Two Sides of the Same Coin: Boons and Banes of Machine Learning in Hardware Security

Wenye Liu, *Graduate Student Member, IEEE*, Chip-Hong Chang, *Fellow, IEEE*, Xueyang Wang, Chen Liu, Jason M. Fung, Mohammad Ebrahimabadi, *Graduate Student Member, IEEE*, Naghmeh Karimi, *Member, IEEE*, Xingyu Meng, *Graduate Student Member, IEEE*, and Kanad Basu, *Senior Member, IEEE*

***Abstract*— The last decade has witnessed remarkable research advances at the intersection of machine learning (ML) and hardware security. The confluence of the two technologies has created many interesting and unique opportunities, but also left some issues in their wake. ML schemes have been extensively used to enhance the security and trust of embedded systems like hardware Trojans and malware detection. On the other hand, ML-based approaches have also been adopted by adversaries to assist side-channel attacks, reverse engineer integrated circuits and break hardware security primitives like Physically Unclonable Functions (PUFs). Deep learning is a subfield of ML. It can continuously learn from a large amount of labeled data with a layered structure. Despite the impressive outcomes demonstrated by deep learning in many application scenarios, the dark side of it has not been fully exposed yet. The inability to fully understand and explain what has been done within the super-intelligence can turn an inherently benevolent system into malevolent. Recent research has revealed that the outputs of Deep Neural Networks (DNNs) can be easily corrupted by imperceptibly small input perturbations. As computations are brought nearer to the source of data creation, the attack surface of DNN has also been extended from the input data to the edge devices. Accordingly, due to the opportunities of ML-assisted security and the vulnerabilities of ML implementation, in this paper, we will survey the applications, vulnerabilities and fortification of ML from the perspective of hardware security. We will discuss the possible future research directions, and thereby, sharing a roadmap for the hardware security community in general.**

Manuscript received March 28, 2021; revised May 17, 2021; accepted May 22, 2021. Date of publication May 27, 2021; date of current version June 14, 2021. This work was supported by the National Research Foundation, Singapore, through its National Cybersecurity R&D Programme/Cyber-Hardware Forensic & Assurance Evaluation R&D Programme under Award CHFA-GC1-AW01. This article was recommended by Guest Editor A. Wu. *(Corresponding author: Chip-Hong Chang.)*

Wenye Liu and Chip-Hong Chang are with the School of Electric and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: wliu015@e.ntu.edu.sg; echchang@ntu.edu.sg).

Xueyang Wang, Chen Liu, and Jason M. Fung are with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: terry.wang@intel.com; chen1.liu@intel.com; jason.m.fung@intel.com).

Mohammad Ebrahimabadi and Naghmeh Karimi are with the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD 21250 USA (e-mail: ebrahimabadi@umbc.edu; nkarimi@umbc.edu).

Xingyu Meng and Kanad Basu are with the Department of Electrial and Computer Engineering, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: xxm150930@utdallas.edu; kanad.basu@utdallas.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JETCAS.2021.3084400.

Digital Object Identifier 10.1109/JETCAS.2021.3084400

***Index Terms*— Machine learning, deep learning, hardware security, hardware Trojan, counterfeit IC, physically unclonable functions, malware detection, edge AI, cloud FPGA, physical attacks, side-channel attacks, adversarial examples.**

I. Introduction

THE past decade has witnessed a rapid advance in **M**achine **L**earning (ML) especially **D**eep **L**earning (DL) algorithms. With their intrinsic capabilities for fast and efficient data processing, ML solutions have come to the fore in various domains including computer vision, self-driving vehicles, **I**nternet-of-**T**hings (IoT), Industry 4.0, healthcare and cybersecurity. An important facet of ML algorithms is the amount and type of data used for training. By training a statistical model adequately for a target application, the trained model can make predictions and inferences that exceed human-level accuracy, which shifts fully autonomous decisions by ML from prospect to imminent reality. In recent years, ML schemes like **S**upport **V**ector **M**achines (SVM), **L**ogistic **R**egression (LR) and **D**eep **N**eural **N**etworks (DNN) have been considered as promising solutions in solving conventional problems for hardware security including detection of **H**ardware **T**rojans (HT) and counterfeit **I**ntegrated **C**ircuits (ICs), assessment of **P**hysically **U**nclonable **F**unction (PUF) and investigation of system malware.

With the improvement of **V**ery **L**arge **S**cale **I**ntegration (VLSI) technology and the shrinking transistor's feature size, more complex systems can be integrated in a single die. The high complexity and cost of IC design and fabrication have motivated the outsourcing of design and fabrication to different parties across the globe. The globalization of IC design flow has given rise to various hardware security threats. Among which, HT and IC counterfeiting have received the lion's share of attention. HT are malicious modifications to a circuit that are inserted by malicious entities present in the untrusted IC supply chain. A HT can be used for various purposes, *e.g.*, circuit malfunction, power draining, compromisation of sensitive information, etc. [1]. Identification and avoidance of counterfeit ICs are highly crucial for the industry, government sectors and end-users. This is because even if such an IC works properly initially, it may have a shorter lifetime,

thus jeopardizing the reliability of the system in which it is embedded [2]. The methods presented in [3]–[6] are but a few examples that benefit from ML approaches in detecting such security vulnerability. On the other hand, IC metering schemes, particularly those that exploit **P**hysical **U**nclonable **F**unctions (PUFs) for chip identification, have been used to detect counterfeit ICs [7]. However, PUFs have been shown to be vulnerable to modeling attacks [8], [9] in recent years. Through the deployment of ML schemes, their challenge-response behavior can be predicted to gain successful authentication. Another area where ML techniques are widely applied is malware detection through hardware signatures. With the advantages of high tamper-resistance and low overhead, hardware signatures-based profiling techniques have been popularly used for the detection of malicious behaviors of programs running on general purpose computing systems as well as resource-constrained embedded and cyber-physical systems [10], [11]. ML techniques, which bring significant benefits in building accurate models and performing effective and efficient classification of benign and malicious behaviors, have been leveraged to improve the accuracy of hardware signature-based detection for various types of malware, such as kernel rootkits, network attacks, ransomware and cryptomining malware [12]–[15].

Despite the prominent successes of using MLs in hardware security [16], the deployed ML models also face a high risk from evolving threat landscape as ML applications become pervasive and new use cases emerge. The training data collected from public surveillance, human biometrics, financial, and medical applications, etc. often contain private and personally identifiable information. Rogue data can also be injected maliciously to poison the learning or corrupt the output of a DNN. The hyperparameters and weights of well-trained ML models are valuable **I**ntellectual **P**roperties (IPs). Their unauthorized access and disclosure can result in not only significant revenue loss but also leakage of unprotected assets for ML models that are deployed for security applications. The economic gain and tangible incentives have attracted various hardware attack surfaces on the ML model executed in both powerful computing platforms in the cloud environment and resource-constrained edge devices. Early works focus on studying the effect of attacking ML hardware under simulated or emulated environments [17]–[19]. Recent fault attacks show that it is indeed possible to falsify the output and breach the integrity of MLs on their physical implementations. Remote attacks can degrade the ML's prediction accuracy on a powerful server through malicious manipulation of model parameters stored in the off-chip memory [20]. On the edge computing platforms, laser beam interference [21] and glitch injections [22], [23] have been exploited to tamper the internal processing of a ML model, resulting in a successful misclassification of the target input. Other vicious goals such as stealing or reverse engineering the trained model architecture/weights and training data are often achieved through side-channel attacks or a combination of side-channel and fault injection attacks. Similar to traditional side-channel attacks that target cryptography, side-channel attacks targeting ML exploit the correlation between ML assets and certain measurable metrics of a specific ML implementation, rather than any weakness of the ML algorithm. Such attack tampers with the privacy and confidentially of ML assets, such as the input data to a ML model, and the model structure and/or parameters. Depending on the computer systems (e.g., **C**entral **P**rocessing **U**nit (CPU), **G**raphical **P**rocessing **U**nit (GPU), **F**ield **P**rogrammable **G**ate **A**rray (FPGA), etc.) on which the ML algorithms are implemented, different types of side-channels have been exploited, such as power [24], **E**lectro**m**agnetic (EM) [25], cache timing [26], and memory access patterns [27].

This paper surveys recent advances of the aforementioned ML applications for hardware security and the security of ML on hardware implementation. The rest of the paper is organized as follows. Hardware Trojan, IC counterfeiting, ML models including SVM, LR and DNN, and hardware implementation platforms for ML are introduced in Section II. Section III presents ML techniques for hardware security enhancement such as circuit abnormality identification, PUF robustness evaluation and malware detection. The hardware attack methodologies and countermeasures of deployed ML models are described in Section IV. A reflection of both sides of ML in the hardware security arena is further discussed in Section V. Section VI concludes the paper.

II. Background

A. Supply Chain Security Threats

1) Hardware Trojan: Globalization of IC design flow has jeopardized the security and trustworthiness of ICs and introduced new security vulnerabilities including but not limited to (1) tampering the circuit to insert malicious circuitry in the form of HTs aiming at denying services or leaking sensitive data; (2) reverse engineering the circuit aiming at gaining information about the design, and consequently stealing and claiming the ownership of the IPs; (3) cloning and unauthorized overproduction by the foundry; and (4) recycling ICs from outdated systems and utilizing them in the target systems. Such security threats can impose significant financial burden on industry, government sectors and end users, and/or jeopardize their security via leaking sensitive data, taking control of the system or forcing denial of service [28].

In practice, malicious modification of hardware in untrusted fabrication facilities has emerged as a major security concern. The inserted Trojan may result in hardware malfunction or leak sensitive information from a chip [29]. A HT can be inserted into a chip through an untrusted fabrication facility or by a malicious system integrator or designer. The HTs inserted are usually stealthy in nature in order to escape the verification and manufacturing test processes. This makes their detection highly difficult, especially when there is no Trojan-free reference to compare with. Design for trust techniques that do not assume a trusted foundry in their threat model, such as logic obfuscation, has thus been actively explored. Besides its primary objective of protection against reverse engineering-based IC piracy, logic obfuscation can prevent Trojan insertion by untrusted system integrators or

the manufacturing facilities by altering the signal transition probabilities or concealing the real functionality of the design to make fulfilling the stealthy requirement of Trojan insertion harder. This can increase the chance of Trojan detection during the manufacturing test process.

A HT mainly consists of trigger and payload circuitries. It alters the target circuit's functionality or leaks the secret data upon activation [30]. The trigger and payload circuitries can be even as small as one gate. Given that the Trojans are rarely triggered, detecting them via structural and functional testing is highly challenging [31]. On the other hand, Trojan detection based on side-channel information collected from ICs, including power side-channel analysis [32], regional supply currents analysis [33] and path delay investigations [30], [34], has received the lion's share of attention in recent years. However, the fabrication process variations make such side-channel based Trojan detection schemes more challenging.

2) IC Counterfeiting: IC counterfeiting is another crucial security threat that needs to be tackled, and mainly refers to the production and/or distribution of illegitimate chips. A counterfeit chip can be recycled, faulty, out-of-spec, overproduced or cloned circuitry [35]. A recycled IC is obtained from outdated systems to be embedded in the target system [36]. However, as the electrical specification of such IC has already been changed due to aging, it experiences a degraded performance and ultimately faster wearout [37].

Realizing illegitimate ICs via reverse engineering a fabricated chip through delayering the chip and subsequent imaging is considered as a major IC counterfeiting threat [38]. Logic obfuscation and camouflaging techniques can help to mitigate such vulnerabilities [39]. On the other hand, IC overproduction without the consent of the design company can result in the infiltration of unauthentic chips into the market. Such IC counterfeiting threat is mainly mitigated via logic obfuscation, split manufacturing [40], and metering schemes [41]. The latter can highly benefit from deploying **P**hysically **U**nclonable **F**unctions (PUFs) to individualize the protection [42], [43]. Thanks to the imperfections of integrated circuits' fabrication process, each PUF generates a unique signature for each die. By embedding a PUF in each fabricated chip, the design company can trace the realized instances. PUFs are not only useful for IC metering purposes, but also for generating secret data such as keys for cryptographic modules. Moreover, PUF primitives can be deployed for authentication purposes in a network of multiple nodes such as IoT frameworks [44], [45].

In sum, several approaches have been proposed in the literature to tackle the supply chain security threats. However, due to the large size and complexity of the state-of-the-art circuits, and the versatility and stealthy nature of hardware attacks, focusing on conventional security-preserving algorithms to encounter hardware security threats is neither sufficient nor efficient. Thereby, deploying learning-based schemes, i.e., ML methods, to leverage the security of the IC can be highly beneficial. Meanwhile, the adversaries also benefit from ML-based schemes to exploit the vulnerabilities of the supply chain. This results in a cat and mouse contest between the legitimate and rogue elements in the supply chain.

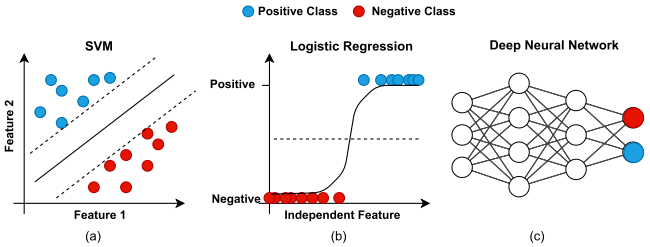


Fig. 1. Machine learning algorithms: (a) SVM, (b) LR, and (c) DNN.

B. ML Models

ML models have been demonstrated to improve the domain of Hardware security in various scenarios, particularly in HT detection. In this section, we will discuss some fundamental ML models.

1) Support Vector Machine: SVM is a supervised ML model that classifies the training data with the help of a hyperplane, as expressed in Eq. (1) [46]. As shown in Fig. 1(a), SVM utilizes the regularization parameter to scale down the error margin that occurs in ML classification by introducing a region between two classes of data, with a margin of $\frac{2}{\|w\|}$ as shown in Eq. (2). This margin is further optimized with the creation of nonlinear classifiers by applying a kernel to maximize the margin of hyperplanes [47]. One of the most common kernels used in SVM is the **R**adial **B**asis **F**unction (RBF). It is important to fine tune the kernel hyperparameters to optimize the performance of the SVM classifier. **O**ne-**C**lass SVM (OC-SVM) is a common category of SVM that employs the training data from one class to develop the model [48].

$$w^T x + b = 0 \tag{1}$$

$$\begin{cases} w^T x + b \geq 1 \\ w^T x + b \leq -1 \end{cases} \tag{2}$$

2) Logistic Regression: Similar to SVMs, LR is also a supervised learning scheme. Eq. (3) shows the linear relationship between the log-odd and the predictor variables x_k. The logistic function is naturally *sigmoid*, which models the binary dependent variables along the function. As shown in Fig. 1(b), the function follows a 'S' shape. Input values are combined linearly using the coefficients to predict a binary output. The LR classifier determines the optimal coefficient values from the training data through the maximum-likelihood estimation approach [49]. Eq. (4) shows the classification equation to find the best fit β parameters of the model.

$$log_b \frac{a}{1-a} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots \tag{3}$$

$$y = \begin{cases} 0, & \beta_1 x_1 + \beta_2 x_2 + \cdots \geq 0 \\ 1, & otherwise \end{cases} \tag{4}$$

3) Deep Neural Network: A DNN consists of several layers, including an input layer, an output layer, and multiple hidden layers in between them. Fig. 1(c) shows a neural network with two hidden layers. The weighted input connection, transfer function, and output connection generate the fundamental computational units, called *neurons*, at each layer, as shown in Fig. 2 [50]. The outputs of each layer pass through an

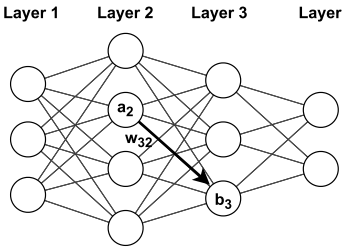


Fig. 2. A simple example of a DNN with 4 layers, where a_2 is the 2^{nd} neuron of layer 2, b_3 is the 3^{rd} neuron of layer 3, and the weight between them is w_{32}.

activation function before they are used as inputs to the subsequent layer, as shown in Eq. (5). The DNN architecture is trained through a back-propagation algorithm, which updates the synaptic weights based on the loss function. Eq. (6) shows the four fundamental equations that support the back-propagation. The first equation shows a symmetric product of the vector $\nabla_a C$ and $\sigma'(z^L)$. δ^L represents the error of the output layer. $\nabla_a C$ is defined as a vector that contains the components of the partial derivatives $\partial C/\partial a^L$, where C is the cost function of the output layer, and a^L is the output layer activation neurons. $\sigma'(z^L)$ is a vector space measures how fast all the activation function σ is changing at the output layer's weighted inputs z^L. The second equation shows a symmetric product of the error at the current layer δ^l in terms of the error in the next layer δ^{l+1}, where $(w^{l+1})^T$ is the transpose of the weight matrix w^{l+1} for the $(l+1)$-th layer. In the third equation, n is the input layer neuron, and m is the output layer neuron. In the fourth equation, w^l_{mn} is the weight at the l-th layer between input neuron n and output neuron m.

$$b^m_l = \lambda\left(\sum_k w^l_{mn} b^{l-1}_n + a^l_m\right) \tag{5}$$

$$\begin{cases} \delta^L = \nabla_a C \odot \sigma'(z^L) \\ \delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \\ \dfrac{\partial C}{\partial a^l_m} = \delta^l_m \\ \dfrac{\partial C}{\partial w^l_{mn}} = a^{l-1}_n \delta^l_m \end{cases} \tag{6}$$

The most prevalent type of DNNs is **C**onvolutional **N**eural **N**etworks (CNN). Majority of the CNNs follow a similar construction of cascading or stacking convolutional sections. Each convolutional section can contain multiple convolutional blocks for feature extraction, followed by one or more fully-connected blocks to compute the class scores. **Re**ctified **L**inear **U**nit (ReLU) is the default activation function for deep network. Typically, each convolutional block contains a convolutional layer, a ReLU layer and discretionally a normalization layer and a pooling layer to reduce the spatial dimension, and each fully connected block consists of a fully connected layer and a ReLU layer. A single step of convolution is implemented by applying a filter to the receptive field of an input as follows:

$$y = \sum_{i=1}^{K}\sum_{j=1}^{K}\sum_{d=1}^{D} w_{i,j,d} \times a_{i,j,d} + \mu \tag{7}$$

where K is the filter height/width, D is the filter depth, w is the filter weight, a is an input element and μ is the bias.

Since y is only an element of the output feature map of a convolution layer, the total computation cost of a convolution layer is that of a single step of convolution in Eq. (7) multiplied by the height, width and depth of the output feature map.

The output features b of a fully-connected layer can be computed by a matrix-vector multiplication as follows:

$$b = Wz + \mu \tag{8}$$

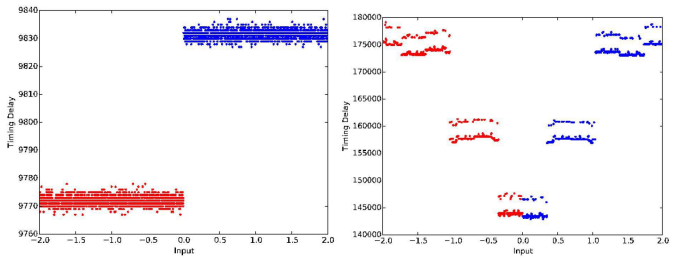
where z is the input vector and W is the weight matrix.

From Eqs. (7) and (8), the fundamental arithmetic operation in both the convolutional layer and the fully-connected layer is the **M**ultiply-**a**nd-**A**ccumulate (MAC) operation. In modern DL models, hundreds of million to billion MACs are executed per image before an inference is made. This enormous computational requirement has posed a great challenge in the efficient hardware implementation of a trained DNN model.

C. Hardware Platforms for ML

A practical ML system is built beyond the consideration of modeling and training. When it comes to the deployment of a ML project, the computing platform is the workhorse to drive the performance per watt in each inference. This can make a difference between the success and failure of a ML project. DNN, as a popular subfield of ML algorithms that revolutionizes computer vision and speech recognition applications, has spurred the research and development in both temporal and spatial architectures for their efficient implementation. This is because the remarkable accuracy of most DNN applications is achieved with an enormous amount of computational power and memories. Three main computing platforms that cater to the computing and storage requirements of different DNN models and workloads, and their prospective security risks are elucidated below.

1) Cloud Servers: Powerful cloud-based servers are usually equipped with GPU and CPU clusters, which aim to provide reliable offline model training and responsiveness to multi-user concurrent queries. In these platforms, DNN performance is highly dependent on the **Ge**neralized **M**atrix **M**ultiply (GEMM) function as both convolutional layer and fully-connected layer can be mapped to a matrix multiplication operation. Different optimization kernels can be implemented for CPU and GPU. For instance, **Open** source **B**asic **L**inear **A**lgebra **S**ubprograms (OpenBLAS) and NVIDIA CUDA® DNN library cuDNN are parallelized backends for GEMM execution on CPU and GPU, respectively [51]. Computational transforms can further reduce the number of operations and storage requirements for different model architectures, for example, the layer types, shapes and sizes. Specifically, **F**ast **F**ourier **T**ransform (FFT) [52] can be adopted if the filter size is greater than 5×5 and Winograd



(a) ReLU (b) Sigmoid

Fig. 3. Timing behaviours of ReLU and sigmoid functions running on ARM Cortex-M3 with CMSIS-NN [25].

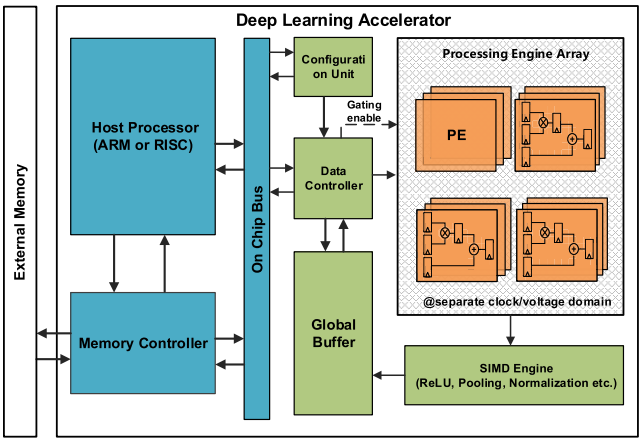


Fig. 4. A general deep learning accelerator with a PE array and separate clock/power domains for DVFS [23].

transformations [53] can be applied for filters of size 3×3 and below. Although the computational efficiency can be improved by mapping and transformation techniques, these optimizations may unconsciously reveal the key parameters of the DNN model [26].

The rise of **A**rtificial **I**ntelligence (AI) has also pressurized cloud developers into the investment of expensive and complex heterogeneous computing resources to satisfy the fast-growing appetite for data processing. FPGA fabrics have been added to overcome the von Neumann bottleneck for very high throughput learning requirements. The run-time reconfigurability of FPGA fabrics enables evolutionary development of multi-tenant storage, applications, and networking, which creates a new business model of cloud AI resource leasing. For example, Baidu has recently released an FPGA-based DNN acceleration service where a single card can provide three **T**era **O**perations **P**er **S**econd (TOPS) of fixed-point computing capabilities [54]. Cloud computing leaders such as Amazon, Microsoft, Tencent and Huawei have also packaged FPGA computing resources into cloud services on their multi-tenant servers, which allow the cloud computing users to purchase FPGA instances for accelerating their specific learning applications without having to maintain a complex IT computing system. This emerging trend of heterogeneous hardware resource sharings for the development of different AI applications may open up new attack avenues for ML poisoning and other security threats. For example, some remote attack vectors like rowhammer from the unprivileged program [55] can be a lurking danger to DL model deployed on cloud-based instances.

2) Embedded Processors: Not all DNN applications can afford the luxury of cloud computing. Owing to connectivity, energy and privacy reasons, some applications may require the DNN to be deployed in low-cost embedded systems, with very limited computing resources and power budget, including the always-on applications and intermittent energy harvesting IoT endpoints. For example, some portable and wearable devices may only have a low-cost low-power microcontroller. Dedicated mathematical functions developed for complicated processors like CPU and GPU are too expensive to be implemented on the embedded processor of these small devices. Thus, lightweight computation kernels, e.g., ARM **C**ortex **M**icrocontroller **S**oftware **I**nterface **S**tandard for **N**eural **N**etwork (CMSIS-NN) [56], optimized for microcontroller have emerged to fill this gap. **S**ingle **I**nstruction, **M**ultiple **D**ata (SIMD) instructions are utilized to compute the neural network operations in parallel. Meanwhile, activation functions are simplified through **SIMD W**ithin **A R**egister (SWAR) for ReLU activation and table-lookup is used for sigmoid and hyper tangent (tanh) functions. By concatenating groups of 32 binary variables into 32-bit registers, SWAR speeds up bitwise operations by 32 times and only three instructions are required to evaluate 32 connections. Observing that a straightforward microcontroller implementation of reduced precision DNN models produces worse performance and energy efficiency, a new **T**ernary weights and **F**our-bit inputs (TF-Net) pipeline was proposed in [57] for the deployment of sub-byte DNN models on microcontrollers. The DNN model is trained with sub-byte inputs and weights. Direct buffer convolution is used to amortize the input unpacking overhead and packed sub-byte multiply-accumulate is used to fuse the MAC operations for four pairs of inputs and weights into one conventional 32-bit multiplication instruction. By adding two extended instructions, **M**ultiply-**S**hift-**A**ccumulate (MSA) and **U**n**p**ac**k** (UPK), to the ARM v7M instruction set architecture, the computation performance and energy efficiency can be further improved.

It is worth noting that different optimizations on various neural network layers yield distinguishable input-dependent timing delays. Fig. 3 shows the timing behaviours of ReLU and sigmoid functions implemented by CMSIS-NN on an ARM microcontroller [25]. These timing patterns leave discriminative traces that can be exploited to identify different types of the activation function. Furthermore, embedded endpoint devices that are publicly accessible are also vulnerable to physical attacks [21], and the firmware on microcontrollers can be more easily hacked.

3) Hardware Accelerators: For latency and performance-critical applications and applications that require the inference to be made on the endpoints where data are generated and processed locally without a network connection, such as self-driving car, neither cloud computing nor embedded microcontroller can fulfil the response time and throughput requirements simultaneously. To address these challenges, dedicated

hardware accelerators for low-latency, high-throughput and energy-efficient DNN processing have emerged [58]–[60]. Customized dataflow, model compression and **D**ynamic **V**oltage **F**requency **S**caling (DVFS) are the major and common optimization techniques used in many DNN accelerators. The specialized dataflow architecture is used to mitigate the memory access bottleneck and maximize the reuse of on-chip data [61]. Model compression and data quantization are effective methods to reduce the number of MAC calculations [62], [63] and the computational complexity of each arithmetic operation [64]. Voltage scaling has been widely implemented to reduce dynamic power consumption. It exploits the excess positive slack of non-critical paths and the guard band preserved at design time to allow the DNN to operate at a lower than rated supply voltage whenever possible without compromising the throughput and accuracy [65], [66]. The throughput is usually boosted by an array of parallel **P**rocessing **E**ngines (PE) running at a higher clock frequency [67]. Fig. 4 depicts a general architecture of a DL accelerator. The dataflow is optimized through the data controller and a global buffer. Each PE is a MAC unit composed of a fixed-point multiplier and accumulator or a fused multiply-accumulation unit. The PE array has a separate clock/voltage domain to facilitate DVFS. Many of these performance optimization techniques, when implemented aggressively and applied without proper privileged control, can be vulnerable to side-channel and other forms of hardware-based attacks. It has been demonstrated that the unique data access patterns of optimized DNN computations can leak sensitive structural parameters of the model [27]. Heavy usage of on-chip SRAM can also create reliability issues, and the performance degradation due to premature aging of hardware components can be an exploitable security vulnerability [17]. Moreover, the power management unit, and separate clock and power domains for DVFS can also be exploited for fault injection attack [68].

III. Machine Learning for Hardware Security

A. Identification of HTs and Counterfeit ICs

1) HTs: Modern IC development is characterized by globally distributed outsourcing activities to focus on core competency, gain manufacturing efficiency and reduce design cycle and turnaround time. By using third-party EDA tools, IPs, and untrusted foundries, it creates opportunities for malicious entities to insert Trojan at various stages of the IC design flow. The malicious modifications of a clean design by HT may lead to severe consequences, *e.g.*, leakage of secret information, denial of service, alteration of major functionalities, etc.. Therefore, HT has become one of the most critical threats to IC production for commercial, consumer as well as military applications [1]. As mentioned in Sec. II-A.1, a HT typically consists of two parts: the Trojan trigger and the Trojan payload. The trigger is designed to activate the Trojan under certain conditions, and the effect of the Trojan depends on the Trojan payload. The Trojan trigger is usually designed to be off from the critical path and is rarely activated. In addition, the Trojan activity is usually dormant during the normal functional execution of the circuit. These factors make it extremely challenging to detect the Trojan due to its stealthy nature. Typically, HTs are classified based on five attributes: insertion phase, abstraction level, location, trigger and payload [69], [70]. An attacker can utilize malicious processors to violate operating system exceptions and modify the open-source processor to create a malicious firmware [71]. Prior works have also demonstrated that Trojans can be introduced during EDA design flow and high-level synthesis [72], [73]. Additionally, Trojan can be activated under unexpected conditions or by silicon wear-out [74], [75]. In what follows, we will describe how various ML algorithms can be used to detect HTs. It is noteworthy to mention that process drift (i.e., the improvement in fabrication process over time that is not reflected in the technology models released to a design house [30]) can affect the accuracy of ML-based Trojan detection schemes in cases where the model has been trained using the pre-fabrication data. Therefore, the impact of such drift needs to be modeled to enhance the Trojan detection accuracy.

a) SVM: SVM has been applied in many prior works for detecting HTs. When a golden (Trojan-free) design is available, on-chip data analysis, gate-level netlist analysis, and runtime traffic information are utilized as feature vectors to distinguish between Trojan and Trojan-free chips using SVM. The on-chip power consumption traces in the frequency domain are adapted to classify Trojans with a two-class SVM [76]. An OC-SVM with RBF kernel trained by the transmission power data collected from on-chip data sensor demonstrates its high accuracy of Trojan detection [77]. Another OC-SVM for Trojan detection was proposed using the minimum number of gates between the input and output nets [78]. Features of Trojan attacks are also extracted from the on-chip traffic to train a SVM for Trojan classification [79]. These models can be further updated to include the latest attacks with **M**odified **B**alanced **W**innow (MBW) algorithm [80]. Besides, SVM is chosen to detect Trojans at run-time based on the hardware complexity analysis of traffic diversion, route looping or core spoofing attack [81]. When a golden chip is not available, the transient power supply current samples are collected and classified by multiple classifiers such as Naive Bayes, Random forests, SVM, LR [82], **P**rocess **C**ontrol **M**onitors (PCMs) and **M**ultivariate **A**daptive **R**egression **S**plines (MARS). The simulation data of the golden chip can provide a predicted side-channel signal of the chip, and are utilized to train an OC-SVM to classify a trojaned IC [83].

b) DNN: DNN models such as **B**ack-**P**ropagation (BP) and Multi-layer Neural Networks are used in various Trojan detection approaches when a golden design is not available. Direct current measurements of a post-deployment chip can be performed anytime when needed. A one-class neural network is trained by the trusted evaluation chip to classify whether the fabricated chip contains Trojans [84]. BP neural networks are used to classify the features of power consumption traces to detect trojaned designs [85]. By passing the features through the hidden layers of the neural network, relevant features will be extracted and used to train the classifier and enhance the classification accuracy. Multi-layer neural network

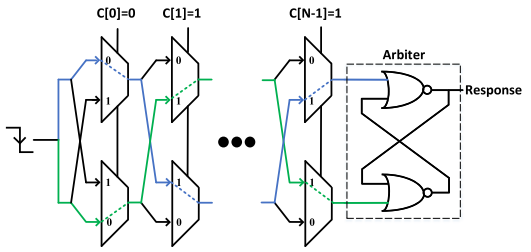
is also applied on gate-level netlist to classify the trojaned design [86], where several features of the netlist are defined and used to train the model for detecting the net corresponding to a Trojan.

c) Others: Statistical learning and Bayesian inference are used to classify the gate profiles and process variation collected from the netlists of Trojan-free and Trojan-inserted chips [87]. A run-time monitoring approach is proposed to detect HTs in microprocessor cores by utilizing **H**alf-**S**pace trees (HS-trees) [88]. HS-trees constitutes a one-class classifier which is trained to provide an early alert of Trojan activation by detecting anomalies in the data streams. **C**ontrollability and **O**bservability for hardware **T**rojan **D**etection (COTD) [89] uses unsupervised k-means clustering to isolate Trojan signals based on the controllability and observability analysis of gate-level netlist. This technique shows its capability in detecting Trojan with high accuracy and low cost, even in the absence of a golden design. Clustering-based learning is also demonstrated to detect Trojan logic by classifying weakly correlated nodes or functionally isolated gates in the netlist [90].

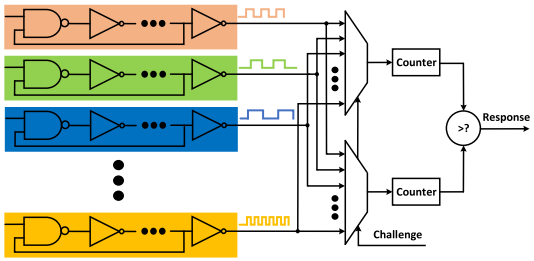
2) IC Counterfeiting: The number of ICs used in electronic systems has increased significantly over the past decades, due to the enhanced complexity of applications and systems [91]. The fabrication of these ICs is outsourced to reduce the overall manufacturing cost, which can lead to the presence of counterfeit IC components. These parts become a crucial threat to the applications related to defense, aerospace and medical systems [92]. Since counterfeiting is a rising threat to the IC manufacturing industry, it is increasingly important to analyze the vulnerabilities of the IC supply chain. The most common counterfeit components are analog ICs, microprocessor ICs, memory ICs, programmable logic ICs, and transistors [91]. A large proportion of counterfeit ICs are actually recycled [91]. The detection of counterfeit components faces significant challenges, such as a wide variety of counterfeit types and the difficulty to inspect potential counterfeit ICs. To improve their detection, it is important to develop regulation of defects and a unique classification of counterfeit components. The use of ML for this purpose is explained as follows. Similar to the case of HT detection, if the model deployed for identifying a counterfeit IC is built based on the pre-fabrication chip specifications, the impact of process variations and process drifts needs to be taken into account when building the ML-based model.

a) SVM: **L**ocal **B**inary **P**atterns (LBP) are non-parametric local features that can be used to train a SVM model in order to distinguish counterfeit and authentic ICs from their registered x-ray images [93]. An OC-SVM is used to classify the used and recycled components from the tests, measurements and analyses of **E**arly **F**ailure **R**ate (EFR) [4], [6]. This model is also used to compare frequency, noticeable performance degradation, and other quality metrics under certain stress conditions to identify recycled FPGAs [5].

b) Neural Network: **A**rtificial **N**eural **N**etwork (ANN) models have been used to provide efficient visual inspections by classifying images of defective and non-defective ICs with image processing techniques [94]. A similar strategy is applied



(a) Arbiter PUF [100].



(b) Ring Oscillator PUF [101].

Fig. 5. Examples of strong and weak PUFs: (a) Arbiter PUF with N multiplexer switches and 2^N CRPs; (b) Ring Oscillator PUF with N ring oscillators and $N(N-1)/2$ CRPs.

with X-ray microscopy of an IC die to differentiate counterfeit from authentic devices with auto-encoder and DNN [93]. DNN is also used to train water-level parametric measurement to identify ICs fabricated in different facilities, which makes it possible to determine if the chips are of the same origin [95].

c) Others: LR is utilized with x-ray 3D imaging to distinguish authentic and recycled ICs by detecting traces of delamination of the dies [96]. Path delays due to aging devices in ICs are leveraged to identify recycled and brand-new devices by establishing fingerprints of brand-new devices with **P**rinciple **C**omponent **A**nalysis (PCA) [97]. Look-up table characteristics and performance degradation are used to train k-mean clustering algorithms, where the recycled FPGAs suffer a higher variation of performance profile [98]. EM fingerprints of ICs are also used in PCA to detect cloned counterfeit ICs [99].

B. Modeling Attacks on PUFs and Robustness Enhancements

PUFs can be divided into two groups of strong and weak PUFs according to the size of their **C**hallenge **R**esponse **P**airs (CRPs). Strong-PUFs (e.g., arbiter-PUF shown in Fig. 5(a)) support an exponentially large set of CRPs, thus are more suitable for authentication purposes. Weak PUFs cover a limited number of CRPs and are more commonly used for device-specific secret key generation for cryptographic circuits, logic locking and watermarking. The **R**ing **O**scillator (RO) PUF (depicted in Fig. 5(b)) is an example of a weak PUF.

Although PUF primitives are supposed to be unclonable and their responses are expected to be unpredictable, their security can be fragile under the ML attacks. In practice, by gaining access to a subset of the target PUF's CRPs, the PUF behavior can be modeled to predict its responses to unseen challenges with high accuracy.

Arbiter-PUF and its derivatives (e.g., feed-forward PUF, XOR-PUF, loop-PUF) that are popularly used in industrial applications [102] have been shown to be prone to modeling attacks [8], [9]. The PUF model is built based on a subset of the targeted PUF's CRPs (via SVM, LR, NN or other ML schemes) obtained by collecting its used CRPs or querying the PUF device directly when it is deployed in the field. This can result in impersonation attacks when the PUFs are used for authentication purposes, and nullifying PUF-based IC metering protection schemes if such schemes are used for IP protection [103], [104]. Due to the limited number of challenges that can be used to provide the required freshness against replay attacks, weak PUFs, such as the RO PUF demonstrated in 5(b), are generally not used for authentication. As weak PUFs are not directly susceptible to ML-based modeling attacks, we will only discuss some ML attacks on strong PUFs and their countermeasures here.

Reliability-based modeling attacks can also jeopardize the security of PUFs. In such an attack proposed in [105], a subset of challenges are given to the PUF repeatedly, and the PUF is modeled based on the similarity/change of the PUF responses to the same challenge. By revealing the sensitivity of each PUF element in the delay-chain to noise, the relative delay of the multiplexers resided in each stage of the arbiter PUF can be modeled. A PUF may also be modeled using its power side-channel where the underlying characteristics of the PUF's circuitry are discerned by monitoring and sampling the current drawn by the PUF [106]. The sampled traces are correlated to the physical specification of the targeted PUF, and can be used to train a ML model to mimic the PUF behavior. In view of the vulnerabilities of strong PUFs to ML-based attacks, ML attack resistance, evaluated under a black-box scenario at least, has become a prerequisite for new strong PUF proposals.

To tackle ML attacks, different types of countermeasures have been proposed in the literature, each of which has its own pros and cons. One group of such countermeasures rely on using symmetric and asymmetric cryptographic algorithms to encrypt the PUF response [107] before its transmission. Another group of countermeasures deploy Hash functions to encrypt the PUF response [108]. These schemes enhance the security against ML-based modeling attacks at the expense of imposing high area overhead. For example, in the scheme proposed in [109], the target strong PUF receives an encrypted instead of a plain challenge. The encryption is performed by an embedded symmetric cipher whose key bits are generated via a weak PUF resided in the same chip.

Hiding the PUF behavior via concealing the real CRPs by modifying the challenge bit-stream or the PUF response has been proposed in the literature to thwart modeling attacks. The former is mainly referred to as challenge obfuscated PUFs [110], [111] and the latter can be realized through controlled PUFs [108]. Fig. 6 shows an overview of a challenge obfuscated scheme presented in [110]. These countermeasures are resilient against CRP-based modeling attacks but may be tackled via power side-channel based modeling attacks [112]. Note that the challenge obfuscation can be deterministic, e.g., [110] or random in nature, i.e., by deploying other PUFs

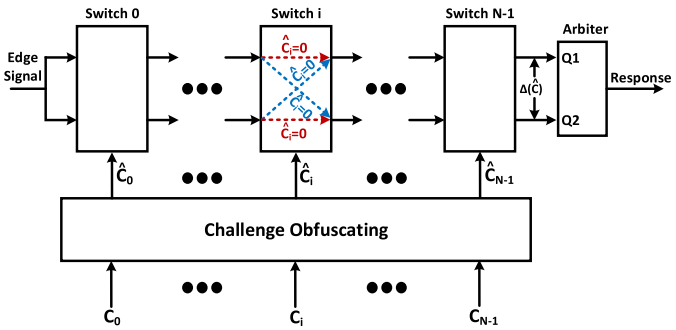


Fig. 6. Challenge obfuscating scheme [110].

to generate a portion of the challenge bit-stream that feeds the targeted PUF [100], [113].

An analog counterpart of the arbiter-PUF, called the **V**oltage **T**ransfer **C**haracteristics PUF (VTC-PUF) has been proposed [114]. As the name implies, it introduces non-linear voltage transfer characteristics to the building blocks to enhance the resiliency of the arbiter-PUF against modeling attacks. This PUF has been shown to be resilient against ML modeling attacks but it can be compromised via side-channel power based modeling attacks [115].

Poisoning the PUF's response to mislead the adversary is another approach to prevent ML attacks. One such method is proposed in [116], where a duplicated or a fake PUF, is embedded in the chip alongside the original PUF to delay the adversary from collecting sufficient correct CRPs for training. It provides fake CRPs for consecutive authentication requests made within a waiting time to mislead the adversary in building an incorrect PUF model. The false PUF multiplexing is fortified against the prediction of waiting time by doubling the waiting time for every unsuccessful guess. Tackling PUF modeling attacks via PUF response poisoning can also be realized via adversarial ML. The approach proposed in [117] sends the toggled PUF response (instead of the genuine response) whenever the challenge bit-stream follows a specific bit pattern or in a periodic manner. In the latter, the PUF response is toggled for every other N received challenges. Such a mixture of genuine and toggled responses prevents an accurate model of the target PUF from being built.

C. Detecting Malware Through Hardware Signatures

Malware detection solutions rooted in hardware bring benefits in lower performance overhead and higher tamper-resistance, compared with software-based ones. There has been a long line of research on detecting different types of malware using signatures based on low-level hardware events collected from hardware components such as **H**ardware **P**erformance **C**ounters (HPCs). The key idea is that the behavior of a program can be modeled with occurrences of low-level hardware events during its execution. Behaviors of malware exhibit unique patterns of hardware events that are distinguishable from benign ones. Such patterns are identified as the signatures of the malware. Therefore, building accurate models to describe the hardware-based signatures of benign and malicious behaviors is critical to improving the capability

of detection techniques. Recent research shows that ML-based techniques have been widely leveraged to improve the effectiveness and efficiency of building models and classifying hardware signatures in malware detection, as elucidated below.

1) Detecting Malicious Applications and System Software: It has been almost a decade since the first proposal of detecting malicious user-level and kernel-level software through hardware signatures. The research from Demme *et al.* [10] was one of the earliest studies in this area, and ML methods had already been leveraged to classify the hardware event traces collected from the execution of 503 malware and 210 benign programs. Different classifiers, such as ***k*-N**earest **N**eighbors (KNN), Decision Trees, and ANNs were implemented and compared in terms of the accuracy of the detection. Though the results were not very promising, it was one of the early attempts to introduce ML to hardware signatures-based malware detection. Tang *et al.* [118] later on proposed an unsupervised ML-based malware detection technique using OC-SVM. Instead of building the hardware profiles of different malware, the proposed technique built the profiles of benign programs. Significant deviations in the profiles will be flagged as potential malware exploitation. Singh *et al.* [12] then extended and expanded the prior works, demonstrating much better accuracy in detecting kernel rootkits. ML feature selection techniques were applied in order to determine the most relevant hardware events for profiling the rootkits' behaviors. The effectiveness of four different classifiers, SVM, OC-SVM, Naive Bayes and Decision Trees, were evaluated for detecting rootkits. Das *et al.* [119] performed a comprehensive survey on the effectiveness of using HPCs for various security applications. In this work, a case study on ML-based malware classification at a fine-grained level was performed. The experimental results showed that different feature extraction classification techniques affected the malware detection accuracy as well as the reproducibility of the findings.

2) Detecting Attacks on Embedded and Cyber-Physical Systems: Microprocessor-based embedded and cyber-physical systems are widely used in critical infrastructure components, and security is becoming increasingly important for those systems [120]. Hardware signatures-based malware detection techniques, which have the advantages of low cost and overhead, have been applied to resource-constrained embedded and cyber-physical systems [11], [121]. ML techniques were leveraged to further reduce the performance/hardware overhead and improve the detection capability. Wang *et al.* [122] introduced ML to the HPC-based technique for detecting malicious firmware running on ARM and PowerPC-based processors. Unsupervised OC-SVM was used to perform the modeling and classification, which significantly reduced the storage overhead when compared with the comparison-based HPC signature matching. A similar ML method was applied to HPC-based anomaly detection for multi-threaded and interrupt-driven processes typically encountered in **P**rogrammable **L**ogic **C**ontrollers (PLCs) [123]. To improve malware detection accuracy with a limited number of available hardware event counters, Sayadi *et al.* [124] proposed a lightweight customized ML-based malware detection method, where a classifier was trained individually with characteristics of a specific class of malware using only four selected HPCs. Enhanced temporal DL was applied to power-grid controller anomaly detection by He *et al.* [125]. The **R**econstruction **E**rror **D**istribution (RED) technique was added to the temporal DL to achieve a high detection accuracy ($>$99.9%) with nearly zero false positives.

3) Detecting Network Attacks, Ransomware and Cryptomining Malware: While hardware signatures-based techniques have been demonstrated to be effective and efficient for detecting network attacks, ransomware and cryptomining malware, ML as a well-proven method to model and classify hardware signatures, has also been applied to the detection of such types of malware. BRAIN [13] combined network statistics and the occurrences of selected low-level hardware events to model the behavior of potential **D**istributed **D**enial-**o**f-**S**ervice (DDoS) attacks. Unsupervised *k*-means clustering was used in the learning and online phases, and the classification was based on supervised SVM. RATAFIA [14] was an unsupervised ransomware detection framework using DNN and **F**ast **F**ourier **T**ransformation (FFT). It was claimed as an accurate, fast and reliable solution to detect ransomware based on hardware signatures collected from HPCs. Mani *et al.* [15] proposed DeCrypto Pro, a framework to detect malware targeting cryptomining using performance counter data. Depending on the available computing resources, the proposed framework is able to switch between a lightweight ML classification model such as Random Forest or KNN, and a **L**ong **S**hort **T**erm **M**emory (LSTM) network for behavior profiling.

IV. Hardware Security of Machine Learning

A. Deployment Threats of ML Models

ML integrity is a primary pillar for AI trust to ensure that the ML systems deliver and maintain the desirable quality of service and are free from unauthorized deliberate or inadvertent manipulation of the system throughout the lifetime of their deployment. Although hardware-oriented attacks on ML systems are not as rampant as they are on cryptographic processors and generic hardware IPs, exploitation of hardware implementation weaknesses and flaws can impact the model integrity and confidentiality of the DL systems. Moreover, the emerging heterogenous hardware platforms supported by the new paradigm of "model once, run optimized anywhere" AI compilers for deploying trained ML models on edge computing devices and the leasing of AI models on the cloud also open out an uncharted territory of security threats. The threat landscape does not preclude existing hardware-oriented attacks such as device reliability [17], [126], [127], malicious attack [19], [21], [22] and side-channel information leakage [25]–[27] from repurposing for ML systems.

Running AI applications on edge devices, often referred to as edge intelligence, raises major concerns about AI model confidentiality. This is because the design of a superior edge AI core for a specific task, such as computer vision, speech processing, natural language processing, etc. requires heavy investment on large labelled training dataset, human expertise and enormous computing power. The training datasets may

consist of private and sensitive information, e.g., medical records, personal traits, demographic profiles, and political views, which should be kept confidential. Unfortunately, the training data are also memorized by the ML models in their parameters, which can be utilized by the adversary to extract subsets of the secret training samples [128]. Thus, a superior and well-trained classifier is not only an IP of high market value, but also a hotbed of opportunities for data thieves owing to the pervasiveness and low security offerings of the edge devices. When the confidentiality of the AI model is compromised, not only will the DNN IP design be counterfeited, the security weaknesses can also be exploited through the extracted model hyperparameters and neuron weights to sneak out the sensitive and confidential training data [129], [130]. The exposed private data can also be leveraged to mount more efficient adversarial attacks towards the DNN system, through for instance, transferable adversarial examples [131], [132].

On the other hand, the run-time reconfigurability of FPGA resources enables users to update data and model structure dynamically on cloud servers. This provides convenience and versatility for small-sized business users and even individuals to quickly iterate their ML models, leading to higher productivity and quality of services. However, the convenience offered by the multi-user FPGA cloud is a two-edge sword. It makes attacks by hardware Trojan easier. Based on the dynamic reconfigurability of the FPGA cloud, attackers have more chances to implement malicious configurations that are detrimental to the system and cause a series of disastrous consequences, such as denial of service, hardware overheating, degraded model performance, and leakage of private information. It is not easy for cloud vendors to assure model integrity merely by identifying the users' identities. This is because the attacker needs not to be a legitimate FPGA cloud user in order to tamper with the configuration scheme and structure when a legitimate user uploads the data and computations onto the FPGA cloud. A closely related attack on DL hardware that leverages outsourced training and transfer learning is the backdoor attack. A neural network backdoor is a hidden pattern injected during training. This hidden pattern will not affect the normal operation, but it can be triggered by a specific input to cause a misclassification or a dysfunction. If the hidden backdoor is embedded in the teacher model, owing to the transfer learning property of DNN model, it can be activated when the model is customized to identify the target label. A malicious tenant can exploit the indirect interaction opportunity with the shared hardware resources of other tenants in a multi-tenant FPGA environment to launch such an attack. As the training data streamed onto the cloud usually includes a large amount of sensitive private information, this may open out unexplored attack surface if a malicious user on the same cluster of FPGA chips can find an access path to the secret information of other users. If the malicious user's codes are encrypted, the cloud vendors may not have the privilege to inspect the decrypted codes before they are configured onto the FPGA fabrics, and it is extremely challenging to detect a malicious hardware bitstream or a program code in encrypted form.

B. Hardware-Based Attacks on Deployed ML Model

Threats concerning DNN model integrity reported in the early stage mainly stem from software-based attacks. Most of these attacks center around input-based adversarial examples [133], data poisoning [134] and software Trojan [135]–[138]. An adversarial example is derived by adding carefully designed perturbations to a target input in either the digital domain or physical form to cause a misclassification when it is presented to the DNN [139], Data poisoning can be achieved by infiltrating carefully crafted samples into the training dataset to manipulate the behavior of the trained model at the inference stage [140]. A software Trojan can be embedded by perturbing the model weights [136]–[138] or inserting an extra module [135] in the pre-trained network. The trojaned model can be activated by specific triggers embedded in normal input data. As opposed to software-based approaches, hardware-oriented attacks can directly modify the ML model's parameters and computation results by tampering the inference process without manipulating the input sample or training data. Some of the reported hardware-oriented attacks are described below.

1) Attacks by Simulation: A number of the attacks reported in the literature are only emulated or simulated without physical implementation. Weight manipulation is one of the attack methods that has been widely evaluated by simulation or emulation to show that the predicted label of a target input to a ML model can be changed by subtle alteration of pre-trained weight parameters. Model weights are stored in the memory cells, which are susceptible to soft errors [17] and permanent faults [141]. The effect of hardware reliability issues on DNN processing has been simulated for CMOS devices [142], [143] and **Re**sistive **R**andom **A**ccess **M**emory (ReRAM) [126]. In contrast to the passive or deliberate aging defects, active malicious fault injection techniques such as laser beam [144] and rowhammer [145] can be utilized to modify the weight values stored in **S**tatic RAM (SRAM) and **D**ynamic RAM (DRAM). If only the effectiveness of an attack is considered without caring about other practical issues, **S**ingle **B**ias **A**ttack (SBA) can be achieved by simply saturating one bias of the last DNN layer [18]. An example of a SBA is illustrated in Fig. 7. It shows that by enlarging the bias of the neuron c_2, the classification result of any input images can be forced into the bird category. SBA can also be extended to the hidden layer with the ReLU activation function, since their outputs are also linearly dependent on the bias. To cause only the selected input images to be misclassified, **G**radient **D**escent **A**ttack (GDA) [18] and fault sneaking attack [146] were simulated. The latter utilized the **A**lternating **D**irection **M**ethod of **M**ultipliers (ADMM) algorithm to minimize the modification of weight parameters. Rakin *et al.* [147] introduced a bit-flip attack by using a progressive bit search method to identify the most vulnerable bits of the model weights to be flipped. Albeit effective, SBA, GDA, fault sneaking attack and bit-flip attack only suggested the values of the weights to be modified in order to cause a misclassification but they did not provide a viable means to make these modifications on the physical hardware implementation of the target DNN model. Other than

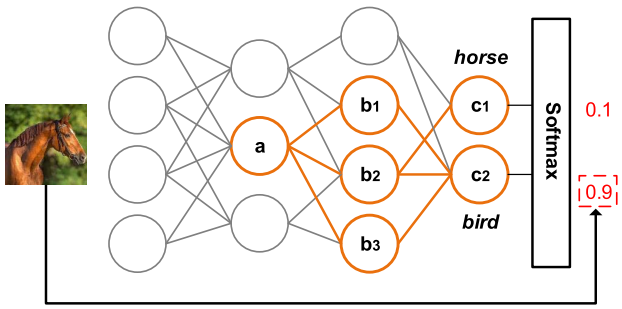


Fig. 7. An illustration of fault injection attack on DNN's weight parameters.

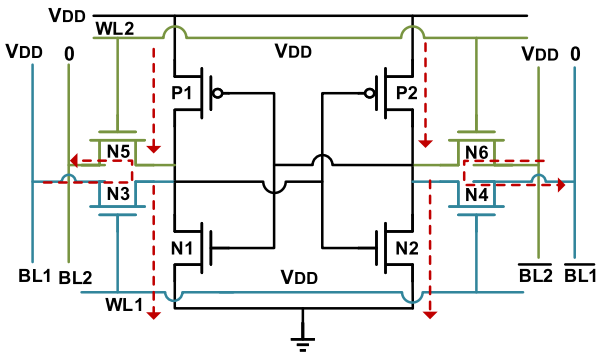


Fig. 8. Memory write collision induced short circuit paths in a dual port RAM cell.

weight interpolation attacks, HT is an alternative approach to maliciously modify ML model operations. By inserting the payload circuits into the activation function, especially the ReLU function, the operation of the neurons can be controlled by a selected trigger key [19]. Embedding Trojan payload in the memory controller can cause zeroing of internal features by identifying the image sequence through the memory traffic [148]. The main limitation of Trojan attacks is that the payload circuit can only be covertly implanted during the hardware development phase, usually through rogue insiders, untrusted foundries, or malicious third-party IP integration. By contrast, fault injection attacks can be applied directly to the deployed hardware. The attacks can be made non-invasive and stealthy without causing a permanent degradation of prediction accuracy on normal or untargeted inputs unlike deliberate aging and reliability-based attacks.

2) Attacks on Physical Implementation: Other than simulating or emulating hardware-based ML attacks, rowhammer was demonstrated physically in [20] to successfully perturb the model parameters of 19 different DNN models, including LeNet5 [149] and its variants, AlexNet [150], VGG16 [151], ResNet50 [152], DenseNet161 [153] and InceptionV3 [154]. The attacked models were implemented in a high performance computing cluster that has 488 nodes. Each node is equipped with an Intel E5-2680v2 2.8GHz 20-core processor, 180 GB of RAM, and 40 of which have two Nvidia Tesla K20m GPUs. The vulnerabilities of different DNN models were evaluated for their severity of discriminative damages by comparing the classification accuracies of the pristine and corrupted models using the respective image classification dataset, MNIST [149], CIFAR10 [155] or ImageNet [156], on which they were trained. The reported rowhammer results indicated that severe accuracy degradation can be inflicted in a surgical attack scenario where specific bits can be flipped. The attacks can still succeed even in a blind attack scenario without any control over the locations of the victim bits in the memory. The attacks revealed that susceptible bits exist widely in common DNN models with 32-bit data representations. This observation gives rise to a potential vulnerability, which is transfer learning can be exploited for a surgical attack on the parameters in the layers that a unknown victim model has in common with a public model.

Besides attacks mounted on powerful server-level computing hardware, Ram-jam [157], a fault injection attack, was demonstrated to successfully attack a pretrained CNN for handwritten character recognition application implemented on Xilinx 7 series Artix-7 (XC7A100T-CS324) FPGAs within a Nexys 4 DDR trainer board. Ram-jam makes use of the write conflict created by writing opposite bit values concurrently into the dual-port RAM cells of the same addresses [157]. The write-conflict is illustrated in Fig. 8. By setting both word lines (WL1 and WL2) to VDD, and writing logic high (VDD) and logic low ("0") on the bitlines BL1 and BL2, respectively, short circuit current paths indicated by the dash lines are formed. By repeatedly triggering the memory collision, a significant amount of transient current is drawn from the power distribution network of the FPGA, which creates a momentary voltage drop on the affected components. The voltage underfeeding can produce bit-flips on the weights stored in memories or cause a timing violation on the **F**inite **S**tate **M**achine (FSM) that controls the activation function for the transitions between the output classes [157]. It is worth noting that the RAM-Jam attack on FSM is activation function agnostic, and it can be launched remotely. Besides causing misclassification, such fault-injection attack can reduce the performance and reliability of ML systems. In other words, RAM-jam is effective even on typical fault-tolerant neural network implementations [158].

Another fault injection attack was demonstrated by using the laser beam to inject faults into commonly used activation functions in the hidden layers of the neural network [21]. The attack was mounted on a simple 3-layer neural network implemented on an ATmega328P microcontroller chip, with sigmoid, ReLU and tanh as the activation function for the second layer and softmax as the activation function for the last layer. A diode pulse laser of 1064 nm wavelength and 20 W pulse power was used for the fault injection. A 20× objective lens was used to reduce the spot size to $15 \times 3.5\ \mu m^2$, which also lowered the pulse power to 8W. The package of this chip was opened to expose the back-side silicon die to the laser. The laser activation time was precisely controlled by a specific delay added after a trigger signal was generated on the commencement of a computation. The chip area is $3 \times 3\ mm^2$ and the sensitivity area to the laser is $\approx 50 \times 70 \mu m^2$. A laser power of 4.5% was sufficient to cause an instruction skip of the activation function. A reasonably good success rate in label misclassification was obtained by perturbing about 50% of the neurons in the chosen layer.

Recently, a more stealthy fault injection attack was demonstrated on nine DNN models pre-trained on Imagenet

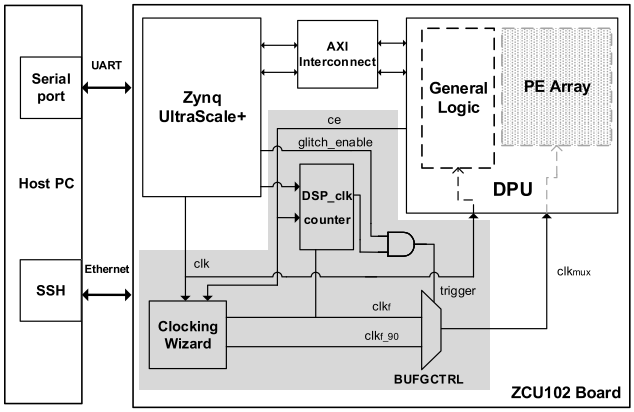


Fig. 9. Clock glitch injection attack module on FPGA-based DNN accelerator [23].

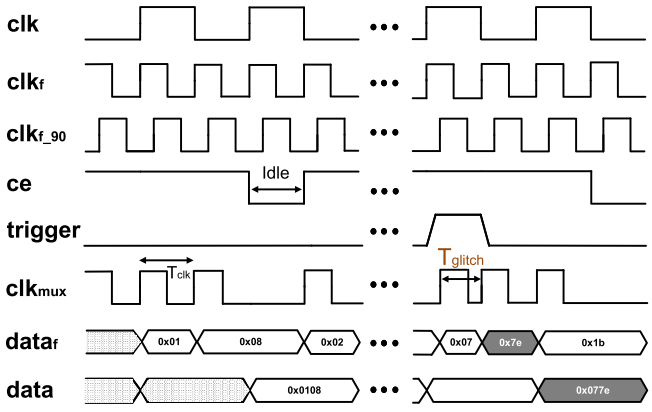


Fig. 10. Timing waveforms of glitch injection attack [23].

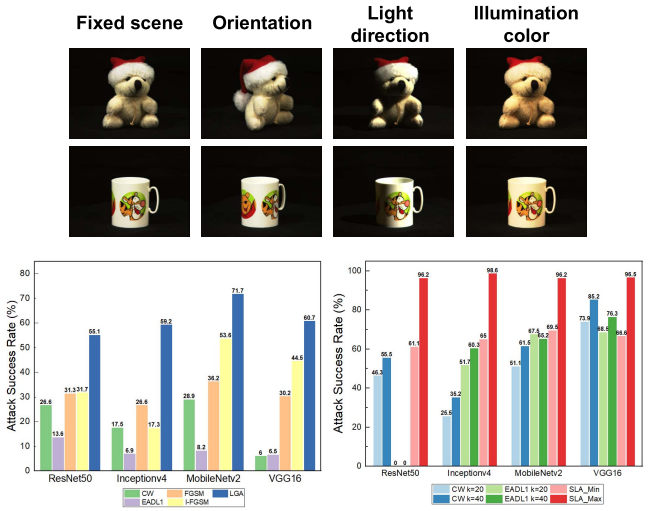


Fig. 11. Examples of two target objects from ALOI dataset, and the attack success rates of LGA and SLA on ResNet50, Inceptionv4, MobileNetv2 and VGG16 DNN models.

dataset, namely Inception v1 to v4, MobileNet v1 and v2, DenseNet121, ResNet50 and VGG16, implemented on a Xilinx ZYNQ UltraScale+ MPSoC device [22]. Instead of targeting conventional temporal and instruction set based architectures or memory array, this attack exploits the separate clock domain used for DVFS and the DNN dataflow for abstracting the pre-trained models into the FPGA overlay of DNN hardware accelerator. Fig. 9 and Fig. 10 illustrate the clock glitch injection module and the corresponding timing waveforms on the FPGA-based DNN accelerator, respectively. The clocking wizard generates the 90° phase shifted clock (clk_{f_90}) for multiplexing into the fast digital signal processing clock (clk_f). The clock counter controls the trigger signal, which determines how frequent and how many glitched clock cycles (T_{glitch}) to be injected into clk_{mux} of the **D**eep Learning **P**rocessing **U**nit (DPU). By perturbing the clock signal of the DPU, it affects only the calculation of the PE array but not other functional modules. The clock buffer with multiplexer enables a glitch to be produced when the trigger signal is high. The intermittent clock glitches can corrupt the intermediate computations of the hidden DNN layers. The errors aggregate as they propagate to the inference layer to cause a misclassification of the target. Once the glitch has subsided, the correct timing will be restored, leaving no trace for detection [22]. With no knowledge about the models, this attack can achieve over 98% of successful target misclassification on eight out of nine models with only ten percent glitches launched into the total computation clock cycles of one inference. Given the model details and inputs, all the models can be successfully attacked regardless of model size, computation latency and original classification accuracy.

Most of the existing hardware-based attacks [20]–[22], [157] on DNN focus on static inputs without considering the object and scene variations. In real-world applications, live captured images are easily affected by multiple factors including object viewing angles, light sources and object distances from the camera. Physical adversarial examples attempt to tackle this problem by extending the software digital input adversarial examples to the physical objects. These adversarial examples are specially crafted shapes or colored stickers [159], [160] that can be attached onto the target object or synthetic shapes generated by 3D printing [161] to fool the DNN classifier in the real scene. What make these adversarial attacks impractical are that they require a decent number of pristine samples of a target to be drawn from different realistic scenarios for generating a physical adversarial example and the generated adversarial example either has a large perturbation area or nonobjective shape [162]. Recently, the clock glitch injection technique [22] was extended to achieve stealthy and robust misclassification for target with variational viewpoints by deriving the attack patterns from only one sample of the target object in a fixed scene [23]. Two attack modes, namely **L**east **G**litch **A**ttack (LGA) and **S**ensitive **L**ayer **A**ttack (SLA), were derived. They differ in the applied glitches defined by the timing patterns of the trigger signal in Fig. 9. A profiling procedure is performed to extract a set of prospective attack patterns on each layer of the deployed DNN to cause a label change of the target. Then, the pattern that has the least glitched cycles on an applied layer is selected as the LGA pattern. The LGA incurs fewer signal transitions in computation and is more covert. On the other hand, the SLA selects

a pattern with the most glitched cycles on the most susceptible layer that has the most prospective patterns. With more glitched cycles injected, the SLA is more robust. The effectiveness of LGA and SLA was demonstrated on ResNet50, Inceptionv4 [163], MobileNetv2 [164] and VGG16 models implemented on the DPU of the Xilinx ZCU102 board. Fig. 11 illustrates the attack scenarios and the corresponding attack success rates of LGA and SLA. 118 objects were selected from the **A**msterdam **L**ibrary of **O**bject **I**mages (ALOI) [165] that captured the sensory variations in object recordings. The attacks were evaluated for different target objects under comprehensive combinations of object-scene variations that include 25 viewing angles ($-60°$ to $60°$), 24 illumination directions and 12 color temperatures. Some of the attacked image samples are shown in Fig. 11. The LGA achieved 55.1% $\sim$ 71.7% attack success rates with only two glitches injected into ten thousand to a million clock cycles of one inference. The attack success rates of the more robust SLA on all the four evaluated models were higher than 96% in any tested combinations of object-scene variational conditions [23].

C. Side-Channel Attacks on ML

With the rising in popularity of **M**achine-**L**earning-**a**s-**a**-**S**ervice (MLaaS), protecting the privacy of ML systems becomes more important. Typically in a ML system, *model* and *data* are two main targets of privacy attacks [166]. Among various attacks that tamper with ML privacy, some of them mainly exploit algorithmic level property to compromise ML privacy, such as membership inference attacks [129], [167], [168] and model inversion attacks [130], [169], which mainly utilize ML **A**pplication **P**rogram **I**nterfaces (APIs) provided by MLaaS vendors to collect information. Unlike these attacks, there are a set of side-channel attacks utilizing vulnerabilities of particular ML hardware implementation to extract confidential information during the ML process. Similar to traditional side-channel attacks, side-channel attacks target measurable ML specific attribute(s) (e.g., power consumption, EM emission, timing, memory access pattern) during either training or inference, and leverage the correlation between ML assets and the selected attributes to deduce the secret. The rest of this subsection will discuss side-channel attacks targeting ML in detail. They are categorized based on the types of assets being leaked.

1) Model Extraction: This type of attack compromises ML model IP privacy by reverse-engineering the structure and parameters of the model to replicate the model. The target assets include the number of layers, type of activation function, connection between layers, parameters of layers, etc. Hua *et al.* [27] proposed one of the first attacks utilizing side-channel information to infer the CNN structure. In the threat model, the CNN accelerator is executed in an isolated environment (e.g., FPGA), so the attacker is not able to observe its internal state. The data are encrypted while being transferred between the CNN accelerator and the off-chip memory. It assumes that the attacker can still probe the memory access pattern, including the address and type (read/write). Similar to a typical *known-plaintext attack* against cryptography, the attacker can feed known input data to the model, observe output from the model, and keep collecting memory access patterns (with timestamp) during inference. It is reported that information of network structure (e.g., boundaries between layers, size of the input/output feature map and filters of each layer, etc.) can be revealed through analyzing the memory access patterns. Furthermore, the attacker is able to reduce the search space of other structural parameters based on a set of design constraints and execution time property. To reveal the network weights, this attack exploits an additional side-channel brought by dynamic zero pruning, which is an optimization technique to eliminate read/write with zero values [170]–[172]. This information is utilized to partially reveal the information of the weights.

Batina *et al.* [25] introduced a power/EM side-channel attack to reverse-engineer model parameters from neural networks. It has been shown that various information of the network can be revealed from the collected power/EM traces. Using **C**orrelation **P**ower **A**nalysis (CPA) with **H**amming **D**istance (HD) leakage model, the weight of each neuron can be recovered without much precision loss. The number of neurons and layers can be discovered by combining **S**imple **P**ower **A**nalysis (SPA) and CPA. The type of activation function being used can be identified by comparing the timing pattern of the collected traces against the profiled pattern of each possible activation function. The attack was validated on 8-bit and 32-bit micro-controllers. Dubey *et al.* [24] have also demonstrated an attack that can successfully reveal the parameters of **B**inarized **N**eural **N**etwork (BNN) using **D**ifferential **P**ower **A**nalysis (DPA). DeepEM [173] reverse engineers the model structure of BNN through the EM side-channel of the FPGA accelerator. It exploits the synthetic dataset generated by Random, FeatureAdversary and FeatureFool algorithms to recover the binarized weight parameters.

Recently, there has been a set of attacks utilizing timing side-channel to extract DNN architecture [26], [174], [175]. Yan *et al.* [26] proposed *cache telepathy* attack based on the observation that DNN heavily utilizes GEMM, which is vulnerable to cache timing side-channel attacks. Therefore, an adversary that co-locates at the same processor as the victim process is able to perform Flush+Reload and Prime+Probe attacks to leak the metadata of GEMM functions. This attack can largely reduce the search space of DNN architectures to facilitate other ML privacy attacks [129], [130]. Hong *et al.* [175] proposed an attack that allows the attacker, who shares instruction cache with the victim running DNN, to perform Flush+Reload attack and infer instructions being used by the victim and further infer the number and types of layers. Duddu *et al.* [174] showed that the execution time of CNN is correlated with the parameters (e.g., stride, kernel size, and the number of filters) of each layer and the number of layers. Linear regression and reinforcement learning are used to reconstruct the model using the execution time information of queries.

Besides FPGA and general purpose CPU, models trained in GPU are also targeted victims of side-channel attacks. Naghibijouybari *et al.* [176] presented an attack that utilizes **M**ulti-**P**rocess **S**ervice (MPS) feature provided by Nvidia

GPUs to allow the adversary kernel to be co-located with the victim kernel on the same GPU core. The adversary kernel can measure the contention of the victim kernel using GPU performance counters. It was shown that the number of neurons can be revealed through this attack. However, this attack can only get coarse-grained information from GPU performance counters because MPS scheduling makes the victim kernel much faster than the adversary kernel [177]. To obtain more fine-grained information, Wei *et al.* [177] proposed to run multiple adversary kernels on the same GPU with the victim and without the MPS feature, which will force the victim to slow down due to context switching. With these settings, not only the number of neurons, but also the filter size, filter number, and stride can be revealed through the side-channel. In [131], Hu *et al.* introduce a generalized framework called *DeepSniffer* to perform model extraction attacks using architectural hints. Unlike previous works, which are not able to identify the dynamic run-time layer sequences due to the architectural and system noises [26], [27], [176], DeepSniffer can perform run-time layer sequence identification to predict a layer execution sequence that is closed to the ground-truth. The framework was demonstrated on a GPU platform using architectural hints, e.g., kernel execution latency, kernel r/w access volume, etc., obtained from either EM-based side-channel or PCIE bus snooping.

2) Leaking Inference Data: Besides the model, input data to inference engine is another victim of side-channel attacks. The input data may contain private information (e.g., medical information) of the **I**nference-**a**s-**a**-**S**ervice (IaaS) users, so it is important to protect the privacy of the data. Wei *et al.* [178] proposed an attack to recover input images of an inference engine from power side-channel information. The attack was performed on an FPGA-based CNN accelerator, using an oscilloscope with a sampling frequency of 2.5GHz to collect the power traces per cycle. For a *passive adversary* who can only monitor the power trace without the capability of triggering the inference, the foreground and background of the recovered image can be distinguished. For an *active adversary* who can further profile the inference engine with arbitrary inputs, the input images can be recovered with higher resolution.

D. Defenses and Countermeasures

The reported hardware vulnerabilities and successful attacks raise increasing concerns that shatter the trustworthiness of ML systems. As opposed to mitigation approaches developed for software-oriented attacks [179]–[181], defense methodologies against hardware-based ML attacks are still in the nascent stage of development. Edge devices such as smartphone, automobiles, robots, drones, cameras, etc. that are endowed with the capability to perceive, reason and act autonomously by running the ML algorithms, are particularly challenging to protect due to their large attack surface, model-specific implementation weaknesses, limited security primitives and design space on edge devices for security-performance trade-offs exploration. Offline analyses motivated by relevant methods to detect adversarial examples and HTs, and custom ML-specific approaches have been proposed to mitigate the threats on ML hardware systems. These proposals can be broadly classified into four categories. The first two categories address the integrity threats. The other two categories are related to model confidentiality protection.

1) Data Type and Model Enhancement: To prevent the dramatic accuracy loss caused by bit-flip of model parameters and in MAC operations, low-precision data representations [20] and bound-constrained dynamic range compression [17], [141] are suggested to limit the error propagation and aggregation. By quantizing the data from 32-bit to 8-bit or even down to 1-bit precision, the proportion of vulnerable parameters due to a single bit error injected into a five layer DNN model running on a high performance server drops from 49% to 0∼2% [20]. On the other hand, clipping the activation output to a bound data range offers around 69% average improvement in classification accuracy of VGG16 running on a GPU workstation at 10^{-5} fault rate [141]. These methods improve ML model robustness on general purpose computing hardware with 32-bit floating point data format. Low-precision fixed-point data type has been a common practice of existing DL accelerator design although the motivation is more on increasing the throughput and energy-efficiency for local inferences in edge applications. To defend against bit-flip attacks targeting quantized model weights, binarization-aware and piece-wise clustering methods [182] are used to train the DL classifier. Binarization can mimic bit-flip noises on the weights, while piece-wise clustering can add fixed single bit-width constraint during the training process. These two training methods can improve the robustness of ResNet-20 and VGG-11 trained on CIFAR-10 dataset by 19.3× and 480.1× respectively, compared to their normally trained counterparts [182]. Weight reconstruction [183] mitigates the effect of bit-flipping by averaging the errors over a grain of weights followed by quantization and clipping of the weight values in a grain. This can minimize the perturbation or error diffusion to the neighboring parameters. It is reported that weight reconstruction can maintain the accuracy of the ResNet-18 model trained on ImageNet dataset at 60% as opposed to the accuracy drop to below 1% of the baseline model under bit-flip attacks. In contrast to conventional data quantization, defensive quantization [184] can further mitigate the error amplification effect of a quantized model during the forward pass of the neural network. The Lipschitz constant in the model training phase is constrained to limit the sensitivity of the input-to-output mapping to white-box L_2-bounded input perturbations. Since this method constrains only the magnitude of the adversarial noise during inference without necessitating any adversarial training [185]–[189] or adaptation of weight matrices, the robustness of the prediction can still be assured even for the unseen test samples. Such defensive quantized networks that are inspired by the theoretical sufficient condition of non-expansiveness are known as $\boldsymbol{L_2}$-**N**onexpansive **N**eural **N**etworks (L2NNNs) [190]. Specifically, defensive quantization is effective in defending against white-box adversarial attacks for quantized models with 5-bit precision and below. Besides, it is shown in [158] that adding more neurons in the hidden layers with restrictions on the parameters' range can further increase the robustness of the model. However,

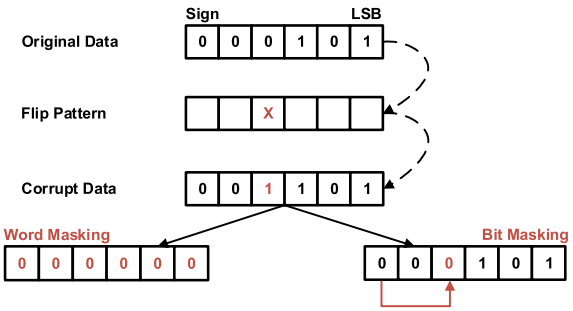


Fig. 12. Word masking and bit masking for fault recovery [172].

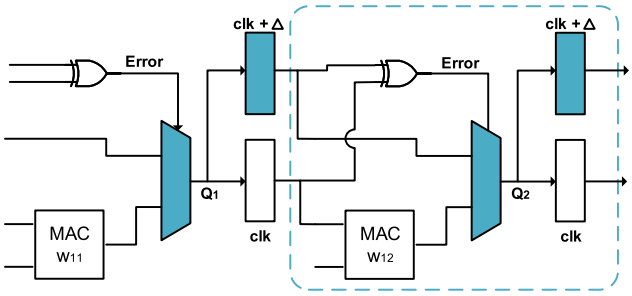


Fig. 13. Block diagram of TE-drop mitigation [191].

the FPGA implementation showed that adding more neurons resulted in significant area/delay overhead.

2) Resilient Hardware: Building error-resilient hardware is an intuitive and direct approach to mitigate the impact of hardware fault attacks. **T**riple **M**odular **R**edundancy (TMR) [192] is a traditional technique to improve the reliability of hardware components against soft errors. As it requires three copies of the functional circuits and a majority voter to correct and mask the faults in any of the copies, the resource and energy overheads incurred are unacceptably high. Reagen *et al.* [172] introduced a DNN accelerator which can tolerate SRAM read faults under voltage variations. Two mitigation techniques, namely word masking and bit masking, are proposed to round the faulty bit(s) towards zero. The difference between these two methods is illustrated in Fig. 12. Word masking resets the whole data word to zero if any bit-flip is detected, whereas bit masking only replaces the flipped bit(s) with the sign bit. With bit masking, 4.4% of the SRAM bitcells can be faulty without causing a prediction error on the DNNs tested on five datasets including MNIST, Forest [193], Reuters [194], WebKB [195] and 20NG [196]. As opposed to making SRAM cells fault-resilient, TE-Drop [191] is an error-tolerant design for the MAC units. It employs an active fault detection module, i.e., Razor flip flops [197], to detect the timing errors in the MAC unit. Instead of correcting the error, it drops the erroneous computation results. This technique is founded on the observations that the worst case timing violations are rare and the omission of partial intermediate results is inherently tolerable by many DNN algorithms. The circuit implementation of TE-Drop is illustrated in Fig. 13. An additional multiplexor and shadow flip flop (marked as blue blocks in Fig. 13) are placed after the MAC unit and in parallel with the original

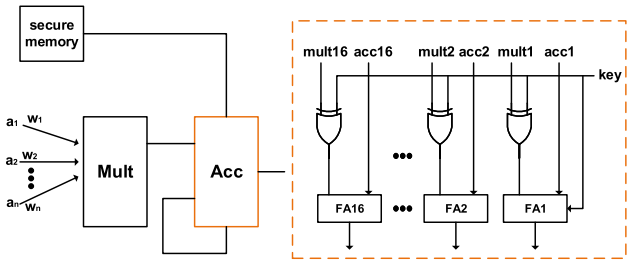


Fig. 14. Hardware architecture of key-dependent accumulator [199].

flip flop, respectively. The shadow flip flop operates with a delayed clock ($clk + \triangle$) while the multiplexor is controlled by the error signal from an XOR gate. Once the error signal is asserted, TE-Drop steals the next clock cycle from its successor MAC unit (shown in the area bounded by the blue dashed box in Fig. 13) to correctly update its own partial sum, and bypass the successor MAC's update. With TE-Drop, the voltage can be scaled down, resulting in 34%-57% of energy savings, and yet introducing only less than 1% of model classification accuracy loss. The results of both masking [172] and TE-Drop [191] were obtained by simulation. Other design fortifications, including hardening selective memory cells [17] and applying modular redundancy on sensitive weights [198], have also been proposed to counteract soft errors in DNN computations.

3) IP and Privacy Protection: Hardware root-of-trust can safeguard DNN IP cores and sensitive user data even in an untrusted environment. Chakraborty *et al.* [199] proposed an obfuscation framework that utilizes hardware-assisted IP protection scheme for DL models. A key-dependent back-propagation algorithm is employed to train the neural network model. Some neurons of the network are locked by the key. During the inference stage, only the trustworthy hardware device with the embedded on-chip key can recover the correct functionality of the model. The hardware design of a key-dependent accumulator [199] is depicted in Fig. 14. The 16 XOR gates for each accumulator take the 16-bit multiplier result and a secure on-chip key bit for a neuron as inputs. The accumulator is assumed to be implemented by a full adder chain (FA1 to FA16). If the key bit of a neuron, $key = 0$, its MAC unit executes normal updates of that neuron by performing a sequence of additions in the accumulator circuit. Otherwise, if $key = 1$, the MAC operation is converted to a two's complement addition for executing a sequence of subtractions. Unauthorized usage of such obfuscated DNN models has led to 73.22% to 80.17% accuracy drop in Fashion-MNIST [200], CIFAR-10 and SVHN [201] datasets based on the simulation results. The notion of **T**rusted **I**nference **E**ngine (TIE) and an associated architecture was introduced in [202]. The idea is to use the **P**seudo **R**andom **N**umber **G**enerators (PRNG) and PUF to perform decryption/deobfuscation of the obfuscated and encrypted DNN models stored in off-chip memory. A decoder and **P**rogrammable **U**sage **C**ontroller (PUC) are required for the key retrieval. Recently, a secure DNN accelerator based on a specific instruction set and memory interface was proposed to protect

user data and model parameters [203]. It includes a memory encryption engine that encrypts the data in DRAM, and an **I**ntegrity **V**erification (IV) engine that detects unauthorized changes on the data read from the external memory. Based on the FPGA prototype, the scheme can preserve the privacy of LeNet, AlexNet, VGG, GoogleNet [204], and ResNet models with less than 2% performance overheads. However, physical side channels were not considered in the threat model.

4) Side-Channel Leakage Mitigation: The unique timing, memory, power/EM and cache patterns in DNN inference execution can expose the structure and weights of a pre-trained DNN model. Internal operation shuffling and computation masking were suggested in [25] to thwart power analysis attacks. Shuffling permutes the order of execution of independent sub-operations to modify the scheduled time of these operations from one execution to another. Thus, it can reduce the success rate of classical differential power/EM attacks. Computation masking assures that sensitive computations are tainted with random values to remove the dependencies between the actual data and the side-channel signatures. If each neuron is individually masked with an independently drawn uniformly random mask for every iteration and every neuron, the attacks reported in [25] can be thwarted. Dubey *et al.* [24] proposed an augmenting masking technique including masked adder trees and ReLU for the fully-connected layers and activation functions, respectively of a three-layer BNN trained on MNIST dataset. The adder tree is masked by splitting each input pixel a_i into two branches, r_i and $a_i - r_i$, where r_i is a unique random number. The final summations of each branch combine and create the original sum. To decorrelate the sign bit from the input, **W**ave **D**ifferential **D**ynamic **L**ogic (WDDL) [205] is applied to compute the sign bit of the adder. A masked **L**ook **U**p **T**able (LUT) [24] is proposed for the carry bits of the binary sign activation function. The MaskedNet [24] inference engine integrates PRNGs into the masked adder tree and activation function to protect the BNN against DPA weight recovery attacks. The first-order attack succeeds at around 200 power traces on the unprotected design. With MaskedNet, the first-order attack fails even with $100k$ power traces while the second-order attack can only succeed with around $3.7k$ collected traces. The overheads of MaskedNet are $2.8\times$, $2.7\times$, $1.7\times$ and $1.3\times$ on latency, LUTs, flip flops and block RAMs, respectively. Another masking design called BoMaNet [206] utilizes a gate-level Boolean masking method to split the secrets and decorrelate the statistical relation between computations and power side-channels. Both linear and non-linear computations of the BNN are converted into a sequence of AND and XOR operations to complete a fully masked hardware implementation. It can resist the first-order attack with up to $2M$ traces at the cost of $3.5\times$ latency and $5.9\times$ area overheads [206].

The weight matrices leaked by cache access patterns can be mitigated through data quantization. This method is particularly effective to safeguard the last few layers of a CNN with smaller filter size [26], but it cannot protect layers with large matrices. Alternatively, cache partitioning can be utilized to assign different ways of the last level cache to different applications. The aim is to block the cache interference between the attacker and the victim [207]. The side-channel leakage from context-switching penalties can be mitigated by reducing the precision of the hardware profiler such as the CUDA profiling tools' interface and the frequency of preemption by suspicious applications [177]. To reduce side-channel leakage through timing and memory patterns, **O**blivious RAM (ORAM) [208] or **M**emory-**T**race **O**bliviousness (MTO) [209] can be utilized. ORAM can conceal its access pattern by continuously shuffling and re-encrypting the data. A practical path ORAM protocol [208] was demonstrated on a secure processor using FPGA in [210]. GhostRider [209] proposed an ORAM-capable processor architecture through a new compiler with an on-chip scratchpad and encrypted RAM. Fake memory accesses can also be created in the TIE for obfuscating the real footprints [202]. Memory timing side-channel attack on GPU can be mitigated by randomizing the width of the coalescing unit and merging transactions across different warps [211]. Besides, GPUGuard [212] prevents side-channel leakage on GPU platform by detecting the spy programs with a decision tree method. Nevertheless, all these mitigations come with a nontrivial area or performance penalty.

V. Reflection and Future Directions

A. On ML Applications in Hardware Security

ML security represents a cat-and-mouse game that is gaining momentum in the AI age. ML-assisted methods on physical silicon security will continue to evolve. For instance, deploying ML schemes to model the PUFs' behavior via their CRPs has been extensively investigated in the literature and several device-level and protocol-level countermeasures have been proposed to tackle such an attack. However, less attention has been paid to the PUF modeling attacks through their power side channel. There is a need for resilient countermeasures against attacks that are realized via monitoring power and EM emanations in PUFs through ML schemes. In practice, to decrease the area and power consumption of delay-based PUFs, instead of implementing multiple PUF circuitries each generating one bit of response, one PUF instance is implemented, and it is queried multiple times to generate a multi-bit response. This can ease the power-based modeling attacks. Thereby, one possible countermeasure to tackle such an attack is increasing the number of PUF instances that are active simultaneously. ML and DL algorithms have been used in collaboration with side-channel attacks. In [213], DL was used to perform cross-device side-channel attacks. 256 devices were used to execute an AES-128 encryption algorithm to generate the power traces. The proposed DL-based methodology was able to achieve 99% accuracy in identifying the various devices in a few seconds.

Another observation is previous ML-assisted HT detection methods were explored mainly via SVM and DNN through extracting design and operation features at various development stages. Design netlist, on-chip sensor and data traffic are used to provide classification features to identify whether the design contains Trojan. SVM and DNN are also applied to detect counterfeit IC by inspecting defects in IC images and

analyzing chip performance to identify recycled components and authentic devices. However, in both fields, most ML approaches require the golden design to establish supervised learning. When the golden design is not available, it is difficult for a design house to verify the authenticity of third-party devices via ML methods. Hence, detection approaches via unsupervised learning, which can still maintain high reliability and accuracy even in the absence of a golden model, is an active research area. For ML-based malware detection through hardware signature, one of the challenges is to determine the most relevant hardware events and suitable classifier for detecting different types of malware. For example, previous works have shown that the selection of hardware events (features) and the classifiers for effectively detecting kernel rootkits and side-channel attacks can be quite different [12], [214]. Without knowing the specific type of malware, it is challenging to employ a universal model and classifier to perform an effective detection. To tackle this issue, Sayadi *et al.* [215] proposed to use a two-stage ML-based approach. The idea is to first classify the applications into one of the known malware classes, such as virus, rootkit or backdoor, using **M**ultinomial **L**ogistic **R**egression (MLR). In the second stage, a specialized ML classifier is selected based on the predicted malware class from the first stage to perform a more accurate classification.

Current security practices follow a form of human intelligence called "deductive reasoning", while on-going security focus on ML approach to increase security level is more inclined towards "inductive reasoning". In inductive reasoning, a conclusion is drawn based on evidences and observations obtained by generalization, statistics, sampling, analogy, trend or causal inference. The conclusion is stronger with more observations and evidences. However, over-reliance on inductive reasoning can lead to logical fallacy because alternative explanations for the assumptions have not been explored. Consequently, a fresh attack which appears for the first time is less likely to be detected by a pre-trained model. Thus, periodic retraining of the ML model with new data is necessary to prevent concept drift, which refers to the accuracy degradation of ML due to the changes in context of the target variable. Deductive reasoning differs from inductive reasoning in that a specific conclusion is reached from a broad observation (universal premise) instead of a broad conclusion is established from known instances (specific premises). It is envisaged that future secure systems will feature a hybrid of deductive, inductive and abductive reasoning. The latter is drawn from two premises, a universal major premise and a specific minor premise with an element of probability. By combining learning with efficient abductive reasoning structure, explainable ML [216] is a plausible direction for future ML-based approach to security enhancement.

B. On Security of Hardware Acceleration of ML

Hardware acceleration for ML applications cuts both ways. Compared with early works that simulate hardware vulnerabilities on ML applications, recent studies demonstrated realistic hardware attacks on deployed classifiers. Software simulation frameworks provide the flexibility and quick turnaround time to evaluate ML implementation vulnerabilities but lack authentic circuit-level information such as transistor switching and gate delay. On the other hand, spice-level simulation is time consuming, and it is infeasible to simulate the actual hardware that runs complex ML models. Given the inevitable tradeoff between the accuracy and complexity of any simulation methods, more attention should be put on evaluating attack vectors for a diversity of ML models and AI applications on real-world ML hardware. The risks and vulnerabilities of existing ML accelerators exposed by deliberate fault injection attacks are practical. They demonstrated that a flawless mapping of a DL model into silicon can make an otherwise secure system vulnerable. Their exploitation is better addressed by new hardware-oriented countermeasures. A foreseeable major challenge in developing the countermeasure comes from its hardware overheads and throughput penalty. In so far as the design of most fault resilient DNN hardware is concerned, the aim is to recover the classification accuracy from uniform errors induced by soft errors [17], [198] or sparse faults caused by voltage instability [66], [172], [191], [217]. However, the faults induced by adversarial attacks are typically neither uniform nor random. From this perspective, the effectiveness of current fault-resilient ML hardware designs against malicious attacks has not been fully attested.

For IP-theft attacks on ML hardware implementations, reverse engineering on model structure and hyperparameters via side-channel information leakage has been reported. Weight recovery on simple and small model architecture has been attempted on edge DNN platforms. Nevertheless, accurate extraction of full ML weights on deployed hardware systems is still a challenging problem. CPU and GPU attacks [26], [177] on conventional shared computing platforms have also been successfully demonstrated recently. Therefore, it is reasonable to believe that these emerging techniques may be adapted for model stealing attacks on FPGA virtualization when multiple users deploy their learning models onto the FPGA cloud. In the multi-tenant FPGA cloud scenario, the biggest barrier is the inability to directly access the victim hardware platform, and the input and output data of the target model. Hence, conventional power side-channel analysis methods are not applicable. The attackers will have to leverage on some backdoor implanted into an instance together with shared power/clock domain to extract the unique hardware traits of the ML model. This is by no means easy due to the huge search space of different compositions of hyper-parameters and layers.

C. On Trust of ML for Electronic Design Automation

Very recently, some new ML models and methodologies like **G**raph **N**eural **N**etwork (GNN) [218] and point cloud as well as machine learning techniques like **D**eep **R**einforcement **L**earning (DRL) [219] and domain adaptation have transformed the way chips are being designed. NP-complete problems are common in **E**lectronic **D**esign **A**utomation (EDA). With the exponentially growing scale of IC, traditional empirical choices or brute-force search methods are inefficient in handling the state space explosion problem. Using ML

methods to accelerate solving of EDA problems has the advantage of knowledge accumulation. The learnt high-level features can be reused to avoid repeating the complicated analysis. According to the comprehensive survey conducted in [220], ML techniques found applications in almost all the stages of an IC design flow. The surveyed ML methodologies in EDA focus merely on accelerating the design space exploration, increasing the efficiency of testing and validation processes, and improving the accuracy of performance prediction and workflow of black-box optimization. Security properties and confidential assets were not incorporated into the predictors, design rules and policies for learning. Trust assurance is a missing piece of the puzzle for the role of ML-based automation in different levels of the IC design hierarchy. With the broad adoption of ML into EDA tool chains, we envisage that more active research will be directed towards enhancing the trust and reliability of the ML-based EDA tools.

At the opposite end, the valuable insights and knowledge gained from the use of ML in EDA flow can also be leveraged to aid adversarial attacks. For example, to improve the search engine for solving EDA problems, NeuroSAT [221] trains a GNN to classify Boolean **Sat**isfiability (SAT) problems and it is simplified in [222] to guide the search process of an existing SAT solver. However, SAT solver can also be utilized as an oracle-guided attack to break logic locking by finding discriminative input patterns to quickly prune the search space of the secret key. In addition, several popular ML models used recently for solving EDA problems such as GNN, **R**ecurrent **N**eural **N**etworks (RNN) [223] and DRL are seldom considered for ML-based hardware attacks and countermeasures. GNN is useful for input data organized in the form of graphs, RNN is good for sequential data such as texts while DRL incorporates DL into the reinforcement learning paradigm for unstructured input data. Comparing with CNN which is more useful for extracting features from grid structure data like images, the inputs to these ML models are closer to the descriptors and representations used for hardware abstraction. These emerging ML models may suggest new insight and opportunities for overcoming the limitations of existing attacks and defenses. For example, GRANNITE [224] is a GPU-accelerated novel GNN model used for fast and accurate RTL power estimation. By representing gate-level netlists as graphs, it possesses good transferability among different designs. This may be exploited for ML-based power side-channel analysis.

VI. Conclusion

ML is an approach to realize AI by learning from experience and making decisions based on sensory data instead of algorithms. Propelled by the hardware technology advancement to support its efficient implementation, ML has now become an indispensable tool for overcoming the dilemma of the growing amount of data and shortage of expertise in many different fields. One notable diversion of ML application from its original goal of mimicking human cognition is its use for hardware security. Rising concerns are also raised about the security of the deployment of ML algorithms on various hardware platforms. On one hand, ML can aid in solving commonly encountered regression, prediction and classification problems in hardware security. On the other hand, adversaries can also leverage ML to achieve nefarious objectives with reduced time and effort. This paper fills the missing space in a remarkable body of surveys on ML research by reviewing the ML journey for hardware security applications, ML hardware implementation vulnerabilities and related countermeasures. We first present the methods for detecting HTs and counterfeit ICs based on SVM, LR and other ML-based approaches. ML-based schemes for detecting malicious applications, firmware, network attacks, ransomwares and cryptomining malwares are introduced along the line. We also expose the other side of the coin by revealing the deployment threats, hardware attack vectors, reverse engineering of model parameters and inference data recovery on cloud and edge ML implementations. As a hardware root-of-trust, strong PUF is singled out for discussion in the light of its susceptibility to ML-based modeling attacks. Recent efforts in developing resilient and trustworthy ML hardware and their limitations are also presented and discussed. Finally, we provide some critical reflections on the challenges, open problems and potential future research directions of ML-assisted methods for hardware security and ML-assisted trusted hardware designs.

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Des. Test. Comput.*, vol. 27, no. 1, pp. 10–25, Jan. 2010.

[2] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[3] O. Aramoon and G. Qu, "Impacts of machine learning on counterfeit IC detection and avoidance techniques," in *Proc. 21st Int. Symp. Qual. Electron. Design (ISQED)*, Santa Clara, CA, USA, Mar. 2020, pp. 352–357.

[4] K. Huang, J. M. Carulli, and Y. Makris, "Parametric counterfeit IC detection via support vector machines," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Austin, TX, USA, Oct. 2012, pp. 7–12.

[5] H. Dogan, D. Forte, and M. M. Tehranipoor, "Aging analysis for recycled FPGA detection," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Amsterdam, The Netherlands, Oct. 2014, pp. 171–176.

[6] K. Huang, Y. Liu, N. Korolija, J. M. Carulli, and Y. Makris, "Recycled IC detection based on statistical methods," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 947–960, Jun. 2015.

[7] U. Guin, D. Forte, and M. Tehranipoor, "Anti-counterfeit techniques: From design to resign," in *Proc. 14th Int. Workshop Microprocessor Test Verification*, Austin, TX, USA, Dec. 2013, pp. 89–94.

[8] U. R Ührmair, F. Sehnke, J. S Ölter, G. Dror, S. Devadas, and J. Ü. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Secur. (CCS)*, Chicago, IL, USA, 2010, pp. 237–249.

[9] U. Ruhrmair *et al.*, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 11, pp. 1876–1891, Nov. 2013.

[10] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Comput. Archit. News*, vol. 41, no. 3, pp. 559–570, Jun. 2013.

[11] X. Wang, C. Konstantinou, M. Maniatakos, and R. Karri, "ConFirm: Detecting firmware modifications in embedded systems using hardware performance counters," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin TX USA, Nov. 2015, pp. 544–551.

[12] B. Singh, D. Evtyushkin, J. Elwell, R. Riley, and I. Cervesato, "On the detection of kernel-level rootkits using hardware performance counters," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Abu Dhabi, UAE, Apr. 2017, pp. 483–493.

[13] V. Jyothi, X. Wang, S. K. Addepalli, and R. Karri, "BRAIN: BehavioR based adaptive intrusion detection in networks: Using hardware performance counters to detect DDoS attacks," in *Proc. 29th Int. Conf. VLSI Design 15th Int. Conf. Embedded Syst. (VLSID)*, Kolkata, India, Jan. 2016, pp. 587–588.

[14] M. Alam, S. Bhattacharya, S. Dutta, S. Sinha, D. Mukhopadhyay, and A. Chattopadhyay, "RATAFIA: Ransomware analysis using time and frequency informed autoencoders," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, McLean, VA, USA, May 2019, pp. 218–227.

[15] G. Mani *et al.*, "DeCrypto pro: Deep learning based cryptomining malware detection using performance counters," in *Proc. IEEE Int. Conf. Autonomic Comput. Self-Organizing Syst. (ACSOS)*, Washington, DC, USA, Aug. 2020, pp. 109–118.

[16] R. Elnaggar and K. Chakrabarty, "Machine learning for hardware security: Opportunities and risks," *J. Electron. Test.*, vol. 34, no. 2, pp. 183–201, Apr. 2018.

[17] G. Li *et al.*, "Understanding error propagation in deep learning neural network (DNN) accelerators and applications," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Denver, CO, USA, Nov. 2017, pp. 1–12, Art. no. 8.

[18] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Irvine, CA, USA, Nov. 2017, pp. 131–138.

[19] J. Clements and Y. Lao, "Hardware Trojan design on neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sapporo, Japan, May 2019, pp. 1–5.

[20] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *Proc. 28th USENIX Secur. Symp. (USENIX Secur.)*, Santa Clara, CA, USA, Aug. 2019, pp. 497–514.

[21] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2018, pp. 2204–2206.

[22] W. Liu, C.-H. Chang, F. Zhang, and X. Lou, "Imperceptible misclassification attack on deep learning accelerator by glitch injection," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[23] W. Liu, C.-H. Chang, and F. Zhang, "Stealthy and robust glitch injection attack on deep learning accelerator for target with variational viewpoint," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1928–1942, 2021.

[24] A. Dubey, R. Cammarota, and A. Aysu, "MaskedNet: The first hardware inference engine aiming power side-channel protection," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, San Jose, CA, USA, Dec. 2020, pp. 197–208.

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

[26] M. Yan, C. W. Fletcher, and J. Torrellas, "Cache telepathy: Leveraging shared resource attacks to learn DNN architectures," in *Proc. 29th USENIX Secur. Symp. (USENIX Secur.)*, Aug. 2020, pp. 2003–2020.

[27] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2018, pp. 4:1–4:6.

[28] Q. Shi *et al.*, "Golden gates: A new hybrid approach for rapid hardware Trojan detection using testing and imaging," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, McLean, VA, USA, May 2019, pp. 61–71.

[29] S. Bhunia and M. Tehranipoor, *The Hardware Trojan War*. Cham, Switzerland: Springer, 2018.

[30] A. Vakil, F. Behnia, A. Mirzaeian, H. Homayoun, N. Karimi, and A. Sasan, "LASCA: Learning assisted side channel delay analysis for hardware Trojan detection," in *Proc. 21st Int. Symp. Qual. Electron. Design (ISQED)*, Santa Clara, CA, USA, Mar. 2020, pp. 40–45.

[31] V. R. Surabhi *et al.*, "Hardware Trojan detection using controlled circuit aging," *IEEE Access*, vol. 8, pp. 77415–77434, 2020.

[32] M. Lecomte, J. Fournier, and P. Maurine, "An on-chip technique to detect hardware Trojans and assist counterfeit identification," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 12, pp. 3317–3330, Dec. 2017.

[33] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: A scalable side-channel approach for hardware Trojan detection," in *Proc. Cryptograph. Hardw. Embedded Syst. (CHES)*, Santa Barbara, CA, USA, Aug. 2010, pp. 173–187.

[34] X. Cui, E. Koopahi, K. Wu, and R. Karri, "Hardware Trojan detection using the order of path delay," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 3, pp. 1–23, Nov. 2018.

[35] R. Karri *et al.*, "Guest editorial special section on hardware security and trust," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 873–874, Jun. 2015.

[36] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2013, pp. 819–823.

[37] A. Vakil, F. Niknia, A. Mirzaeian, A. Sasan, and N. Karimi, "Learning assisted side channel delay test for detection of recycled ICs," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Tokyo, Japan, Jan. 2021, pp. 455–462.

[38] G. L. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "TimingCamouflage: Improving circuit security against counterfeiting by unconventional timing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 91–96.

[39] M. Yasin and O. Sinanoglu, "Transforming between logic locking and IC camouflaging," in *Proc. 10th Int. Design Test Symp. (IDT)*, Amman, Jordan, Dec. 2015, pp. 1–4.

[40] Y. Wang, P. Chen, J. Hu, G. Li, and J. Rajendran, "The cat and mouse in split manufacturing," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 5, pp. 805–817, May 2018.

[41] A. Cui, X. Qian, G. Qu, and H. Li, "A new active IC metering technique based on locking scan cells," in *Proc. IEEE 26th Asian Test Symp. (ATS)*, Taipei, Taiwan, Nov. 2017, pp. 40–45.

[42] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Secur. (CCS)*, Washington, DC USA, Nov. 2002, pp. 148–160.

[43] C.-H. Chang, Y. Zheng, and L. Zhang, "A retrospective and a look forward: Fifteen years of physical unclonable function advancement," *IEEE Circuits Syst. Mag.*, vol. 17, no. 3, pp. 32–62, Aug. 2017.

[44] U. Chatterjee *et al.*, "Building PUF based authentication and key exchange protocol for IoT without explicit CRPs in verifier database," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 3, pp. 424–437, May 2019.

[45] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-based secure communication protocol for IoT," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 3, p. 67, 2017.

[46] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 2008.

[47] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, Pennsylvania, PA, USA, Jul. 1992, pp. 144–152.

[48] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, Jan. 2004.

[49] D. R. Cox, "The regression analysis of binary sequences," *J. Roy. Stat. Soc. B, Methodol.*, vol. 20, no. 2, pp. 215–232, Jul. 1958.

[50] M. H. Hassoun, *Fundamentals Of Artificial Neural Networks*. Cambridge, MA, USA: MIT Press, 1995.

[51] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[52] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," presented at Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, May 2014.

[53] A. Lavin and S. Gray, "Fast algorithms for convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 4013–4021.

[54] *FPGA Cloud Compute*. (Accessed: Mar. 9, 2021). [Online]. Available: https://cloud.baidu.com/product/fpga.html

[55] D. Gruss *et al.*, "Another flip in the wall of rowhammer defenses," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2018, pp. 245–261.

[56] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for ARM Cortex-M CPUs," 2018, *arXiv:1801.06601*. [Online]. Available: https://arxiv.org/abs/1801.06601

[57] J. Yu, A. Lukefahr, R. Das, and S. Mahlke, "TF-Net: Deploying sub-byte deep neural networks on microcontrollers," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 5, pp. 1–21, Oct. 2019.

[58] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[59] B. Moons and M. Verhelst, "An energy-efficient precision-scalable ConvNet processor in 40-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.

[60] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Optimizing the convolution operation to accelerate deep neural networks on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 7, pp. 1354–1367, Jul. 2018.

[61] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Jan. 2016, pp. 262–263.

[62] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," presented at the Int. Conf. Learn. Represent. (ICLR), San Juan, Puerto Rico, May 2016. [Online]. Available: http://arxiv.org/abs/1510.00149

[63] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, Republic of Korea, Jun. 2016, pp. 243–254.

[64] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5784–5789, Nov. 2018.

[65] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Honolulu, HI, USA, Jun. 2016, pp. 1–2.

[66] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei, "14.3 A 28 nm SoC with a 1.2 GHz 568 nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2017, pp. 242–243.

[67] *DPU for Convolutional Neural Network V3.0*, Xilinx, San Jose, CA, USA, 2019.

[68] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: Exposing the perils of security-oblivious energy management," in *Proc. 26th USENIX Secur. Symp.*, Vancouver, BC, Aug. 2017, pp. 1057–1074.

[69] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, Oct. 2010.

[70] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust*. New York, NY, USA: Springer, 2011.

[71] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in *Proc. 1st Usenix Workshop Large-Scale Exploits Emergent Threats (LEET)*, San Francisco, CA, USA, 2008, pp. 1–8, Art. no. 5.

[72] K. Basu *et al.*, "CAD-base: An attack vector into the electronics supply chain," *ACM Trans. Design Autom. Electron. Syst.*, vol. 24, no. 4, pp. 1–30, Jul. 2019.

[73] C. Pilato, K. Basu, F. Regazzoni, and R. Karri, "Black-hat high-level synthesis: Myth or reality?" *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 4, pp. 913–926, Apr. 2019.

[74] C. Dunbar and G. Qu, "Designing trusted embedded systems from finite state machines," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 5s, pp. 1–20, Dec. 2014.

[75] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," in *Proc. IEEE Int. Symp. Hardware-Oriented Secur. Trust*, San Diego, CA, USA, Jun. 2011, pp. 67–70.

[76] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki, "Detection technique for hardware Trojans using machine learning in frequency domain," in *Proc. IEEE 4th Global Conf. Consum. Electron. (GCCE)*, Osaka, Japan, Oct. 2015, pp. 185–186.

[77] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware Trojan design and detection in wireless cryptographic ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 4, pp. 1506–1519, Apr. 2017.

[78] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *Proc. IEEE 22nd Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, Sant Feliu de Guixols, Spain, Jul. 2016, pp. 203–206.

[79] A. Kulkarni, Y. Pino, and T. Mohsenin, "Adaptive real-time Trojan detection framework through machine learning," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, McLean, VA, USA, May 2016, pp. 120–123.

[80] V. R. Carvalho and W. W. Cohen, "Single-pass online learning: Performance, voting schemes and online feature selection," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Philadelphia, PA, USA, Aug. 2006, pp. 548–553.

[81] A. Kulkarni, Y. Pino, and T. Mohsenin, "SVM-based real-time hardware Trojan detection for many-core platform," in *Proc. 17th Int. Symp. Qual. Electron. Design (ISQED)*, Santa Clara, CA, USA, Mar. 2016, pp. 362–367.

[82] M. Xue, J. Wang, and A. Hu, "An enhanced classification-based golden chips-free hardware Trojan detection technique," in *Proc. IEEE Asian Hardware-Oriented Secur. Trust (AsianHOST)*, Yilan, Taiwan, Dec. 2016, pp. 1–6.

[83] Y. Liu, K. Huang, and Y. Makris, "Hardware Trojan detection through golden chip-free statistical side-channel fingerprinting," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2014, pp. 1–6.

[84] Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ICs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2012, pp. 965–970.

[85] J. Li, L. Ni, J. Chen, and E. Zhou, "A novel hardware Trojan detection based on BP neural network," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Oct. 2016, pp. 2790–2794.

[86] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, Thessaloniki, Greece, Jul. 2017, pp. 227–232.

[87] X. Chen, L. Wang, Y. Wang, Y. Liu, and H. Yang, "A general framework for hardware Trojan detection in digital circuits by statistical learning algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1633–1646, Oct. 2017.

[88] R. Elnaggar, K. Chakrabarty, and M. B. Tahoori, "Run-time hardware Trojan detection using performance counters," in *Proc. IEEE Int. Test Conf. (ITC)*, Fort Worth, TX, USA, Oct. 2017, pp. 1–10.

[89] H. Salmani, "COTD: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 338–350, Feb. 2017.

[90] B. Çakır and S. Malik, "Hardware Trojan detection for gate-level ICs using signal correlation based clustering," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2015, pp. 471–476.

[91] U. Guin, D. DiMase, and M. Tehranipoor, "Counterfeit integrated circuits: Detection, avoidance, and the challenges ahead," *J. Electron. Test.*, vol. 30, no. 1, pp. 9–23, Feb. 2014.

[92] US Congress, "Ike Skelton national defense authorization act for fiscal year 2011," U.S. Government Printing Office, Washington, DC, USA, Tech. Rep., 2011.

[93] K. Mahmood, P. L. Carmona, S. Shahbazmohamadi, F. Pla, and B. Javidi, "Real-time automated counterfeit integrated circuit detection using X-ray microscopy," *Appl. Opt.*, vol. 54, no. 13, p. D25, 2015.

[94] N. Asadizanjani, M. Tehranipoor, and D. Forte, "Counterfeit electronics detection using image processing and machine learning," *J. Phys., Conf. Ser.*, vol. 787, Jan. 2017, Art. no. 012023.

[95] A. Ahmadi, M.-M. Bidmeshki, A. Nahar, B. Orr, M. Pas, and Y. Makris, "A machine learning approach to fab-of-origin attestation," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Austin, TX, USA, Nov. 2016, pp. 1–6.

[96] B. Ahmadi, B. Javidi, and S. Shahbazmohamadi, "Automated detection of counterfeit ICs using machine learning," *Microelectron. Rel.*, vols. 88–90, pp. 371–377, Sep. 2018.

[97] X. Zhang, K. Xiao, and M. Tehranipoor, "Path-delay fingerprinting for identification of recovered ICs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Austin, TX, USA, Oct. 2012, pp. 13–18.

[98] M. M. Alam, M. Tehranipoor, and D. Forte, "Recycled FPGA detection using exhaustive LUT path delay characterization," in *Proc. IEEE Int. Test Conf. (ITC)*, Fort Worth, TX, USA, Nov. 2016, pp. 1–10.

[99] A. Stern, U. Botero, B. Shakya, H. Shen, D. Forte, and M. Tehranipoor, "EMFORCED: EM-based fingerprinting framework for counterfeit detection with demonstration on remarked and cloned ICs," in *Proc. IEEE Int. Test Conf. (ITC)*, Phoenix, AZ, USA, Oct. 2018, pp. 1–9.

[100] M. Ebrahimabadi, M. Younis, W. Lalouani, and N. Karimi, "A novel modeling-attack resilient arbiter-PUF design," in *Proc. 34th Int. Conf. VLSI Design 20th Int. Conf. Embedded Syst. (VLSID)*, Feb. 2021, pp. 123–128.

[101] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, Jun. 2007, pp. 9–14.

[102] M. Ebrahimabadi, M. Younis, and N. Karimi, "Hardware assisted smart grid authentication," in *Proc. IEEE Int. Conf. Commun.*, Montreal, QC, Canada, Jun. 2021.

[103] J. Zhang *et al.*, "Design and implementation of a delay-based PUF for FPGA IP protection," in *Proc. Int. Conf. Comput.-Aided Design Comput. Graph.*, Guangzhou, China, Nov. 2013, pp. 107–114.

[104] J. X. Zheng and M. Potkonjak, "A digital PUF-based IP protection architecture for network embedded systems," in *Proc. 10th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Los Angeles, CA, USA, Oct. 2014, pp. 255–256.

[105] G. T. Becker, "The gap between promise and reality: On the insecurity of XOR arbiter PUFs," in *Proc. Cryptograph. Hardw. Embedded Syst. (CHES)*, Saint-Malo, France, Sep. 2015, pp. 535–555.

[106] T. Kroeger, W. Cheng, S. Guilley, J.-L. Danger, and N. Karimi, "Cross-PUF attacks on arbiter-PUFs through their power side-channel," in *Proc. IEEE Int. Test Conf. (ITC)*, Washington, DC, USA, Nov. 2020, pp. 1–5.

[107] J. A. Roy, F. Koushanfar, and I. L. Markov, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, Oct. 2010.

[108] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. 18th Annu. Comput. Secur. Appl. Conf.*, Las Vegas, NV, USA, Dec. 2002, pp. 149–160.

[109] E. I. Vatajelu, G. D. Natale, M. S. Mispan, and B. Halak, "On the encryption of the challenge in physically unclonable functions," in *Proc. IEEE 25th Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, Rhodes, Greece, Jul. 2019, pp. 115–120.

[110] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, "Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1109–1123, Apr. 2019.

[111] Y. Gao *et al.*, "Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PerCom Workshops)*, Sydney, NSW, Australia, Mar. 2016, pp. 1–6.

[112] T. Kroeger et al, "Making obfuscated PUFs secure against power side-channel based modeling attacks," in *Proc. Des. Automat. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021.

[113] P. H. Nguyen, D. P. Sahoo, C. Jin, K. Mahmood, U. Rührmair, and M. Van Dijk, "The interpose PUF: Secure PUF design against state-of-the-art machine learning attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 243–290, Aug. 2019.

[114] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2015, pp. 653–658.

[115] T. Kroeger, W. Cheng, S. Guilley, J.-L. Danger, and N. Karimi, "Effect of aging on PUF modeling attacks based on power side-channel observations," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2020, pp. 454–459.

[116] C. Gu, C.-H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing lightweight-PUF-based authentication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1183–1196, Jun. 2021.

[117] S.-J. Wang, Y.-S. Chen, and K. S.-M. Li, "Adversarial attack against modeling attack on PUFs," in *Proc. 56th Annu. Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 2019, pp. 1–6.

[118] A. Tang, S. Sethumadhavan, and S. J. Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, Gothenburg, Sweden, Sep. 2014, pp. 109–129.

[119] S. Das, J. Werner, M. Antonakakis, M. Polychronakis, and F. Monrose, "SoK: The challenges, pitfalls, and perils of using hardware performance counters for security," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 20–38.

[120] F. Khorrami, P. Krishnamurthy, and R. Karri, "Cybersecurity for control systems: A process-aware perspective," *IEEE Des. Test. Comput.*, vol. 33, no. 5, pp. 75–83, Oct. 2016.

[121] K. Ott and R. Mahapatra, "Hardware performance counters for embedded software anomaly detection," in *Proc. IEEE 16th Int. Conf Dependable, Autonomic Secure Comput., 16th Int. Conf Pervas. Intell. Comput., 4th Int. Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Athens, Greece, Aug. 2018, pp. 528–535.

[122] X. Wang *et al.*, "Malicious firmware detection with hardware performance counters," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 160–173, Jul. 2016.

[123] P. Krishnamurthy, R. Karri, and F. Khorrami, "Anomaly detection in real-time multi-threaded processes using hardware performance counters," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 666–680, 2020.

[124] H. Sayadi, H. M. Makrani, O. Randive, S. Manoj PD, S. Rafatirad, and H. Homayoun, "Customized machine learning-based hardware-assisted malware detection in embedded devices," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, New York, NY, USA, Aug. 2018, pp. 1685–1688.

[125] Z. He, A. Raghavan, G. Hu, S. Chai, and R. Lee, "Power-grid controller anomaly detection with enhanced temporal deep learning," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Rotorua, New Zealand, Aug. 2019, pp. 160–167.

[126] T. Liu, W. Wen, L. Jiang, Y. Wang, C. Yang, and G. Quan, "A fault-tolerant neural network architecture," in *Proc. 56th ACM/IEEE Des. Automat. Conf. (DAC)*, Las Vegas, NV, USA, Jun. 2019, pp. 1–6.

[127] W. Liu and C.-H. Chang, "Analysis of circuit aging on accuracy degradation of deep neural network accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sapporo, Japan, May 2019, pp. 1–5.

[128] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, Oct. 2017, pp. 587–601.

[129] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 3–18.

[130] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Denver, CO, USA, Oct. 2015, pp. 1322–1333.

[131] X. Hu *et al.*, "DeepSniffer: A DNN model extraction framework based on learning architectural hints," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Lausanne, Switzerland, Mar. 2020, pp. 385–399.

[132] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: Protecting against DNN model stealing attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS P)*, Stockholm, Sweden, Jun. 2019, pp. 512–527.

[133] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.

[134] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2018, pp. 19–35.

[135] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for Trojan attack in deep neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 218–228.

[136] T. Liu, W. Wen, and Y. Jin, "SIN^2: Stealth infection on neural network—A low-cost agile neural Trojan attack methodology," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Apr. 2018, pp. 227–230.

[137] A. S. Rakin, Z. He, and D. Fan, "TBT: Targeted neural network attack with bit Trojan," in *Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 13198–13207.

[138] "Trojaning attack on neural networks," in *Proc. Netw. Distrib. Syst. Secur. (NDSS) Symp.*, San Diego, CA, USA, Feb. 2018, pp. 1–15.

[139] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.

[140] A. Shafahi *et al.*, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Montréal, QC, Canada, Dec. 2018, pp. 6106–6116.

[141] L.-H. Hoang, M. A. Hanif, and M. Shafique, "FT-ClipAct: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2020, pp. 1241–1246.

[142] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2018, pp. 1–6.

[143] M. Sabbagh, C. Gongye, Y. Fei, and Y. Wang, "Evaluating fault resiliency of compressed deep neural networks," in *Proc. IEEE Int. Conf. Embedded Softw. Syst. (ICESS)*, Las Vegas, NV, USA, Jun. 2019, pp. 1–7.

[144] B. Selmke, S. Brummer, J. Heyszl, and G. Sigl, "Precise laser fault injections into 90 nm and 45 nm SRAM-cells," in *Proc. Smart Card Res. Adv. Appl.*, Bochum, Germany, Nov. 2015, pp. 193–205.

[145] Y. Kim *et al.*, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *Proc. ACM/IEEE 41st Int. Symp. Comput. Archit. (ISCA)*, Minneapolis, MN, USA, Jun. 2014, pp. 361–372.

[146] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *Proc. 56th Annu. Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 2019, pp. 165:1–165:6.

[147] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 1211–1220.

[148] Y. Zhao *et al.*, "Memory Trojan attack on neural network accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Grenoble, France, Mar. 2019, pp. 1415–1420.

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

[150] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[151] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," presented at the Int. Conf. Learn. Represent. (ICLR), San Diego, CA, USA, May 2014.

[152] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[153] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2261–2269.

[154] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.

[155] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[156] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.

[157] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions," in *Proc. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC)*, Atlanta, GA, USA, Aug. 2019, pp. 48–55.

[158] M. Alam *et al.*, "Enhancing fault tolerance of neural networks for security-critical applications," 2019, *arXiv:1902.04560*. [Online]. Available: http://arxiv.org/abs/1902.04560

[159] 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 *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Vienna, Austria, Oct. 2016, pp. 1528–1540.

[160] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1625–1634.

[161] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 284–293.

[162] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't believing: Towards more robust adversarial attack against real world object detectors," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, London, U.K., Nov. 2019, pp. 1989–2004.

[163] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2017, pp. 4278–4284.

[164] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.

[165] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, Jan. 2005.

[166] B. Liu, M. Ding, S. Shaham, W. Rahayu, F. Farokhi, and Z. Lin, "When machine learning meets privacy: A survey and outlook," *ACM Comput. Surveys*, vol. 54, no. 2, pp. 1–36, Apr. 2021.

[167] L. Song, R. Shokri, and P. Mittal, "Membership inference attacks against adversarially robust deep learning models," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 50–56.

[168] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "MemGuard: Defending against black-box membership inference attacks via adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Nov. 2019, pp. 259–274.

[169] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp. (USENIX Security)*, Austin, TX, USA, 2016, pp. 601–618.

[170] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, Jun. 2016, pp. 1–13.

[171] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Toronto, ON, Canada, Jun. 2017, pp. 27–40.

[172] B. Reagen *et al.*, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, Jun. 2016, pp. 267–278.

[173] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "DeepEM: Deep neural networks model recovery through EM side-channel information leakage," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, San Jose, CA, USA, Dec. 2020, pp. 209–218.

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

[175] S. Hong *et al.*, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," 2018, *arXiv:1810.03487*. [Online]. Available: http://arxiv.org/abs/1810.03487

[176] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, "Rendered insecure: GPU side channel attacks are practical," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Toronto, ON, Canada, Oct. 2018, pp. 2139–2153.

[177] 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 *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Valencia, Spain, Jun. 2020, pp. 125–137.

[178] 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 *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, San Juan, Puerto Rico, Dec. 2018, pp. 393–406.

[179] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against Trojan attacks on deep neural network systems," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2020, pp. 897–912.

[180] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "STRIP: A defence against Trojan attacks on deep neural networks," in *Proc. 35th Annu. Comput. Secur. Appl. Conf.*, Dec. 2019, pp. 113–125.

[181] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[182] Z. He, A. S. Rakin, J. Li, C. Chakrabarti, and D. Fan, "Defending and harnessing the bit-flip based adversarial weight attack," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 14083–14091.

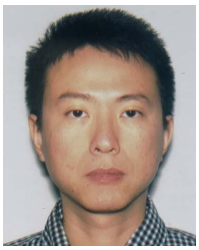
[183] J. Li *et al.*, "Defending bit-flip attack through DNN weight reconstruction," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6.

[184] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," presented at the Int. Conf. Learn. Represent. (ICLR), New Orleans, LA, USA, May 2019.
[185] A. Shafahi *et al.*, "Adversarial training for free!" in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2019, pp. 1–12.
[186] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," presented at the Int. Conf. Learn. Represent. (ICLR), Vancouver, BC, Canada, Apr. 2018, pp. 1–10.
[187] B. Li, C. Chen, W. Wang, and L. Carin, "Certified adversarial robustness with additive noise," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2019, pp. 1–11.
[188] Z. He, A. S. Rakin, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 588–597.
[189] A. Shafahi, M. Najibi, Z. Xu, J. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," in *Proc. 24th AAAI Conf. Artif. Intell.*, New York, NY, USA, Feb. 2020, pp. 5636–5643.
[190] H. Qian and M. N. Wegman, "L2-nonexpansive neural networks," presented at the Int. Conf. Learn. Represent. (ICLR), New Orleans, LO, USA, May 2019.
[191] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "ThUnderVolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2018, pp. 1–6.
[192] J. M. Johnson and M. J. Wirthlin, "Voter insertion algorithms for FPGA designs using triple modular redundancy," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Program. Gate Arrays (FPGA)*, Monterey, CA, USA, Feb. 2010, pp. 249–258.
[193] J. A. Blackard, "Comparison of neural networks and discriminant analysis in predicting forest cover types," Ph.D. dissertation, Dept. Forest Sci., Colorado State Univ., Fort Collins, CO, USA, 1998, Art. no. aAI9921979.
[194] D. D. Lewis. (1997). *Reuters-21578 Text Categorization Collection Data Set*. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection
[195] M. Craven *et al.*, "Learning to extract symbolic knowledge from the world wide Web," in *Proc. 15th AAAI Conf. Artif. Intell.*, Madison, WI, USA, Jul. 1998, pp. 509–516.
[196] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization," in *Proc. 14th Int. Conf. Mach. Learn.*, Nashville, TN, USA, Jul. 1997, pp. 143–151.
[197] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 22nd Digit. Avionics Syst. Conf.*, San Diego, CA, USA, USA, Dec. 2003, pp. 7–18.
[198] N. Khoshavi, A. Roohi, C. Broyles, S. Sargolzaei, Y. Bi, and D. Z. Pan, "SHIELDeNN: Online accelerated framework for fault-tolerant deep neural network architectures," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[199] A. Chakraborty, A. Mondai, and A. Srivastava, "Hardware-assisted intellectual property protection of deep learning models," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[200] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: https://arxiv.org/abs/1708.07747
[201] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. Neural Inf. Process. Syst. (NIPS) Workshop Deep Learn. Unsupervised Feature Learn.*, Granada, Spain, Dec. 2011, pp. 1–9.
[202] M. Isakov, L. Bu, H. Cheng, and M. A. Kinsy, "Preventing neural network model exfiltration in machine learning hardware accelerators," in *Proc. Asian Hardw. Oriented Secur. Trust Symp. (AsianHOST)*, Hong Kong, Dec. 2018, pp. 62–67.
[203] W. Hua, M. Umar, Z. Zhang, and G. E. Suh, "GuardNN: Secure DNN accelerator for privacy-preserving deep learning," 2020, *arXiv:2008.11632*. [Online]. Available: https://arxiv.org/abs/2008.11632
[204] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.
[205] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, vol. 1, Paris, France, Feb. 2004, pp. 246–251.
[206] A. Dubey, R. Cammarota, and A. Aysu, "BoMaNet: Boolean masking of an entire neural network," in *Proc. 39th Int. Conf. Comput.-Aided Design*, San Diego, CA, USA, Nov. 2020, pp. 1–9.
[207] F. Liu *et al.*, "CATalyst: Defeating last-level cache side channel attacks in cloud computing," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Barcelona, Spain, Mar. 2016, pp. 406–418.
[208] E. Stefanov *et al.*, "Path ORAM: An extremely simple oblivious RAM protocol," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Berlin, Germany, Nov. 2013, pp. 299–310.
[209] C. Liu, A. Harris, M. Maas, M. Hicks, M. Tiwari, and E. Shi, "GhostRider: A hardware-software system for memory trace oblivious computation," in *Proc. 20th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Istanbul, Turkey, Mar. 2015, pp. 87–101.
[210] M. Maas *et al.*, "PHANTOM: Practical oblivious computation in a secure processor," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Berlin, Germany, 2013, pp. 311–324.
[211] E. Karimi, Y. Fei, and D. Kaeli, "Hardware/software obfuscation against timing side-channel attack on a GPU," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, San Jose, CA, USA, Dec. 2020, pp. 122–131.
[212] Q. Xu, H. Naghibijouybari, S. Wang, N. Abu-Ghazaleh, and M. Annavaram, "GPUGuard: Mitigating contention based side and covert channel attacks on GPUs," in *Proc. ACM Int. Conf. Supercomput.*, Phoenix, AZ, USA, Jun. 2019, pp. 497–509.
[213] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-DeepSCA: Cross-device deep learning side channel attack," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.
[214] H. Wang, H. Sayadi, S. Rafatirad, A. Sasan, and H. Homayoun, "SCARF: Detecting side-channel attacks at real-time using low-level hardware features," in *Proc. IEEE 26th Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, Jul. 2020, pp. 1–6.
[215] H. Sayadi *et al.*, "2SMaRT: A two-stage machine learning-based approach for run-time specialized hardware-assisted malware detection," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 728–733.
[216] R. Goebel *et al.*, "Explainable AI: The new 42?" in *Proc. Int. Cross-Domain Conf. Mach. Learn. Knowl. Extraction*, Hamburg, Germany, Aug. 2018, pp. 295–303.
[217] D. Shin, W. Choi, J. Park, and S. Ghosh, "Sensitivity-based error resilient techniques with heterogeneous multiply–accumulate unit for voltage scalable deep neural network accelerators," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 3, pp. 520–531, Sep. 2019.
[218] G. Zhang, H. He, and D. Katabi, "Circuit-GNN: Graph neural networks for distributed circuit design," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA, vol. 97, Jun. 2019, pp. 7364–7373.
[219] H. Wang *et al.*, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.
[220] G. Huang *et al.*, "Machine learning for electronic design automation: A survey," *ACM Trans. Des. Autom. Electron. Syst.*, 2021. [Online]. Available: https://arxiv.org/abs/2102.03357
[221] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill, "Learning a SAT solver from single-bit supervision," presented at the Int. Conf. Learn. Represent. (ICLR), New Orleans, LA, USA, May 2019.
[222] D. Selsam and N. Bjørner, "Guiding high-performance SAT solvers with unsat-core predictions," in *Proc. Int. Conf. Theory Appl. Satisfiability Test.*, Lisboa, Portugal, Jul. 2019, pp. 336–353.
[223] H. Li, F. Jiao, and A. Doboli, "Analog circuit topological feature extraction with unsupervised learning of new sub-structures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2016, pp. 1509–1512.
[224] Y. Zhang, H. Ren, and B. Khailany, "GRANNITE: Graph neural network inference for transferable power estimation," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

Wenye Liu (Graduate Student Member, IEEE) received the B.S. degree in microelectronics from Shenzhen University, China, in 2014, the B.S. degree in physics from Umea University, Sweden, in 2014, and the M.S. degree in IC design engineering from The Hong Kong University of Science and Technology. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His current research interests include hardware security, machine learning accelerator, and fault injection attack.

Chip-Hong Chang (Fellow, IEEE) received the B.Eng. degree (Hons.) from the National University of Singapore in 1989, and the M.Eng. and Ph.D. degrees from Nanyang Technological University (NTU). Singapore, in 1993 and 1998, respectively.

He held joint appointments with NTU as the Assistant Chair of Alumni from 2008 to 2014, the Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and the Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He is currently an Associate Professor with the School of Electrical and Electronic Engineering (EEE), NTU. He has coedited five books, published 13 book chapters, more than 100 international journal articles (more than 70 are in IEEE), more than 180 refereed international conference papers (mostly in IEEE), and delivered over 40 colloquia. His current research interests include hardware security, machine learning security, unconventional computer arithmetic circuits, and low-power and fault-tolerant digital signal processing algorithms and architectures. He is an IET Fellow. He serves as the Senior Area Editor of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY. He is an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I and IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. He served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 2016 to 2019, IEEE ACCESS from 2013 to 2019, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I from 2010 to 2013, *Integration, the VLSI Journal* from 2013 to 2015, *Journal of Hardware and System Security* (Springer) from 2016 to 2020, and *Microelectronics Journal* from 2014 to 2020. He guest edited around ten Special Issues and served in the organizing and technical program committee of more than 60 international conferences (mostly IEEE). From 2018 to 2019, he was a Distinguished Lecturer of the IEEE Circuits and Systems Society.



Xueyang Wang received the B.S. degree in automation from Zhejiang University, Zhejiang, China, in 2008, and the M.S. and Ph.D. degrees in computer engineering and electrical engineering from the Tandon School of Engineering, New York University, Brooklyn, NY, USA, in 2010 and 2015, respectively. He currently works as an Offensive Security Researcher at Intel. He has published more than ten peer-reviewed papers in premier international conferences and journals and has served in technical and program committee of several international conferences. Besides academic publications, he also presented his research in industrial security conferences, such as Black Hat and Bsides. His research interests include secure architectures, virtualization and its application to security, hardware support for software security, and hardware security.

Chen Liu received the B.S. degree in communication engineering from Southeast University, Nanjing, China, in 2010, and the Ph.D. degree in computer engineering from the University of Delaware in 2016. He is currently an Offensive Security Researcher at Intel. He has authored/coauthored more than ten journal and conference papers and served as technical program committee of several international conferences. His current research interests include system security, non-volatile memory security and privacy, side-channel attacks, and mitigation.

Jason M. Fung received the two B.S. degrees in computer science and mathematics and in electrical and computer engineering and the M.S. degree in electrical and computer engineering from Carnegie Mellon University in 1997 and 1998, respectively. He is currently the Director of Offensive Security Research and Academic Research Engagement at Intel. He oversees the security assurance and emerging threat research of key technologies that power Intel's edge, communication, and data center products. In addition, he leads academic and industry collaborations that advance product security assurance best practices for the semiconductor industry. His recent contributions include the creation of the community-driven Hardware Common Weakness Enumeration (CWE) and the industry-first Hardware Capture-the-Flag Competitions (HackAtEvent.org) that inspire researchers to address some of the toughest challenges in hardware security. He has over two decades of industry experience in SoC architecture and performance, verification automation, product security penetration testing, consultation, research and path-finding, engineering, and risk management. Since 2013, he has been serving in steering committees and founded the security tracks for several premier conferences at Intel. He has authored five U.S. patents and over 20 publications in peer-reviewed academic, industry, and Intel internal conferences. He is a Founding Member of the CAPEC/CWE Advisory Board.

Mohammad Ebrahimabadi (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering from Zanjan University, Zanjan, Iran, in 2008, and the M.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2011. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Maryland Baltimore County, MD, USA. He is a member of the SECure, REliable and Trusted Systems (SECRETS) Research Laboraotry. His current research focus is on developing PUF-based authentication and secure communication protocols in IoT networks.

Naghmeh Karimi (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in Computer Engineering from the University of Tehran, Iran in 1997, 2002, and 2010, respectively. She was a visiting researcher at Yale University, USA between 2007 and 2009, and a post-doctoral researcher at Duke University, USA during 2011-2012. She has been a visiting assistant professor at New York University and Rutgers University between 2012 and 2016. She joined University of Maryland Baltimore County as an assistant professor in 2017 where she leads the SECure, REliable and Trusted Systems (SECRETS) research lab. She has published three book chapters and authored/co-authored more than 50 papers in referred conference proceedings and journal manuscripts. Her current research interests include hardware security, VLSI testing, design-for-trust, design-for-testability, and design-for-reliability. She has been serving as an Associate Editor for the Springer Journal of Electronic Testing: Theory and Applications (JETTA). She is a recipient of the National Science Foundation CAREER Award in 2020.

Xingyu Meng (Graduate Student Member, IEEE) received the B.E. degree in electronics science and technology from Nankai University in 2015, and the M.S. degree in system engineering from the University of Texas at Dallas, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering. His research interests include hardware and system security, Trojan detection, and hardware verification.

Kanad Basu (Senior Member, IEEE) received the Ph.D. degree from the Department of Computer and Information Science and Engineering, University of Florida. His thesis was focused on improving signal observability for post-silicon validation. During his Ph.D., he interned at Intel. After his Ph.D., he worked in various semiconductor companies like IBM and Synopsys. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, University of Texas at Dallas. Prior to this, he was an Assistant Research Professor with the Electrical and Computer Engineering Department, NYU. He has authored two U.S. patents, two book chapters, and several peer reviewed journal and conference papers. His current research interests are hardware and systems security. He was awarded the Best Paper Award at the International Conference on VLSI Design 2011.