2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH) | 978-1-6654-0959-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/NANOARCH53687.2021.9642246

Deep Neural Network Security From a Hardware Perspective

Tong Zhou, Yuheng Zhang, Shijin Duan, Yukui Luo, and Xiaolin Xu
Department of Electrical and Computer Engineering
Northeastern University, Boston, MA, USA
{zhou.tong1, zhang.yuhe, duan.s, luo.yuk, x.xu}@northeastern.edu

***Abstract*—Deep neural networks (DNNs) have been deployed on various computing platforms for acceleration, making the hardware security of DNNs an emerging concern. Several attacking methods related to the hardware accelerator of DNN have been introduced, which either affect the DNN inference accuracy or leak the privacy of DNN architectures and parameters. To provide a generic understanding of this emerging research area, in this survey, we systematically review the recent research progress of DNN security from a hardware perspective. Specially, we discuss the existing hardware-oriented attacks targeting different DNN acceleration platforms, and point out the potential vulnerabilities.**

I. Introduction

Deep Neural Networks (DNNs) have achieved great success in various real-world applications, such as face recognition [1], self-driving cars [2], and medical image analysis [3]. However, recent studies have revealed different types of vulnerabilities that make the DNNs less reliable and secure. While most existing studies have been focused on exploring the DNN security from a software view, the hardware perspective has not been systematically explored. To mitigate this gap, in this survey paper, we investigate the security issues on DNNs from a hardware perspective. Specifically, we categorize the vulnerabilities and attacks associated with DNN based on different implementation platforms: CPU-only, CPU-GPU, and FPGA-based deployments, i.e., following a "controller-executor" manner. While the CPU-only or microcontroller (MCU) platforms are not the major acceleration platform for DNN training and inference, they still find important utilization in specific scenarios, such as serving the compressed DNN inference procedure in a low-power application. In contrast, providing high computational capability and throughput, the CPU-GPU platform has become the mainstream acceleration paradigms for DNN execution, in which the CPU serves as the controller and the GPU as the executor for single instruction multiple data (SIMD) parallelism computing [4]. In recent years, the FPGA-based platform has also been explored for the DNN acceleration purpose, as a promising alternative hardware acceleration platform [5]. In an FPGA, the users can customize their hardware processing engines and the instruction flow. Although the FPGA-based platform is more power-efficient than the CPU-GPU platform, it suffers from limited memory size due to the small Block RAM (BRAM) size. As a result, the FPGA-based accelerators usually require more frequent inter-chip data communication between the FPGA and peripheral memory, resulting in lower performance than GPU platform on parallel data processing [6].

When DNN models are executing on these hardware acceleration platforms, the adversaries can conduct two types of attacks, i.e., side-channel attack and fault injection attack, which lead to model extraction and inference accuracy reduction, respectively. Although the accurate model parameters, such as weights, have not been practically extracted by these existing hardware-oriented attacks, the coarse-grained DNN secrets (e.g., the architecture) are demonstrated as vulnerable to various side-channel attacks [7]–[10]. Differently, fault injection attack is the main strategy utilized to degrade the DNN inference accuracy, i.e., by accurately introducing faults to the execution of the hardware platforms [11], [12]. We summarize the threat models for different attacks in Table I.

The remainder of this paper is organized as follows. Sec. II briefly introduces the DNN structure and general DNN attacks. Then we discuss the vulnerabilities and attack strategies for DNN on the CPU/CPU-GPU platforms and CPU-FPGA platforms in Sec. III and Sec. IV, respectively. Sec. V concludes this paper and predicts potential research directions.

II. Background

A. Deep Neural Networks

A DNN is a neural network with multiple layers between the input layer and the output layer, which can accomplish complicated tasks like image classification, object detection, and semantic segmentation. The architectural parameters of a DNN mainly includes the number of input channels, output channels, and hidden layers, as well as the kernel size and the type of hidden layers (e.g., convolutional layers). Additionally, another group of DNN parameters often refer to the weights and biases, and the values of which are learned during the DNN training. The trained DNNs can used to make predictions for unseen data (i.e., the test samples), which is known as the model inference.

In the following discussion, we take Convolutional neural network (CNN) as the representative victim DNN model. As a typical DNN model class, CNNs are widely used for image processing. Depending on the layer connection-ship and model complexity, CNNs can be classified as sequential, such as LeNet [13], or more complicated architecture that involves an extra connection between layers, like the residual connection in ResNet [14]. A CNN model consists of millions of parameters,

TABLE I: Threat models for different attacks: (1) Device physical access; (2) System privilege; (3) Specialized inputs; (4) Edge system applicable; (5) Cloud system applicable; (6) Shared resource.

			(1)	(2)	(3)	(4)	(5)	(6)
Traditional platforms (CPU-only and CPU-GPU)	Side-channel attack	Electromagnetic (EM) trace reverse engineering	✓		✓	✓		
		PCIe traffic analysis	✓		✓	✓		
		Memory access analysis		✓		✓	✓	
		Context-switching penalty		✓	✓	✓	✓	✓
	Fault injection attack	Rowhammer		✓		✓	✓	✓
		Voltage/frequency scaling (VFS) based attack		✓		✓	✓	
FPGA-based platforms (Edge or Cloud FPGA)	Side-channel attack	EM trace reverse engineering	✓		✓	✓		
		Time-to-digital convter (TDC) trace					✓	✓
	Fault injection attack	Power distribution network (PDN) based attack					✓	✓
		PDN-based attacks with TDC guidance					✓	✓

thus involves intensive computing in the inference. To save the storage and computational overhead, the binary neural network (BNN) is designed by replacing full-precision weights of CNNs with binary weight values, but the model performance is correspondingly sacrificed.

B. DNN Adversarial Attacks

Previous studies have proved that DNNs are vulnerable to adversarial attacks, which target at either degrading the model inference accuracy or extracting the model secrets like architecture. Common strategies used in these adversarial attacks include introducing perturbations or injecting faults to DNN models.

The most representative way to add perturbations is called adversarial example attack, which manipulates the original input image of the victim DNN model, as introduced by Szegedy *et al.* in [15]. Specifically, to make such attacks more stealthy, the perturbations should be carefully designed to be imperceptible for human eyes but capable of misleading the DNN model inference results. Multiple methods have been proposed to create adversarial examples, such as the Fast Gradient Sign Method (FSGM) [16]. As a well-known gradient-based method to craft the perturbation, FSGM modifies the input image along the direction of gradient of the loss function with respect to the clean input, which causes overall accuracy degradation. Compared to the previous work [15], this method only performs one-step update, thus is more time-efficient.

One limit of the existing attacks is that they are all conducted in a data-dependent manner, i.e., unique perturbation needs to be specifically designed for each image. To address this issue, Universal Adversarial Perturbations (UAP) [17] was later proposed, which leverages the DeepFool method [18] to obtain a small perturbation of each sample then updates it to the overall perturbation.

In practical scenarios, the adversaries may not have direct access to feed the image to the trained model. Therefore, some studies have explored physical attacks. One example is [19], in which the authors showed that by wearing a pair of eyeglass frames, the attackers can be recognized as another person by a state-of-the-art face recognition algorithm. Another study [20] conducted by Eykholt *et al.* provides a different physical attack example, in which a stop sign can be misclassified as another traffic sign (e.g., speed limit sign) after putting some stickers on it.

Adversarial example attack is not only used to conduct misclassification, but also leveraged to extract DNN models (i.e., the model architectures or parameters). Such model extraction attacks can be further classified into two categories. One is the query-based attack, and the other is the side-channel-based attack, which is detailed in Sec. III-A and Sec. IV-A. Yu *et al.* provides an example of the former type using adversarial examples [21], which first generates unlabeled adversarial examples as a synthetic dataset, and then use them to query the victim model. After labeling the adversarial examples according to the output of the victim model, the adversary can train the local substitute model with labeled data, then use the trained model for predictions, which is expected to match the performance of the victim model. This method significantly reduces the number of queries compared to other query-based methods [22]–[25].

Although the input-based attacks are effective on the software implementation, they are not stealthy enough, as discussed in several defense strategies [26], [27]. On the other hand, the DNN attacks from hardware perspectives are more focused on modifying the model parameters to achieve malicious goals. As a result, such attacks are more stealthy and difficult to be eliminated. Considering that DNNs are deployed on hardware, such as CPU, GPU and FPGA, the model parameters will correspondingly be stored in the memory of hardware. Therefore, the parameter-modification-based attacks become possible via hardware vulnerabilities like *Rowhammer* [28]. For example, Liu *et al.* proposed two parameter-based methods, i.e., the Single Bias Attack (SBA) and the Gradient Descent Attack (GDA) in [29] to perform untargeted attack and targeted attack, respectively. To increase the stealthiness, they applied gradient descent method with Layer-searching and Modification Compression techniques to minimize the weight modification. Recently, a more stealthy parameters-based attack was proposed by Rakin *et al.* [30], which is a fine-grained attack that aims to flip the weight bits of DNNs to reduce the model inference accuracy. Unlike the previous work [31] that only discusses performing bit flip on full-precision models, Bit-Flip Attack (BFA) can work on quantized models. To conduct an efficient attack, BFA proposes a progressive bit search algorithm

comprising of gradient ranking and progressive search to locate the most vulnerable bits, then performs bit-flipping only on the selected bits.

III. Attack on Traditional DNN Execution Platforms

As the mainstream device to support the model training process and inference, the CPU-GPU platform can mitigate the performance issues of the CPU-only platform. In the current usage, the CPU-only platforms are commonly utilized in the training and inference of DNNs that have fewer parameters. Therefore, without loss of generality, in this section, we also cover some attacks targeted the CPU-only platform.

A. Side-Channel Attack

Side-channel attacks exploit the indirect measurements of intermediate computation to cause the leakage of secret information, with which the adversaries can apply reverse engineering to achieve malicious goals like model extraction. There are several side channels that can be exploited on the CPU-only or CPU-GPU platforms.

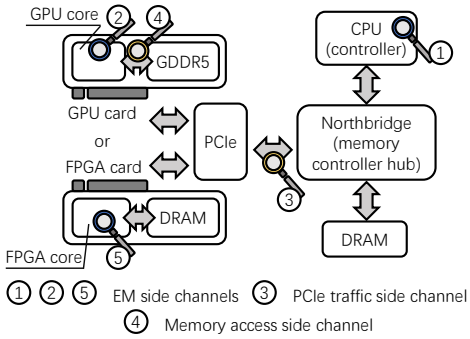

Fig. 1: Side channels in different components.

1) Electromagnetic (EM) Side Channel: By exploiting the fact that all electronic devices emit EM radiation, an adversary can collect EM traces and perform signal analysis on it to reveal information. Batina *et al.* first performed the EM-based reverse engineering on the multilayer perceptron (MLP) and DNNs executed on an ARM Cortex-M3 microcontroller [32]. Specifically, they analyzed the timing behavior observed on the EM trace (Fig. 1 ①) for the different activation functions with random inputs in the range of [-2,2], demonstrating that the timing behavior of each activation function is distinguishable. This method can thus be used to extract the activation functions used in the victim DNNs. Besides, the number of hidden layers can also be recovered by leveraging the EM trace. Similarly, an EM side-channel attack (Fig. 1 ②) can be launched on the advanced system-on-module (SoM) [33], which is constructed by CPU-GPU architecture in one board (e.g., Nvidia Jetson Nano).

2) PCIe-Based Probing: Multiple parts of the CPU-GPU platform could result in information leakage, such as the I/O buses, as shown in Fig. 1 ③. These I/O buses (e.g., GPU's PCIe bus), as the critical parts connecting individual functional components, can be compromised to reverse engineer the application execution details. In [7], Zhu *et al.* presented a two-phase PCIe snooping-based DNN reverse engineering, i.e., using the offline and online phases, named *Hermes Attack*. In details, the adversary of such an attack uses the Teledyne LeCroy Summit PCIe protocol analyzer [34] to intercept and log the bus events. Specifically, in the offline phase, the attacker builds a database with the GPU profiling information and PCIe traffic logging included. The primary purpose of this database is to construct the mappings between GPU kernels and DNN layer types, and between GPU kernels and the offset of architecture's configurations. To reconstruct the DNN architecture, the online phase utilizes the offline database associated with a command extractor, which is a filter to eliminate the raw PCIe traffic noise by checking the address continuity, as transmitted data is usually consecutive in DRAM.

3) GPU Memory Access Analysis: If an attacker can observe the GPU memory access trace through the GDDR5 memory bus (Fig. 1 ④), s/he can obtain the kernel read/write access volume and memory access traces. These information can help the attacker to extract a complex DNN architecture with high accuracy [8], which is based on the fundamental observation that only the feature map data (activation data) can introduce read-after-write (RAW) memory access patterns, especially for convergent and divergent layers. This fact combined with the read cache miss rate of the kernel can reconstruct the configuration of the architecture.

4) Context-Switching Penalty: While the aforementioned GPU attack methods mostly assume the attacker can acquire high-resolution side-channel information, such as PCIe traffic events and GDDR5 memory access traces, Wei *et al.* used a more practical side channel, i.e., context-switching penalties, to extract fine-grained DNN secrets [9]. Besides the basic premise that the victim and attacker can run their kernels on the same GPU, the threat model of this work is also assumed that the attacker has the privilege to access the victim kernel resource usage by sampling the CUDA Profiling Tools Interface (CUPTI) [35]. During execution, the context-switching occurs when a kernel is changed to another, since the contexts of different kernels on GPU cannot be shared. Specifically, they demonstrated that different operations (corresponding to various kernels) would induce different switching time overheads. Without the multi-process service (MPS) enabled, these time overheads, namely context-switching penalty, could be significant when the switching actions execute, such as switching data access. Motivated by this vulnerability on GPU, they proposed an attack framework to extract secrets of victim applications. As a validation, they successfully recovered the structural secrets of a DNN model with high accuracy, including operation sequence, layer hyper-parameters, and optimizer.

B. Fault Injection Attack

Unlike side-channel attacks that are non-invasive and have no impact on the DNN outputs, fault injection attacks will partially change the internal states of DNN executions to crash the expected output. To keep the fault injection as stealthy

as possible, attackers usually aim to minimize the number of injected faults without compromising the attack effect. Moreover, since fault injection attacks require more accurate operations and attack positions, they have more strict conditions in the threat models, such as gray-box or even white-box attacks. Two of the most critical attack methods are selected to indicate the vulnerability on the GPU platforms.

1) Rowhammer in DRAM: One stealthy but critical vulnerability in current DRAM devices is that some stored data is vulnerable to crafted data disturbance, i.e., continuously accessing data in the same area (like one row) in DRAM could flip the DNN model data [28], where the corresponding attack mechanism is called *Rowhammer*. One explanation of this disturbance error is that the frequent charge toggling will have cross-talk to nearby memory cells. For example, if one row in DRAM is accessed very often, the word-line voltage of that row must be boosted frequently. This voltage fluctuation will accelerate the charge leakage on adjacent rows; if some memory cells lose too much charge before refreshing, the data could be changed and disturbance error occurs, which is shown in Fig. 2. As many DRAM devices are proved vulnerable to the *Rowhammer* attack [28], [36], [37], many methodologies have been proposed to mount the *Rowhammer* under various scenarios, such as [38], from inducing severe result faults to recovering application secrets.

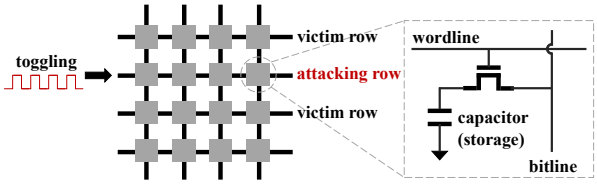

Fig. 2: Illustration of rowhammer attack on DRAM.

Recently, Yao *et al.* proposed a comprehensive hardware-based attack framework on DRAM for quantized DNN, namely *DeepHammer* [11]. To successfully apply the *DeepHammer* attack, the attacker are assumed to have the knowledge of all DNN model parameters and the deployment details on the DRAM; further, s/he is able to co-locate attacks with the DNN application on certain mapped pages and decide when to trigger the bit-flipping attack, which can be categorized as a white-box approach. *DeepHammer* is composed of three steps: memory templating, flip-aware bit searching, and bit-flipping *Rowhammer*. After the memory templating profiling all physical pages in targeted DRAM with vulnerable bit offset information, flip-aware bit searching locates the bit-flipping combination to achieve low inference accuracy with as few to-be-flipped bits as possible. With one or multiple chains of bits to be flipped, the designed memory-efficient *Rowhammer* will accurately locate memory page and perform desired bit flips. To evaluate the effectiveness of *DeepHammer* attack, Yao *et al.* implemented it on real GPU systems with 11 DNN architectures and 4 datasets. The results show that *DeepHammer* can make the inference accuracy of DNNs as poor as near-random guesses with just a few bits flipped.

2) Voltag/Frequency Scaling (VFS) Based Attack: VFS is a newly proposed strategy that could be exploited to deploy fault injection attacks on the GPU platform. The power management module on GPU can adaptively adjust the voltage and frequency based on the execution condition. While the dynamic voltage/frequency scaling (DVFS) uses preset discrete configurations on voltage and frequency, the adaptive voltage/frequency scaling (AVFS) adaptively adjust voltage and frequency at real-time, to run the workload in a lower-power or a higher-performance mode. Hence, AVFS is adopted in current GPU platforms, such as AMD Polaris GPU [39]. However, the AVFS will only trigger the adjustment when the voltage drop is larger than 2.5% and only reduce the clock frequency up to 20% [40], which means if the kernel workload is heavy but within 2.5% voltage drop (*undervolting* data corruption), or the kernel has such a high frequency that the 20% frequency reduction is not enough to ensure fault-free execution (*overclocking* data corruption), the faulty execution might be incurred. The faulty scenario will be more obvious when combining these two conditions, called *overdrive* data corruption, which could be utilized to inject faults during execution, extract secrets, or even conduct Denial-of-Service (DoS) attack.

Sabbagh *et al.* first explored the VFS-based fault attack on model inference by introducing silent data corruption (SDC) on a CNN model [41] to achieve misclassification. The threat model of this attack is more practical, i.e., the attacker only needs to obtain the privilege of VFS control and reside on the victim-used GPU. Specifically, they used AVFS [39] to craft fault configuration on *overdrive operation performance points (overdrive OPPs)*, which makes the GPU executed with certain low voltage and high frequency to cause SDC without hang/crash. As demonstrated, the pre-delay time (the delay before activating the *overdrive OPPs*) have significant influence on the injected fault position and fault data values. With these observations, they constructed a fault-injecting model that controls fault injections on certain kernel executions. However, this work only applied the proposed strategy to a certain AMD GPU and two applications, thus its applicability on other commercial GPUs and security-sensitive applications still needs to be explored.

IV. Attack on Edge/Cloud FPGA-based Platform

Apart from traditional DNN execution platforms, the FPGA-based platform recently attracts much attention thanks to its configurability. However, there are several vulnerabilities existing in FPGA that can be exploited to conduct malicious attacks [10], [12], [42]–[44]. In this survey, we mainly focus on attacks on multi-tenant FPGAs, which allow multiple users to reside on the same FPGA to improve resource utilization. Due to many hardware resources are shared across tenants in a multi-tenant FPGA system, e.g., power distribution network (PDN), a malicious tenant may take advantage of this indirect interaction to launch a variety of novel attacks.

Similar to the case of traditional platforms, side-channel attacks and fault injection attacks are the two main categories

of attacks against multi-tenant FPGAs. We also introduce the EM-based side-channel attack, which can only be used on edge devices, as a complementary of attacks on FPGA.

A. Side-Channel Attack

1) EM Side Channel: Utilizing the fact that the execution time is proportional to the parameters of each layer, Yu *et al.* analyzed the the number of parameters of different layers, including convolutional layers, pooling layers, and fully-connected layers, and showed that different layers could result in different EM emission patterns [42]. Thus, they demonstrated that the depth of neural networks and the type of each layer can be extracted by observing EM traces. Once these important information is extracted, an attacker can generate a few candidate architectures with different layer dimensions and filter sizes by following the relationship between the hyper-parameters of each layer, then select the best architecture that outperforms others on the same test set as the substitute network. To evaluate the effectiveness of this method, Yu *et al.* implemented BNN classifiers on the Pynq-Z1 board and successfully built the substitute network that achieved the similar performance as the victim model.

2) Time-to-Digital Converter (TDC) Side Channel: TDC is a customized circuit that can be deployed on FPGAs, to convert the propagation delay change on a look-up-table (LUT) delay chain into a digital value. Using a TDC, a remote power attack can be conducted to determine the detail of the DNN architecture, which is implemented on the FPGA-based Versatile Tensor Accelerator (VTA) [10]. In a multi-tenant FPGA, the TDC can capture the voltage fluctuations while a DNN is running on VTA because of the shared PDN. The proposed method uses TDC-captured voltage traces as the side channel, to recover the parameters of General Matrix Multiplication (GEMM) instructions, then extracts parameters of convolutional layers by analyzing peaks and valleys in the trace. Specifically, by observing TDC traces, Tian *et al.* found that the number of input batches is linearly correlated to the difference between peaks and valleys, and the linearity also exists in the number of output channels and the interval between adjacent valleys. Besides, the number of input channels can be inferred by counting the number of peaks. With this information, the model structure can be successfully extracted.

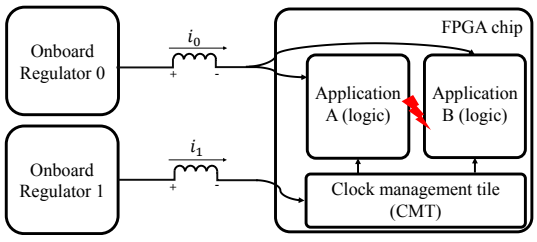

Fig. 3: Power Distribution Network in multi-tenant FPGA cloud-platform.

B. Fault Injection Attack

1) PDN-Based Attack: Adversarial weight duplication (AWD) attack is a new type of fault injection on FPGA that takes advantage of the co-tenancy when multiple tenants are on the FPGA [12]. The malicious tenant can indirectly affect other tenants by interfering the PDN as shown in Fig. 3 and cause the FPGA overloaded. AWD attack injects faults on the data transmission between the off-chip memory and the on-chip buffer, which incurs duplicated weight values on the receiver side. Such weight duplication occurs when the power exceeds the maximum capability that the PDN can supply to the processing engine (PE), while the integrity of the on-chip clock management tile (CMT) highly relies on the separate power supply provided by the PE. Therefore, the influenced PDN compromises the CMT integrity, causing the timing violation of DNN execution during the transmission process.

2) PDN-Based Attack with TDC Guidance: DeepStrike is proposed to lower the DNN inference accuracy by exploiting the advantages of both TDC-based side-channel attacks and PDN-based fault injection attacks [43]. By using the TDC to analyze the voltage fluctuation of the DNN accelerator, the attacker can accurately control the timing to activate a malicious circuit, which can be used to overload the PDN and inject faults. The attacker constructs a specialized attack scheme based on voltage traces of the victim to launch the *DeepStrike* attack efficiently with low hardware overhead. Specifically, the *DeepStrike* guides the PDN-based fault injected on the computing time of PEs, affecting the engine generating duplicated intermediate values. Furthermore, the author also investigates different DNN model layer configurations with different resilience against the PDN-based attack.

V. Summary and Discussion

This survey investigates the emerging security concerns of deep neural networks (DNNs) from a hardware perspective. Compared with the software-oriented attacks against DNN, the hardware attacking strategies mainly utilize the internal weakness of the acceleration platforms. Specifically, we summarize the existing vulnerabilities of DNN applications on CPU-only, CPU-GPU, and FPGA-based platforms, which are followed by the corresponding attack strategies. These attacks can achieve malicious objectives like model extraction and inference accuracy reduction, though some of them are more difficult to be implemented and require extra knowledge, such as the detailed deployment information of the DNNs. Hardware-based DNN attacks are also deriving various forms to adapt to different implementing scenarios, which calls for studies on more general and practical defense strategies.

As for the future research directions of the DNN security from a hardware perspective, we envision that fine-grained weight extraction is one of the challenging tasks using hardware attacking methods. Moreover, it is meaningful to explore whether a DNN model is still vulnerable to fault injection or side-channel attacks, if the attacker only has limited privilege, i.e., s/he cannot pre-obtain the configuration of the targeted application.

REFERENCES

[1] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical image analysis using convolutional neural networks: a review," *Journal of medical systems*, vol. 42, no. 11, pp. 1–13, 2018.

[4] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler, "vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–13.

[5] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 26–35.

[6] J. Cong, Z. Fang, M. Lo, H. Wang, J. Xu, and S. Zhang, "Understanding performance differences of fpgas and gpus," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2018, pp. 93–96.

[7] Y. Zhu, Y. Cheng, H. Zhou, and Y. Lu, "Hermes attack: Steal {DNN} models with lossless inference accuracy," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[8] M. Inkawhich, Y. Chen, and H. Li, "Snooping attacks on deep reinforcement learning," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 557–565.

[9] J. Wei, Y. Zhang, Z. Zhou, Z. Li, and M. A. Al Faruque, "Leaky dnn: Stealing deep-learning model secret with gpu context-switching side-channel," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 125–137.

[10] S. Tian, S. Moini, A. Wolnikowski, D. Holcomb, R. Tessier, and J. Szefer, "Remote power attacks on the versatile tensor accelerator in multi-tenant fpgas," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2021, pp. 242–246.

[11] F. Yao, A. S. Rakin, and D. Fan, "Deephammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *29th {USENIX} Security Symposium ({USENIX} Security)*, 2020.

[12] A. S. Rakin, Y. Luo, X. Xu, and D. Fan, "Deep-dup: An adversarial weight duplication attack framework to crush deep neural network in multi-tenant fpga," *arXiv preprint arXiv:2011.03006*, 2020.

[13] 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, 1998.

[14] 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*, 2016, pp. 770–778.

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

[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[17] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1765–1773.

[18] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[19] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 acm sigsac conference on computer and communications security*, 2016, pp. 1528–1540.

[20] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.

[21] 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 *NDSS*, 2020.

[22] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[23] Y. Shi, Y. Sagduyu, and A. Grushin, "How to steal a machine learning classifier with deep learning," in *2017 IEEE International symposium on technologies for homeland security (HST)*. IEEE, 2017, pp. 1–5.

[24] B. Wang and N. Z. Gong, "Stealing hyperparameters in machine learning," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 36–52.

[25] A. Dmitrenko *et al.*, "Dnn model extraction attacks using prediction interfaces," 2018.

[26] C. Lyu, K. Huang, and H.-N. Liang, "A unified gradient regularization family for adversarial examples," in *2015 IEEE international conference on data mining*. IEEE, 2015, pp. 301–309.

[27] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 582–597.

[28] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 361–372, 2014.

[29] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 131–138.

[30] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1211–1220.

[31] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2204–2206.

[32] 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)*, 2019, pp. 515–532.

[33] Ł. Chmielewski and L. Weissbart, "On reverse engineering neural network implementation on gpu," in *International Conference on Applied Cryptography and Network Security*. Springer, 2021, pp. 96–113.

[34] "Teledyne lecroy. summit analyzer," 2012, https://teledynelecroy.com/protocolanalyzer/pci-express.

[35] Nvidia. (2019) Cupti cuda toolkit documentation. [Online]. Available: https://docs.nvidia.com/cuda/cupti/index.html

[36] L. Cojocar, K. Razavi, C. Giuffrida, and H. Bos, "Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 55–71.

[37] V. Van Der Veen, Y. Fratantonio, M. Lindorfer, D. Gruss, C. Maurice, G. Vigna, H. Bos, K. Razavi, and C. Giuffrida, "Drammer: Deterministic rowhammer attacks on mobile platforms," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1675–1689.

[38] M. Seaborn and T. Dullien, "Exploiting the dram rowhammer bug to gain kernel privileges," *Black Hat*, vol. 15, p. 71, 2015.

[39] "Dissecting the polaris architecture," AMD, Tech. Rep., May 2016.

[40] M. Sabbagh, Y. Fei, and D. Kaeli, "A novel gpu overdrive fault attack," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[41] ——, "Overdrive fault attacks on gpus," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 2021, pp. 68–69.

[42] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "Deepem: Deep neural networks model recovery through em side-channel information leakage," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2020, pp. 209–218.

[43] Y. Luo, C. Gongye, Y. Fei, and X. Xu, "Deepstrike: Remotely-guided fault injection attacks on dnn accelerator in cloud-fpga," *arXiv preprint arXiv:2105.09453*, 2021.

[44] Y. Luo, C. Gongye, S. Ren, Y. Fei, and X. Xu, "Stealthy-shutdown: Practical remote power attacks in multi-tenant fpgas," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 545–552.