Security of Neural Networks from Hardware Perspective: A Survey and Beyond

Qian Xu
qxu1234@umd.edu
University of Maryland
College Park, Maryland

Md Tanvir Arafin
mdtanvir.arafin@morgan.edu
Morgan State University
Baltimore, Maryland

Gang Qu
gangqu@umd.edu
University of Maryland
College Park, Maryland

ABSTRACT

Recent advances in neural networks (NNs) and their applications in deep learning techniques have made the security aspects of NNs an important and timely topic for fundamental research. In this paper, we survey the security challenges and opportunities in the computing hardware used in implementing deep neural networks (DNN). First, we explore the hardware attack surfaces for DNN. Then, we report the current state-of-the-art hardware-based attacks on DNN with focus on hardware Trojan insertion, fault injection, and side-channel analysis. Next, we discuss the recent development on detecting these hardware-oriented attacks and the corresponding countermeasures. We also study the application of secure enclaves for the trusted execution of NN-based algorithms. Finally, we consider the emerging topic of intellectual property protection for deep learning systems. Based on our study, we find ample opportunities for hardware based research to secure the next generation of DNN-based artificial intelligence and machine learning platforms.

CCS CONCEPTS

• **Security and privacy → Security in hardware**; **Malicious design modifications**; **Side-channel analysis and countermeasures**; • **Computing methodologies → Neural networks**.

KEYWORDS

neural networks, hardware security, side-channel attacks, hardware trojan, fault injection attack, trusted execution environment

ACM Reference Format:
Qian Xu Md Tanvir Arafin Gang Qu. 2021. Security of Neural Networks from Hardware Perspective: A Survey and Beyond. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3394885.3431639

1 INTRODUCTION

Deep neural network (DNN) based computing methodologies have ignited tremendous improvement in solving long-standing problems in the artificial intelligence (AI) and machine learning (ML) domain [21]. Superhuman performance of DNN models for tasks such as object recognition, gaming, and natural language processing has poised deep learning (DL) solutions critical for the next generation of autonomous systems. Although DNN-based designs enjoy great success in solving complex problems, detailed security analysis for trusted and explainable DL based algorithms, applications, and systems are still under development.

Deep learning techniques' success can largely be attributed to the rise of graphical processing units (GPUs), tensor processing units (TPUs), and their application in tackling data-intensive computational workloads. GPUs and TPUs are designed on simple but massively parallel security oblivious architectures. DNN model development requires a significant amount of physical resources. For example, a recent NLP solution, Generative Pre-trained Transformer 3(GPT-3), has an estimated cost of $5 million for GPU hardware alone [22]. Moreover, newer models require an extensive amount of manually tagged training and testing dataset, which incurs additional development costs. Thus, security issues such as intellectual property (IP) protection, trusted model training and execution, verifiable and explainable computation are of paramount importance for successfully integrating DNNs in security-critical learning tasks.

Hardware security, a sub-domain of computer security research, studies vulnerabilities of the computing hardware. Hardware-oriented attack techniques such as fault injection, side-channel analysis, and Trojans have exposed critical vulnerabilities in deep learning hardware platforms. Hence, in this work, we focus on the following objectives to examine the current state of security for neural networks from a hardware perspective.

- We review hardware-based attack techniques that aim to steal or corrupt DNN models;
- We report current research progress in detecting hardware-based attacks and the related countermeasures;
- Based on our analysis, we propose future directions in hardware security research for neural network-based computing paradigms.

We find that the security problems in deep learning are still underdeveloped and require broader attention from the hardware security research community.

2 BACKGROUND AND MOTIVATION

Deep learning algorithms utilize stacked layers of neural networks for solving supervised and unsupervised machine learning tasks. Input data is fed into the system using the input layer, and the results are obtained from the output layer. There can be a large number of hidden/intermediate layers between the input and output layers. The architecture of these hidden layers depends on the task. For

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASPDAC '21, January 18–21, 2021, Tokyo, Japan
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431639

example, convolutional neural networks (CNNs), standard in image processing algorithms, contains convolutional layers, fully connected layers, pooling layers, loss layers, and activation layers. On the other hand, recurrent neural networks used in natural language processing tasks contains gated recurrent units (GRUs), long short term memories (LSTMs) *etc.* that implements temporal dynamic behavior. Thus, the architecture and the connection parameters (*i.e.,* weights and biases) defines the complete neural network.

A neural network is trained via supervised or unsupervised techniques. For example, stochastic gradient descent (SGD) combined with back-propagation has been very successful for the supervised training of a neural network [21]. These algorithms are fundamentally based on parallel matrix/tensor computation such as generalized matrix multiplication (GEMM) and block fused multiply-add (FMA). Typical CPU architecture is not efficient in matrix computation. Therefore, specialized hardware such as GPUs and TPUs, containing optimized scalar, vector, matrix multiplication units (MXUs) *etc.*, are employed in model training and inference tasks. Thus, vulnerabilities from the hardware stack for these components can compromise the security of the deep learning system.

Deep neural networks suffer vulnerabilities on both hardware and software stack. Software-side attacks such as model extraction, model inversion, membership inference, and poisoning exploit the weaknesses in the algorithms, libraries, and tools utilized in designing DNNs [12]. Moreover, adversarial examples can fool the network to predict wrong labels or infer invalid decisions [20]. Thus, adversarial attacks on object detection in autonomous driving, text processing in translation, or DL-based malware detection algorithms can cause severe damage in security-critical systems. As a result, software security for DL algorithms has received significant research attention over the last few years.

In contrast, hardware security research for DL platforms is still under development. Architecture and designs for the new generations of GPUs and TPUs are refreshed continuously to gain performance and power-efficiency. This creates a moving target for hardware attacks. Additionally, hardware-oriented approaches target simpler architecture and fundamental blocks, which does not reflect the efficacy of such attacks in field-deployed neural networks with hundreds of layers with millions of parameters. However, recent progress in hardware-oriented attacks and defenses has demonstrated the promises of these approaches in compromising DL-based AI/ML platforms' security.

An attacker can target several physical layer weaknesses in a DNN. An example of hardware attack surfaces on a modern object detection platform YOLO is shown in Fig 1. For hardware-based attacks, the goals of an attacker are listed below.

- **Model Corruption** Compromise the model parameters stored in memory so that the model fails in all of the tasks;
- **Backdoor Insertion** Alter the model stored in memory so that it fails on a subset of tasks;
- **Model Extraction** Extract the model from the device during run-time or via proving non-volatile memory;
- **Spoofing** Corrupt the input data by changing in the environment or the input sensors;
- **Information Extraction** Infer model information from the physical side-channels;

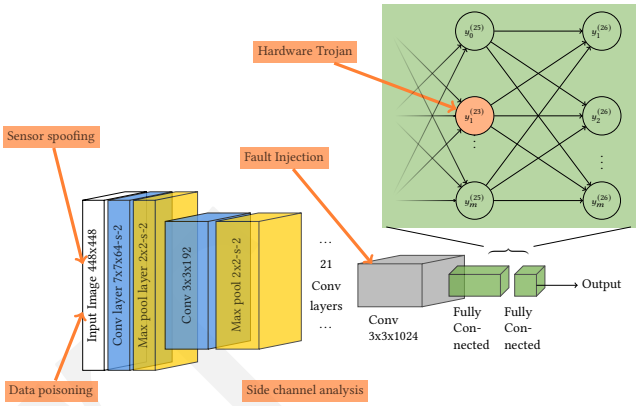

Figure 1: Architecture of YOLO, a common object detection framework [38]. The architecture is composed of 24 convolutional layers, followed by two fully connected layers. Example attacks on different components of this DNN are marked in myorange.

- **Sybil Attack** Introduce fake devices in a collaborative learning platform so that the training leads to invalid models

3 APPROACH

Once the attack goals has been identified, they can be achieved by several hardware-based approaches. The most common techniques include hardware trojan, side-channel attacks and fault injection attacks, etc.

3.1 Hardware Trojan

Trojan attacks to a neural network make malicious modifications such that wrong outputs would only be generated given deliberately designed inputs. Trojan could be added either by software or hardware. Software trojan is usually embedded into the neural network parameters during the training process by poisoning the data, causing a back door that only shows its effectiveness given inputs containing trojan triggers. Hardware trojan, on the other hand, is inserted into the circuitry of the integrated circuit by malicious designers or fabrication providers. Hardware trojan is usually carefully designed with a trigger and the corresponding payload for compromising the neural network's integrity.

The hardware trojan trigger can be classified into three different types: bit trigger, combinational trigger, and sequential trigger. Bit trigger adds a hidden extra pin to the circuit for controlling the switch between normal and triggered mode [24]. Combinational trigger focuses on the current input fed into the compromised neural network. The trigger could be the whole input [16, 26, 53, 54], partial input such as image pixels [50], and even intermediate computation result affected by the input [5, 6, 34]. The sequential trigger can be activated by a sequence of inputs with attacker-designed attributes, for example, a particular order of labels [32]. Overall, an extra pin or hidden information inside inputs could be utilized as hardware trojan triggers.

Regarding the payload location, there are four components in the neural network that can be compromised: input, computation

Table 1: Privacy breach from side-channel attacks

side-channel attack	model		input
	architecture	parameters	
cache	Y [13, 48]	N	N
memory-access pattern	Y [15, 18, 30]	Y [18]	N
EM/power	Y [2, 46]	Y [46, 51]	Y [2, 45]
timing	Y [10]	N	Y [7]
GPU	Y [33, 44]	N	N

block, intermediate data, and output. The approaches used to compromise inputs include providing illegal input data stored separately from the legitimate ones [34] and resetting input data back to zero [16, 53]. Some payloads involve modifying the computation block either by pruning multiply-accumulate (MAC) [24] or by adding gates to modify activation function [6]. Compromising intermediate computation result can be achieved through reduced bit width [26] and introduced perturbations [5]. To directly contaminate the output, targeted wrong labels are usually hidden in the triggers, and a multiplexer is utilized to make modifications [32, 50]. To conclude, the payload can be inserted into each component in the neural network chain.

3.2 Side-Channel Attack

A side-channel attack on a neural network utilizes the physical information produced by implementing computer systems to compromise privacy. Some examples of physical information that is likely to be overheard by attackers are cache information, memory-access pattern, timing information, electromagnetic leaks, power consumption, platform-specific leaks, etc. With such information, the attacker could deduce the inputs during inference and the neural network's architecture and parameters.

Some cache-based attacks utilize the property that the victim process and the attack process share the same instruction cache to steal sensitive information. In [48], GEMM functions are monitored using traditional side-channel techniques like Prime+Probe and Flush+Reload for inferring the network architecture. Later in [13], more target functions that reflect the network architecture are observed, and a meta-model is trained to learn this relationship and predict victim network architecture. Overall, based on the current literature, cache-based attacks can successfully infer the architecture of a neural network running on a CPU platform, but it fails to show its effectiveness on other platforms and its capacitance to infer others sensitive information like inputs or parameters.

While the cache-based attack relies on memory sharing and code access, side-channel attacks based on memory-access patterns compromise the model privacy through the memory bus. During memory bus snooping, an attacker could obtain the number of times the victim network accesses the memory, the memory address, copy size, and a timestamp for each access. With the help of this information, the attacker can reconstruct the network architecture [15, 18, 30] and get partial information about network parameters by figuring out the relationship between weight and bias under the dynamic zero pruning scheme [18]. To conclude, memory-based pattern leakage is a significant concern from a security standpoint.

Another frequently used side-channel attack is the timing attack, which exploits how much time a specific sequence of computations takes for revealing secret information from a neural network. [10] explores the execution time for the victim network given a specific input and then uses it to deduce the network depth and narrow down the architecture search space. Other than architecture, inputs of the model could also be inferred using a timing attack. [7] proposes to use the running time of floating-point multipliers of the first hidden layer for extracting the input. In short, timing attacks have successfully obtained the architecture and the inputs.

Electromagnetic and power-based side-channel attacks utilize the computer system's EM/power leakage to obtain network architecture, network parameters, or inputs. One interesting observation is that network parameters and inputs cannot be inferred at the same time; researchers assume that one of them is known and utilize EM/power data to infer the other one. For example, on the one hand, input information could be extracted with a known network through EM traces [2] or through power traces [45]. On the other hand, network parameters could be obtained with a known input through EM traces [51] and through power traces [46]. Besides, [2, 46] also use EM/power leaks to infer network architecture. To summarize, EM/power side-channel attack is a potent attack in compromising privacy.

Considering that most neural networks are trained and evaluated on GPU platforms, some general-purpose GPU-based attacks have been proposed. For example, recent studies proposed to utilize CUDA spy application to monitor hardware performance counter values over time [33] and GPU context-switching penalty [44] to infer network architecture. Except for the side-channel attacks to GPU platforms, the attacks on other platforms like TPU and edge devices have never been explored.

3.3 Fault Injection Attack

In fault injection attacks, faults could be injected into neural networks through parameters or functions. When injecting faults to parameters, two cases have been considered: soft error [23, 35, 37] and row hammer attack [14, 36]. While the soft error is produced by nature when secondary particles from cosmic rays hit the chip or a defect in computer system design, row hammer error is produced on purpose when an attacker exploits DRAM's structure causes disturbance errors. If an error correction scheme does not exist or is deliberately turned off, such faults might introduce large errors into the data. The faults could also be injected into functions under bad physical conditions such that computation blocks would generate wrong results. Overall, fault injection attacks, without proper defense, might significantly compromise the network's integrity.

As far as we are concerned, all reference papers about fault injection attacks to neural networks try to answer three questions: Where to inject faults? What is the outcome of injection? How to inject faults?

The first research problem that needs to be solved is where the best location to inject faults is. As the first paper to answer the question, [31] proposes a single bias attack which modifies only one parameter for misclassification and gradient descent attack which considers the stealthiness by impacting outputs with specific input patterns. Later, [52] formally formulates this optimization problem

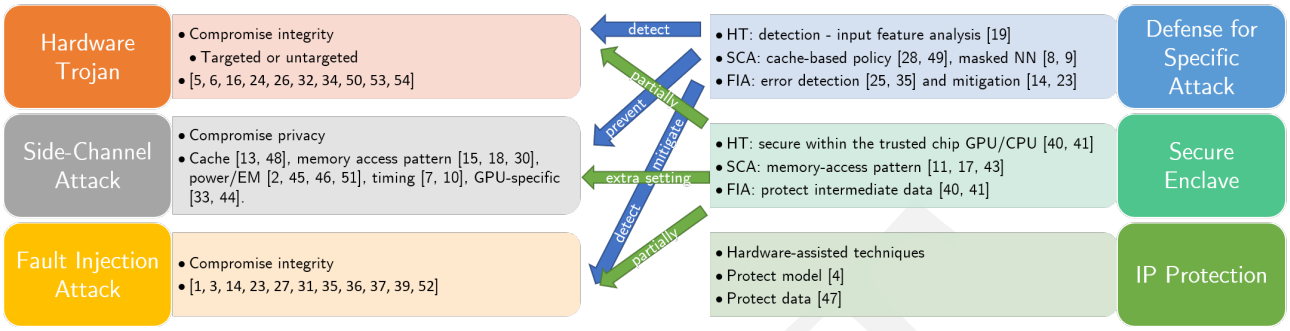

Figure 2: Hardware-related attacks and countermeasures for neural networks. The left column lists the goals for each type of attack. The right column lists how each type of countermeasure helps in improving security for neural network.

and utilizes the alternating direction method of multipliers to solve it. The progressive bit search method is proposed in [36] to figure out the most vulnerable bits for accuracy degradation.

Second, it is of great significance to understand how different network architectures are affected by the same fault injection attack. Most researches focus on faults caused by soft error or row hammer. Based on this assumption, a thorough robustness analysis is conducted regarding how bit error rate affects the overall performance of networks with various type of layers, activation functions, data types, datasets, *etc* [23, 35, 37]. To evaluate the robustness against fault injection attack, [14] use the percentage of vulnerable bits to evaluate the networks' robustness.

Most importantly, a variety of techniques have been used to inject faults. A laser is utilized to cause misclassification by targeting four common nonlinear activation functions [3]. Reduced voltage has been applied to multiple components in FPGAs to characterize the reliability-power trade-off in convolutional neural network [39]. When introduced to the circuits during network execution, clock glitch can disrupt intermediate computation results, leading to wrong outputs [27]. Memory collision in dual-port RAM can be deliberately raised by reducing voltage and increasing temperature, resulting in a higher probability of bit flips [1]. Such techniques all successfully inject faults into neural networks and cause malfunction.

4 COUNTERMEASURES

Since the attacks described in the previous section may significantly compromise the neural network's integrity and privacy, some countermeasures are proposed to detect the existence or mitigate the impact of these attacks.

4.1 Countermeasures for Specific Attack

4.1.1 Defense against Hardware Trojan. Although many software trojan detections and mitigation techniques have been proposed, defenses against hardware trojans are not well-explored. For software trojan, which is usually embedded into the network parameters during the training process, trojan detection can be done by figuring out the minimal inputs [42] or minimal perturbations to the inputs [29] which can cause misclassification. However, such techniques for solving an optimization problem are not useful for detecting hardware trojans. This is because hardware trojan is usually embedded into the network by adding extra circuits or functionality after the models have been trained. Only when the inputs are the same as the attacker defined ones, the trojan can be triggered; partial inputs matching the trigger cannot push the classification result to the wrong label. In the proposed countermeasures against trojan, the only one which may help in hardware trojan is trigger detection by input feature analysis [19]. However, this method is still not applicable to all hardware trojan triggers, especially for sequential ones [32] and even combinational ones associated with a tiny portion of the inputs [6, 50].

4.1.2 Defense against Side-Channel Attack. Regarding side-channel attacks which obtain memory-access pattern or power/EM leaks, some countermeasure has been proposed. To prevent the attackers from overhearing the access pattern from the victim process, [49] proposes secure hierarchy-aware cache replacement policy and [28] utilizes oblivious shuffle to the memory address to protect neural network architecture from being deduced. To further protect the neural network from power/EM side-channel attacks, fully-masked neural networks [8, 9] is proposed to break the primitives of these attacks that inputs are known to attackers. To hide the secret inputs from attackers, blinding [8] and multi-party computation [9] are utilized. Such countermeasures focusing on destroying the attacks primitives, the information attackers are assumed to be known, to protect the neural network privacy.

4.1.3 Defense against Fault-injection Attack. As mentioned in the previous section, faults could be injected into the parameters by soft error or row hammer, or be injected into the function by bad physical conditions like reduced voltage and clock glitch. While the latter type of attack could be detected by voltage and clock signal monitors, the former type of attack exploiting bit flips to cause unexpected outputs is harder to detect. To defend against the former type, a variety of methods have demonstrated their effectiveness. On the one hand, to detect bit flips and significant change in parameter values, redundant neurons are added to the vital data path [25] and a check bit is added to each weight in its binary representation [35]. On the other hand, to mitigate the effect of large value changes, activation magnitude restriction, and low-precision numbers via quantization or binarization are used in [14].

Selective latch hardening technique is also proposed to make data path less sensitive to bit-flip errors [23].

4.2 Secure Enclave

A secure enclave is a hardware-isolated, trusted execution environment that can run kernels in isolation from other processes or code blocks. It is designed for general-purpose processors to provide integrity and privacy guarantees such that attackers cannot snoop or modify the data and the instructions. Thus, the neural network vulnerabilities described in Section III make secure enclave a great choice to counter hardware-related attacks.

Secure Enclave could *partially* protect the neural network from the hardware trojan and fault injection attack. Since the hardware base for a secure enclave is assumed to be trusted, hardware trojan cannot be directly added to the network. However, it could still be inserted between the trusted processor and the peripheral memory devices, enabling memory-based hardware trojan [53]. Besides, a secure enclave protects the code and data from being tampered with by the attackers during the execution; thus, it helps prevent fault injection attacks, especially those targeting parameters in the middle of computation. Nevertheless, techniques like reduced voltage and clock glitch can still cause a malfunction in computation blocks, and others like row hammer in DRAM cannot be excluded. A general-purpose secure enclave for GPU has been proposed in [41] and outsourcing scheme between trusted CPU and untrusted GPU has shown its efficiency and integrity in completing a deep neural network task [40]. Overall, a secure enclave assists in improving the integrity of the neural network.

Even though attackers cannot directly obtain model architecture, parameters, or inputs in a secure enclave due to code and data protection schemes, these secrets can still be inferred via side-channel attacks. Several techniques have been proposed to improve the secure enclave's ability to defend against side-channel attacks. By enabling input-oblivious inference-as-a-service, memory access pattern is prevented from inferring inputs or parameters [11]. By adding instruction and data security unit [43] or designing application-specific memory protection scheme [17], memory- and timing-based side-channel attacks are prevented from inferring the network architecture. To summarize, a secure enclave could help protect against memory- and timing-based side-channel attacks, but none of the enclaves have demonstrated the capacitance in defending against power/EM side-channel attacks.

4.3 IP Protection

Hardware-based techniques can protect the valuable IP of a neural network. Labeling the training data and training the neural network, the two necessary steps in obtaining an application-specific machine learning model, are resource-intensive. Thus, the labeled dataset and the trained model are valuable IPs protected by the model-owner. For example, the model inversion attack has demonstrated that the training data could be extracted from the model weights. To protect training data from being leaked via this attack, an approximate memory system is exploited to mitigate the leakage level and protect training data [47]. Another example of IP violation is model duplication by stealing parameters and building a new model without the model owner's permission. To protect the model from being duplicated by attackers, [4] proposes hardware-assisted neuron obfuscation so that the published model can only be run on a trustworthy hardware device by an authorized user. To summarize, hardware-related methods have shown their effectiveness in protecting IP and should be investigated deeper in the future.

5 CONCLUSIONS & FUTURE DIRECTIONS

In this paper, we surveyed the hardware-based attacks and countermeasures for neural networks. We showed attackers' motivation to steal or corrupt DNN models by exploiting the vulnerabilities exhibited in the training and evaluation processes. Then we categorized the attack approaches and the countermeasures to give a bird's eye view of the hardware-related security problems of neural networks. Finally, we conclude the paper with a brief mention of future directions.

Most current hardware-related attacks still focus on pretty primitive neural network structures dealing with small and simple datasets like MNIST or CIFAR-10. However, in reality, neural networks are usually utilized in much more complex applications like self-driving or natural language processing systems dealing with a significant amount of data. Thus, researchers need to turn their attention to realizing attacks for realistic DNN models.

Plenty of studies aim at bringing up new attack technologies. In contrast, limited research studies focus on detecting or mitigating the impact of hardware-related attacks on neural networks. Therefore, significant work is required to transform today's underdeveloped countermeasures to a set of mature protection mechanisms or even developed security standards of tomorrow.

REFERENCES

[1] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte. 2019. RAM-Jam: Remote Temperature and Voltage Fault Attack on FPGAs using Memory Collisions. In *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*. 48–55.
[2] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. 2019. CSI NN: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 515–532.
[3] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu. 2018. DeepLaser: Practical Fault Attack on Deep Neural Networks. *arXiv: Cryptography and Security* (2018).
[4] A. Chakraborty, A. Mondai, and A. Srivastava. 2020. Hardware-Assisted Intellectual Property Protection of Deep Learning Models. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6.
[5] Joseph Clements and Yingjie Lao. 2018. Hardware trojan attacks on neural networks. *arXiv preprint arXiv:1806.05768* (2018).
[6] J. Clements and Y. Lao. 2019. Hardware Trojan Design on Neural Networks. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5.
[7] G. Dong, P. Wang, P. Chen, R. Gu, and H. Hu. 2019. Floating-Point Multiplication Timing Attack on Deep Neural Network. In *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*. 155–161.
[8] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. 2019. MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection. *arXiv: Cryptography and Security* (2019).
[9] Anuj Dubey, Rosario Cammarota, and Aydin Aysu. 2020. BoMaNet: Boolean Masking of an Entire Neural Network. *ArXiv* abs/2006.09532 (2020).
[10] Vasisht Duddu, Debasis Samanta, D Vijay Rao, and Valentina E Balas. 2018. Stealing neural networks via timing side channels. *arXiv preprint arXiv:1812.11720* (2018).
[11] Karan Grover, Shruti Tople, Shweta Shinde, Ranjita Bhagwan, and Ramachandran Ramjee. 2019. Privado: Practical and Secure DNN Inference with Enclaves. arXiv:1810.00602 [cs.CR]
[12] Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. 2020. Towards Security Threats of Deep Learning Systems: A Survey. *IEEE Annals of the History of Computing* 01 (2020), 1–1.
[13] Sanghyun Hong, Michael Davinroy, Yiğitcan Kaya, Stuart Nevans Locke, Ian Rackow, Kevin Kulda, Dana Dachman-Soled, and Tudor Dumitraş. 2018. Security

analysis of deep neural networks operating in the presence of cache side-channel attacks. *arXiv preprint arXiv:1810.03487* (2018).

[14] Sanghyun Hong, Pietro Frigo, Yigitcan Kaya, Cristiano Giuffrida, and Tudor Dumitras. 2019. Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 497–514.

[15] Xing Hu, Ling Liang, Shuangchen Li, Lei Deng, Pengfei Zuo, Yu Ji, Xinfeng Xie, Yufei Ding, Chang Liu, Timothy Sherwood, and Yuan Xie. 2020. DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 385–399.

[16] X. Hu, Y. Zhao, L. Deng, L. Liang, P. Zuo, J. Ye, Y. Lin, and Y. Xie. 2020. Practical Attacks on Deep Neural Networks by Memory Trojaning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020), 1–1.

[17] Weizhe Hua, Muhammad Umar, Zhiru Zhang, and G. Edward Suh. 2020. GuardNN: Secure DNN Accelerator for Privacy-Preserving Deep Learning. arXiv:2008.11632 [cs.CR]

[18] W. Hua, Z. Zhang, and G. E. Suh. 2018. Reverse Engineering Convolutional Neural Networks Through Side-channel Information Leaks. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. 1–6.

[19] Mojan Javaheripi, Mohammad Samragh, Gregory Fields, Tara Javidi, and Farinaz Koushanfar. 2020. CLEANN: Accelerated Trojan Shield for Embedded Neural Networks. In *International Conference On Computer Aided Design*. ACM, ACM.

[20] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236* (2016).

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[22] Chuan Li. [n.d.]. *OpenAI's GPT-3 Language Model: A Technical Overview, 2020. Available at https://lambdalabs.com/blog/demystifying-gpt-3/.*

[23] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W. Keckler. 2017. Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '17)*. Association for Computing Machinery, New York, NY, USA, Article 8, 12 pages.

[24] W. Li, J. Yu, X. Ning, P. Wang, Q. Wei, Y. Wang, and H. Yang. 2018. Hu-Fu: Hardware and Software Collaborative Attack Framework Against Neural Networks. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 482–487.

[25] Yu Li, Yannan Liu, Min Li, Ye Tian, Bo Luo, and Qiang Xu. 2019. D2NN: A Fine-Grained Dual Modular Redundancy Framework for Deep Neural Networks. In *Proceedings of the 35th Annual Computer Security Applications Conference* (San Juan, Puerto Rico, USA) *(ACSAC '19)*. Association for Computing Machinery, New York, NY, USA, 138–147.

[26] T. Liu, W. Wen, and Y. Jin. 2018. SIN2: Stealth infection on neural network — A low-cost agile neural Trojan attack methodology. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 227–230.

[27] W. Liu, C. H. Chang, F. Zhang, and X. Lou. 2020. Imperceptible Misclassification Attack on Deep Learning Accelerator by Glitch Injection. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6.

[28] Y. Liu, D. Dachman-Soled, and A. Srivastava. 2019. Mitigating Reverse Engineering Attacks on Deep Neural Networks. In *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 657–662.

[29] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. 2019. ABS: Scanning Neural Networks for Back-Doors by Artificial Brain Stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (London, United Kingdom) *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 1265–1282.

[30] Yuntao Liu and Ankur Srivastava. 2020. GANRED: GAN-Based Reverse Engineering of DNNs via Cache Side-Channel. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop* (Virtual Event, USA) *(CCSW'20)*. Association for Computing Machinery, New York, NY, USA, 41–52.

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

[32] Z. Liu, J. Ye, X. Hu, H. Li, X. Li, and Y. Hu. 2020. Sequence Triggered Hardware Trojan in Neural Network Accelerator. In *2020 IEEE 38th VLSI Test Symposium (VTS)*. 1–6.

[33] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh. 2018. Rendered Insecure: GPU Side Channel Attacks are Practical. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (Toronto, Canada) *(CCS '18)*. Association for Computing Machinery, New York, NY, USA, 2139–2153.

[34] Tolulope A Odetola, Hawzhin Raoof Mohammed, and Syed Rafay Hasan. 2019. A stealthy hardware trojan exploiting the architectural vulnerability of deep learning architectures: Input interception attack (IIA). *arXiv preprint arXiv:1911.00783* (2019).

[35] Minghai Qin, Chao Sun, and Dejan Vucinic. 2017. Robustness of neural networks against storage media errors. *arXiv preprint arXiv:1709.06173* (2017).

[36] A. S. Rakin, Z. He, and D. Fan. 2019. Bit-Flip Attack: Crushing Neural Network With Progressive Bit Search. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 1211–1220.

[37] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G. Wei. 2018. Ares: A framework for quantifying the resilience of deep neural networks. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. 1–6.

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 779–788. https://doi.org/10.1109/CVPR.2016.91

[39] B. Salami, E. B. Onural, I. E. Yuksel, F. Koc, O. Ergin, A. Cristal Kestelman, O. Unsal, H. Sarbazi-Azad, and O. Mutlu. 2020. An Experimental Study of Reduced-Voltage Operation in Modern FPGAs for Neural Network Acceleration. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 138–149.

[40] Florian Tramer and Dan Boneh. 2019. Slalom: Fast, Verifiable and Private Execution of Neural Networks in Trusted Hardware. In *International Conference on Learning Representations*.

[41] Stavros Volos, Kapil Vaswani, and Rodrigo Bruno. 2018. Graviton: Trusted Execution Environments on GPUs. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 681–696.

[42] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy (SP)*. 707–723.

[43] X. Wang, R. Hou, Y. Zhu, J. Zhang, and D. Meng. 2019. NPUFort: A Secure Architecture of DNN Accelerator against Model Inversion Attack. In *Proceedings of the 16th ACM International Conference on Computing Frontiers* (Alghero, Italy) *(CF '19)*. Association for Computing Machinery, New York, NY, USA, 190–196.

[44] J. Wei, Y. Zhang, Z. Zhou, Z. Li, and M. A. Al Faruque. 2020. 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)*. 125–137.

[45] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang Xu. 2018. I Know What You See: Power Side-Channel Attack on Convolutional Neural Network Accelerators. In *Proceedings of the 34th Annual Computer Security Applications Conference* (San Juan, PR, USA) *(ACSAC '18)*. Association for Computing Machinery, New York, NY, USA, 393–406.

[46] Y. Xiang, Z. Chen, Z. Chen, Z. Fang, H. Hao, J. Chen, Y. Liu, Z. Wu, Q. Xuan, and X. Yang. 2020. Open DNN Box by Power Side-Channel Attack. *IEEE Transactions on Circuits and Systems II: Express Briefs* 67, 11 (2020), 2717–2721.

[47] Q. Xu, M.T. Arafin, and G. Qu. 2020. MIDAS: Model Inversion Defenses Using anApproximate Memory System. In *2020 IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*.

[48] Mengjia Yan, Christopher W. Fletcher, and Josep Torrellas. 2020. Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2003–2020.

[49] Mengjia Yan, Bhargava Gopireddy, Thomas Shull, and Josep Torrellas. 2017. Secure Hierarchy-Aware Cache Replacement Policy (SHARP): Defending Against Cache-Based Side Channel Attacks. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (Toronto, ON, Canada) *(ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 347–360.

[50] J. Ye, Y. Hu, and X. Li. 2018. Hardware Trojan in FPGA CNN Accelerator. In *2018 IEEE 27th Asian Test Symposium (ATS)*. 68–73.

[51] K. Yoshida, T. Kubota, S. Okura, M. Shiozaki, and T. Fujino. 2020. Model Reverse-Engineering Attack using Correlation Power Analysis against Systolic Array Based Neural Network Accelerator. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5.

[52] Pu Zhao, Siyue Wang, Cheng Gongye, Yanzhi Wang, Yunsi Fei, and Xue Lin. 2019. Fault Sneaking Attack: A Stealthy Framework for Misleading Deep Neural Networks. In *Proceedings of the 56th Annual Design Automation Conference 2019* (Las Vegas, NV, USA) *(DAC '19)*. Association for Computing Machinery, New York, NY, USA, Article 165, 6 pages.

[53] Y. Zhao, X. Hu, S. Li, J. Ye, L. Deng, Y. Ji, J. Xu, D. Wu, and Y. Xie. 2019. Memory Trojan Attack on Neural Network Accelerators. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1415–1420.

[54] Minhui Zou, Yang Shi, Chengliang Wang, Fangyu Li, WenZhan Song, and Yu Wang. 2019. PoTrojan: powerful neural-level trojan designs in deep learning models. arXiv:1802.03043 [cs.CR]