multi-tenant FPGA setting, researchers have already demonstrated some security threats, mainly focusing on attacks on cryptographic algorithms [9], [10]. The existing attacks have shown remote that side or covert channels in multi-tenant FPGAs can be created using signal cross-talk [11], [12], temperature [13], and on-FPGA voltage monitoring [9], [10].

This work was supported in part by NSF grants 1901901 and 1902532. One of the FPGA boards used in the experimentation was provided thanks to support of the Deputy Dean Vincent Wilczynski and the Yale School of Engineering and Applied Science.

This work extends multi-tenant attacks to machine learning algorithms. We present the first, remote power attack in a multi-tenant FPGA setting that can extract information about a machine learning algorithm's architecture. This FPGA-based attack targets an off-the-shelf Versatile Tensor Accelerator (VTA) [14] that runs a domain-specific instruction set architecture (ISA), including `LOAD`, `GEMM`, `ALU`, and `STORE` instructions, for the acceleration of machine learning models. VTA is built on top of Apache's Tensor Virtual Machine (TVM) deep learning compiler [15]. Attacking VTA is challenging since hardware-software co-design is used for TVM and the VTA hardware. Some instructions are performed on the host ARM processor, while others are implemented on the FPGA-based VTA accelerator. The VTA is a CPU-like processor with an instruction fetch module, a compute module, and load and store modules. It uses statically-scheduled task-level pipeline parallelism (TLPP) [16] to execute different instructions in parallel, and computes on instruction groups, which do not have a strict one-to-one relationship with the layers of a machine learning algorithm. These features make attacking VTA much more challenging.

Despite the challenges, we are able to realize a new remote power attack on VTA that recovers details of the different instruction groups and deduces approximate parameters of instructions within a group. Each machine learning model executed on the VTA is mapped to a unique number of groups, each with a different quantity or type of instructions. We use the recovered information to reverse-engineer the structure and type of the machine learning model's layers.

A time-to-digital converter (TDC) is used in our attack to perform the remote power measurements as the VTA executes. The measurements are collected on a Xilinx Zynq ZC706 board. We emulate a multi-tenant setting by instantiating an attacker module, with a TDC sensor, on the same FPGA as the VTA module. The TDC is logically isolated from the VTA, but due to the shared power distribution network (PDN), we are able to obtain traces of voltage fluctuations as the VTA executes. This information allows us to obtain insights about the victim's machine learning model.

II. Versatile Tensor Accelerator

This work focuses on the VTA hardware that can be implemented on FPGAs [14]. VTA is built on top of the TVM deep learning compiler stack. TVM can be used to deploy popular deep learning frameworks, such as TensorFlow, MXNet, Caffe,

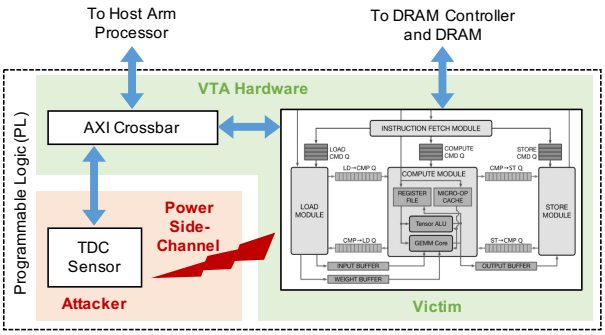


Fig. 1: Diagram of the VTA hardware (shown on green background, adapted from [14]) and the added attacker module (shown on red background). Attack setup is discussed in Section III. The VTA is unmodified. An attacker module is added to the AXI crossbar to emulate a multi-tenant FPGA setting.

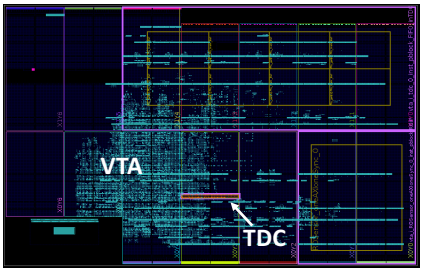


Fig. 2: Floorplan (horizontal view) for the ZC706 board showing the location of the victim VTA and attacker TDC modules.

and PyTorch, across diverse hardware back-ends, including CPUs, GPUs, and FPGA-based accelerators [15].

In this work, we use VTA as the FPGA-based accelerator back-end for the TVM. Prior to offloading computations to the VTA, the TVM/VTA software realizes a machine learning model as sets of VTA instructions, and collates them into instruction groups. A machine learning model layer may be realized by one or more instruction groups. Each group contains a mix of `LOAD`, `GEMM`, `ALU`, or `STORE` instructions. As defined by users or optimized by TVM software, some groups are executed on the ARM processor, while others are off-loaded to the VTA. Individual instruction groups must be matched to specific layers before the machine learning algorithm structure can be reverse-engineered. Using TLPP, VTA can execute multiple instructions in parallel, introducing further challenge for collecting attack measurements. Prior FPGA-based machine learning attacks, e.g., [17], [18], did not consider accelerators that have such a level of parallelism.

III. Remote Power Attack on VTA

In our attack, the VTA (victim) and TDC (attacker) modules are co-located next to each other in the FPGA, as shown in Figure 1. A floorplan showing the physical placement of the two modules is shown in Figure 2. While the VTA occupies multiple clock regions, the attacker's TDC can be placed in close proximity to the victim's VTA module, but not within the VTA circuitry due to the logic placement limitations of multi-tenancy. Because of the shared power distribution network within the FPGA, the TDC module is able to capture voltage traces as the VTA module executes. The traces are then used to extract machine learning model information.

TABLE I: Details of the tested neural networks.

Model	Total Layers	Layers Off-loaded to VTA	Num. Inst. Groups on VTA
ResNet-18 v1	18	16	307
MobileNet v1	28	26	210

The attack was tested on a Xilinx Zynq-7000 SoC ZC706 board (`xc7z045ffg900-2`). The VTA and TDC run on the same 120MHz clock. As described in Section IV, the TDC traces show a clear voltage drop when VTA computations begin. This drop can be used as a trigger for attacks, so strict synchronization of the two modules is not necessary. The TDC collects one measurement every five clock cycles, providing a sampling rate of 24Mhz.

A. Threat Model

The victim VTA executes machine learning inference operations, while the attacker attempts to steal information, such as model architecture and kernel sizes of each layer, that the victim desires to hide or protect. We assume that the victim and attacker are co-located on the same FPGA, but are logically isolated. As shown in Figure 1, the victim and attacker modules can communicate with their respective FPGA modules through a shared AXI crossbar. However, the shared crossbar is not used in the attack itself. All AXI communication is assumed to be secure, possibly encrypted, and we do not use AXI timing or contention as part of the attack. It is also assumed that the victim and the attacker are on the FPGA without other tenants, and share the underlying power distribution network (PDN). Thus, the attacker's goal is to observe voltage changes in the PDN, using a TDC module [9], [19], as the VTA executes different instructions.

B. Attacker TDC Sensor

The attacker uses a 256-stage TDC that contains an adjustable delay module, followed by a chain of *Carry4* lines used as the taps [20]. In the TDC, each measurement, a value between 0 and 256, records the delay of a circuit by observing how far through a tapped delay line a signal travels during a single measurement period. The delay is directly related to voltage: lower voltages causes the signals to propagate a shorter distance, and TDC outputs a smaller value. In the PDN, voltage drops occur in the vicinity of a target module, in this case the VTA, due to both resistive, IR, and inductive, $L\frac{di}{dt}$ voltage drop effects. [21]. We show that the VTA causes sufficient voltage drops during its operation for observation of its operation to be possible.

C. Machine Learning Models used in the Evaluation

In the evaluation we run the popular ResNet-18 v1 [22] and MobileNet v1 [23] machine learning models on the VTA. Table I shows the number of layer computations offloaded

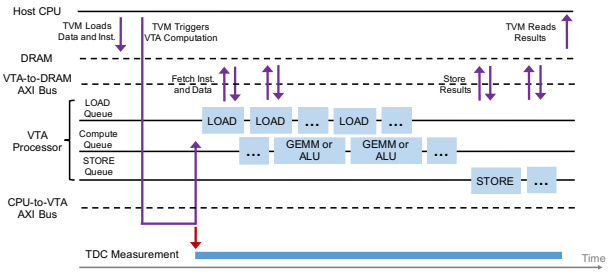


Fig. 3: Execution timeline of one VTA instruction group, which causes multiple VTA instructions to be executed. One instruction group corresponds to one AXI transaction group on the CPU-to-VTA AXI bus. The TDC is triggered at the start of the instruction group. Purple arrows show communication and commands sent on the different buses. Red arrow shows TDC measurement start.

to the VTA, and total number of VTA instruction groups used to realize the layer computations. Note that different layers are realized using different numbers of instruction groups. All of the networks were pre-trained on the ImageNet dataset [24] provided by MXNet Gluon Model Zoo [25]. The networks were used to perform inference operations on randomly selected input images as the TDC module captured power traces.

D. Unit Tests and Layers Tests used in the Evaluation

Prior to evaluating the full machine learning models, we used VTA unit tests and our own layer tests. Unit tests test the operation of specific VTA instructions. Layer tests test the operation of a group of instructions that implement one layer in a machine learning model. Three unit tests were used: GEMM, ALU-Add, and LOAD-and-STORE. TLPP is not used in the unit tests, allowing straightforward instruction behavior analysis. Layer tests simply correspond to each layer of the tested machine learning models, and have TLPP enabled.

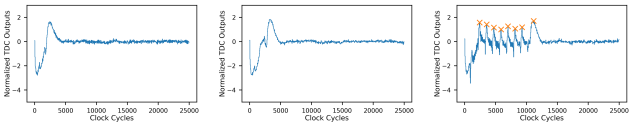
IV. Evaluation

Our evaluation focuses on understanding how to extract the information of machine learning models as VTA executes. Figure 3 shows an execution timeline of one instruction group on the VTA hardware. The on-FPGA computation is performed in groups of instructions, and different instructions within a group can be performed in parallel due to the use of instruction queues within the processor.

Each graph in Figures 4-7 shows an average TDC trace for one instruction group (either an instruction group of a unit test, or an instruction group at the start of different layers of the target neural network). The graphs in Figures 4-7 are each averages of 50 TDC traces of the same experimental settings, thus the attack requires $(50 \cdot N)$ executions of the neural network, where N is the number of instructions groups that are to be captured.

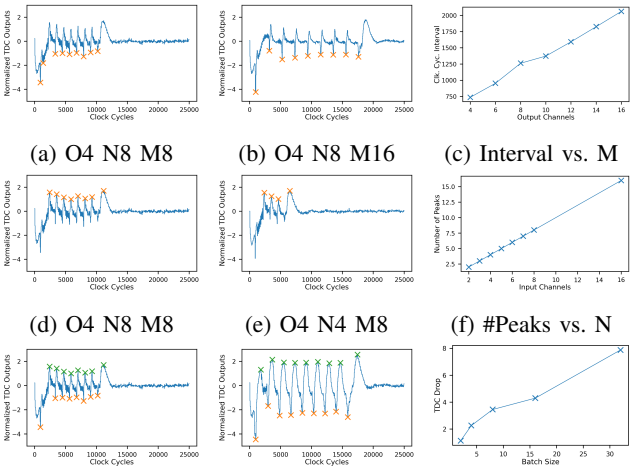
A. Sensitivity to Input Values

We first analyzed different instructions using the unit tests, and measured TDC traces for different input values (but for the same input sizes, e.g., same batch size and same number



(a) Load-and-Store (b) ALU-Add (c) GEMM

Fig. 4: Comparison of different unit tests, with batch size = 4, input channels = 8, and output channels = 8.



(a) O4 N8 M8 (b) O4 N8 M16 (c) Interval vs. M

(d) O4 N8 M8 (e) O4 N4 M8 (f) #Peaks vs. N

(g) O4 N8 M8 (h) O16 N8 M8 (i) Peak-Valley vs. O

Fig. 5: TDC traces used to recover the parameters of different `GEMM` instructions. It can be seen that the parameter of `GEMM` instructions can be reverse-engineered by calculating the interval between adjacent peaks, the number of peaks, and the drop depth from TDC trace.

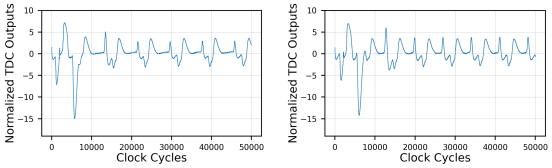
of input and output channels). We observed that different input values do not result in significant TDC measurement changes – while different sizes do, as we show in section IV-B. Thus the attacker is able to extract information about the neural network's architecture by taking measurements for many runs, while the victim VTA executes the same model with possibly different inputs.

B. Sensitivity to Instructions and Their Parameters

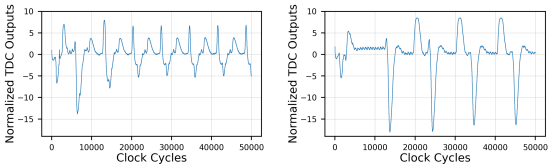
To distinguish different workloads running on VTA, we first collected TDC traces for the three unit tests: GEMM, ALU-Add, and Load-and-Store, using input data of the same size.[1] As shown in Figure 4, different unit tests, and thus different types of instructions, can be easily distinguished based on their unit test TDC trace waveforms.

We further analyzed the GEMM unit test traces with different data parameters for `GEMM` instructions. In GEMM unit tests, 'O', 'N', 'M' denote the number of input batches, input channels, and output channels respectively, which define the dimensions of the matrix [26]. As shown in Figures 5a, 5b, and 5c, the number of output channels is related to the interval between adjacent valleys, which is linearly correlated

[1]Note that the tests include data `LOAD` and `STORE` instructions needed to provide data for, e.g., `GEMM` or `ALU` instruction computations.

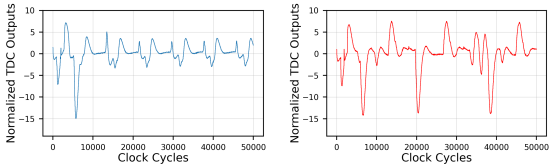


(a) (56, 64, 3), 4.72 (b) (28, 128, 3), 5.07

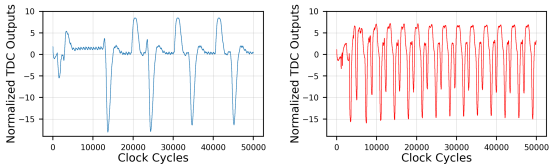


(c) (14, 256, 3), 7.88 (d) (7, 512, 3), 13.30

Fig. 6: The TDC traces for the first $50,000$ clock cycles of different instruction groups used to realize the convolution layers of ResNet-18 on VTA. A caption label, for example, "(56, 64, 3), 4.72" means the convolution output size is 56x56, with 64 3x3 filters, and the average peak-to-valley difference is 4.72.



(a) ResNet-18 Layer-2 (b) MobileNet Layer-2



(c) ResNet-18 Layer-17 (d) MobileNet Layer-27

Fig. 7: Comparison of TDC traces capturing the first $50,000$ clock cycles for different layers in ResNet-18 and MobileNet, showing clear differences for model recognition.

to 'M'. Figure 5c shows the relationship between intervals and 'M', for 'M' from 4 to 16 with the same 'O' and 'N'. Similarly, Figures 5d, 5e, and 5f show that 'N' can be recovered by counting the number of peaks, and Figures 5g, 5h, and 5i indicate that batch size mostly influences the difference between peak and valley. Given this knowledge, an attacker can analyze the TDC data to recover the approximate configuration for each `GEMM` instruction. This configuration information in turn can be used to recover information about the architecture of the neural network's layers.

C. Distinguishing Different Convolution Layers

For a convolution layer computation workload, input data is fed into `GEMM` instructions first before `ALU` computations. This property allows us to distinguish different convolution layers based on the recovered `GEMM` instruction's parameters. The TDC traces of four instruction groups corresponding to four common convolution layer parameters in ResNet-18 [22] are shown in Figure 6. The different convolution layer parameters can be distinguished by observing the patterns of peaks and valleys in the traces, even when a trace does not cover the whole instruction group's duration.

D. Distinguishing Different Machine Learning Models

Each neural network model includes several convolution layer parameters, e.g., ResNet-18 has 4 common layer parameters: (56x56, 3x3, 64), (28x28, 3x3, 128), (14x14, 3x3, 256), (7x7, 3x3, 512). Figure 7 shows the TDC traces for the first $50,000$ clock cycles of the first and last layers of ResNet-18 and MobileNet offloaded to VTA. Clear differences in the traces can be used to distinguish different models based on the traces of the different convolution layers. A set of traces can thus be compared to reference traces to distinguish a network, and approximate convolution layer parameters can be recovered by analyzing the peaks and valleys in the trace.

V. Related Work

Existing attacks for the recovery of neural network architectures on FPGAs generally require physical access, e.g., [3] and [4]. On ARM processors, attacks have leveraged electromagnetic emanations, e.g., [27]. There are also software attacks that recover neural network architectures by abusing the APIs [1] or analyzing outputs and confidence values [28].

We are aware of only three remote attacks targeting neural network algorithms on FPGAs in a multi-tenant setting. Boutros et al. [29] showed that voltage manipulations by an adversary co-tenant are unable to affect CNN inference accuracy due to model redundancy. They did not attempt to recover model information like we do. Moini et al. [17] were able to extract input images to a binarized neural network (BNN) by monitoring on-FPGA voltage fluctuations during convolution operations using a TDC. They targeted a custom, hard-coded BNN algorithm, and they assumed a known BNN architecture. Hua et al. [18] found the neural network algorithm's architecture (number of layers, type of each layer, and weight values) by feeding inputs to the accelerator and observing the resulting off-chip memory accesses. They require the control of inputs and the means to monitor the CPU to FPGA memory bus. Our attack does not require the control of inputs, is contained fully inside the FPGA (does not depend on ability to monitor external memory) and does not require knowledge of the neural network algorithms architecture as the architecture is what our attack aims to recover.

VI. Conclusion

This work presented the first remote power attack for the extraction of machine learning algorithm architectures in a multi-tenant FPGA setting. This attack targeted the Versatile Tensor Accelerator which supports many neural network algorithms, and is not a model-specific accelerator. Given the high value of the intellectual property contained in neural network algorithm architectures, this attack demonstrates the threats to intellectual property when multi-tenant FPGAs are used.

REFERENCES

[1] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium (USENIX Security)*, 2016, pp. 601–618.

[2] J. Wei, Y. Zhang, Z. Zhou, Z. Li, and M. A. A. Faruque, "Leaky DNN: Stealing deep-learning model secret with GPU context-switching side-channel," in *International Conference on Dependable Systems and Networks (DSN)*, 2020, pp. 125–137.

[3] A. Dubey, R. Cammarota, and A. Aysu, "MaskedNet: The first hardware inference engine aiming power side-channel protection," in *International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020, pp. 197–208.

[4] 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 *Annual Computer Security Applications Conference (ACSAC)*, 2018, pp. 393–406.

[5] Microsoft Azure, "Deploy ML models to field-programmable gate arrays (FPGAs) with Azure Machine Learning," https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-fpga-web-service, Accessed: 2021-01-21.

[6] Amazon Web Services, "Amazon EC2 F1 Instances," https://aws.amazon.com/ec2/instance-types/f1/, Accessed: 2021-01-21.

[7] A. Khawaja, J. Landgraf, R. Prakash, M. Wei, E. Schkufza, and C. J. Rossbach, "Sharing, protection, and compatibility for reconfigurable fabric with Amorphos," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018, pp. 107–127.

[8] J. M. Mbongue, A. Shuping, P. Bhowmik, and C. Bobda, "Architecture support for FPGA multi-tenancy in the cloud," in *International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2020, pp. 125–132.

[9] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," in *Design, Automation & Test in Europe (DATE)*, 2018, pp. 1111–1116.

[10] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *IEEE Symposium on Security and Privacy (S&P)*, 2018, pp. 229–244.

[11] C. Ramesh, S. Patil, S. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier, "FPGA side channel attacks without physical access," in *International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2018, pp. 45–52.

[12] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, "Leakier wires: Exploiting FPGA long wires for covert- and side-channel attacks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–29, Sep. 2019.

[13] S. Tian and J. Szefer, "Temporal thermal covert channels in cloud FPGAs," in *International Symposium on Field Programmable Gate Arrays (FPGA)*, 2019, pp. 298–303.

[14] T. Moreau, T. Chen, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy, "VTA: an open hardware-software stack for deep learning," *arXiv preprint arXiv:1807.04188*, 2018.

[15] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "TVM: An automated end-to-end optimizing compiler for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018, pp. 578–594.

[16] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin *et al.*, "A hardware–software blueprint for flexible deep learning specialization," *IEEE Micro*, vol. 39, no. 5, pp. 8–16, 2019.

[17] S. Moini, S. Tian, D. Holcomb, J. Szefer, and R. Tessier, "Remote power side-channel attacks on BNN accelerators in FPGAs," in *Design, Automation & Test in Europe (DATE)*, 2021.

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

[19] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *International Symposium on Field Programmable Gate Arrays (FPGA)*, 2013, pp. 101–104.

[20] S. Moini, X. Li, P. Stanwicks, G. Provelengios, W. Burleson, R. Tessier, and D. Holcomb, "Understanding and comparing the capabilities of on-chip voltage sensors against remote power attacks on FPGAs," in *International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020, pp. 941–944.

[21] D. R. Gnad, C. D. K. Nguyen, S. H. Gillani, and M. B. Tahoori, "Voltage-based covert channels in multi-tenant FPGAs." *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 1394, 2019.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[23] 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.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[25] Apache MXNet, "MXNet Gluon Model Zoo," https://mxnet.apache.org/versions/1.4.1/api/python/gluon/model_zoo.html#gluon-model-zoo, Accessed: 2021-01-15.

[26] VTA tutorial, "Simple Matrix Multiply," https://tvm.apache.org/docs/vta/tutorials/matrix_multiply.html, Accessed: 2021-01-15.

[27] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI-NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *USENIX Security Symposium (USENIX Security)*, 2019, pp. 515–532.

[28] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Asia Conference on Computer and Communications Security (Asia-CCS)*, 2017, pp. 506–519.

[29] A. Boutros, M. Hall, N. Papernot, and V. Betz, "Neighbors from Hell: Voltage attacks against deep learning accelerators on multi-tenant FPGAs," in *International Conference on Field-Programmable Technology (FPT)*, 2020.