

INVITED: Building Robust Machine Learning Systems: Current Progress, Research Challenges, and Opportunities

Jeff (Jun) Zhang^{*}, Kang Liu^{*}, Faiq Khalid[†], Muhammad Abdullah Hanif[†], Semeen Rehman[†],
Theocharis Theocharides[§], Alessandro Artusi[§], Muhammad Shafique[†], Siddharth Garg^{*}

^{*}New York University, U.S.A; [†] Technische Universität Wien (TU Wien), Austria; [§]University of Cyprus, Cyprus
{sg175,jeffjunzhang,kang.liu}@nyu.edu;{ttheocharides,aartus01}@ucy.ac.cy
{muhammad.hanif,faiq.khalid,semeen.rehman,muhammad.shafique}@tuwien.ac.at

ABSTRACT

Machine learning, in particular deep learning, is being used in almost all the aspects of life to facilitate humans, specifically in mobile and Internet of Things (IoT)-based applications. Due to its state-of-the-art performance, deep learning is also being employed in safety-critical applications, for instance, autonomous vehicles. Reliability and security are two of the key required characteristics for these applications because of the impact they can have on human's life. Towards this, in this paper, we highlight the current progress, challenges and research opportunities in the domain of robust systems for machine learning-based applications.

KEYWORDS

Machine Learning, Deep Learning, Reliability, Security, Robustness, Permanent Faults, Timing Errors, Adversarial Attacks.

1 INTRODUCTION

Machine learning (ML) has emerged as a leading tool for data analysis because of its ability to learn directly from raw data, with minimal human intervention. In particular, Deep Neural Networks (DNNs) offer state-of-the-art accuracy for many ML applications. Current research in ML is focused on improving state-of-the-art DNNs to develop learning algorithms which can help learn proper functionalities without bias and thereby can help improve the accuracy of ML-based systems. Moreover, as DNNs are inherently compute intensive, optimization methods are also being studied which can significantly reduce the computational complexity as well as the memory requirements of these algorithms. Apart from the aforementioned-objectives, the digital system design community is focusing on developing efficient hardware accelerators which can further boost the efficiency gains by designing application-specific hardware for DNN-based applications [25]. Moreover, driven by the current progress, DNNs are also being explored for use in safety-critical applications; for example, autonomous driving [8] and smart healthcare [7]. These applications

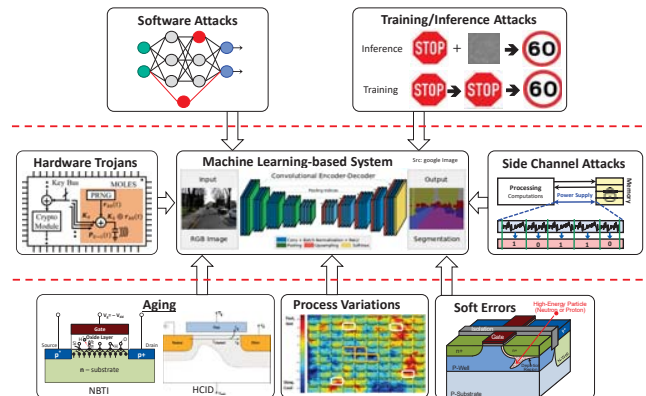


Figure 1: Reliability and security threats on machine learning-based systems. (Source of Images: [23, 24])

have stringent robustness constraints as defined by the standardization authorities because of the risks involved in the operation of these systems.

Robustness refers to two main characteristics of a system, i.e., resilience against *reliability* threats and against *security* vulnerabilities. Several reliability and security vulnerabilities are highlighted in Fig. 1, which are later discussed in Section 2 and 3.

In this paper, we discuss the current state-of-the-art related to DNN reliability and security. We also highlight the challenges which are being faced in building reliable and secure yet efficient DNNs.

2 RELIABLE MACHINE LEARNING

Reliability threats are mainly due to faults that arise at the hardware-layer of a system and can propagate all the way to the application-layer, potentially causing mis-predictions. There are many types of reliability faults, e.g., soft errors, timing faults, and permanent faults. Different techniques have been proposed for mitigating these faults. However, most of them are based on redundancy-based approaches where spatial/temporal redundancy is exploited for executing multiple instances of an application that vote to ensure the correctness of execution [29]. Based on the compute-intensive nature of DNNs, naively applying these approaches might obviate many of the gains obtained from hardware acceleration.

In this section, we highlight the impact of reliability faults on the accuracy of several state-of-the-art networks for different datasets. The details of the networks and the datasets are presented in Table 1. Later in the section, we also discuss a few methods which can be used for efficiently mitigating the permanent faults and timing errors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DAC '19, June 2–6, 2019, Las Vegas, NV, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6725-7/19/06...\$15.00

<https://doi.org/10.1145/3316781.3323472>

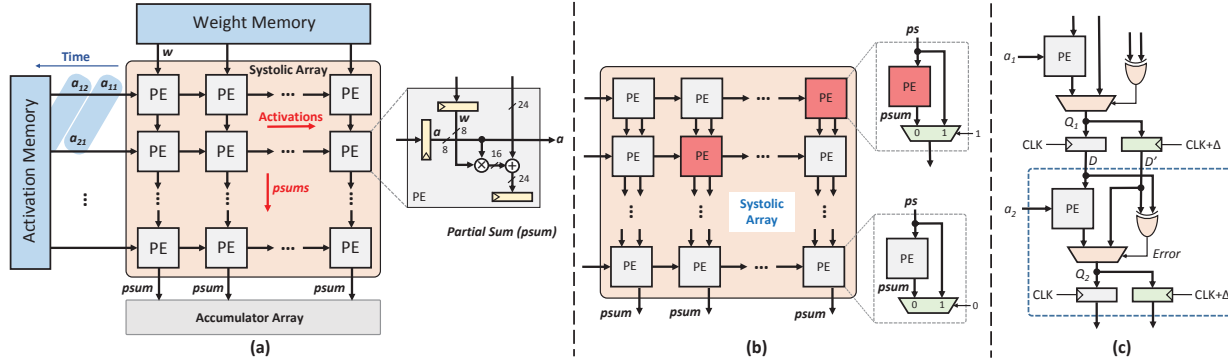


Figure 2: (a) Architecture of a systolic array-based DNN accelerator that serves as a baseline; (b) Modified systolic array for permanent fault mitigation; (c) Architectural modifications for mitigating timing faults. (Adapted from [32, 33])

Table 1: Datasets and the corresponding 8-bit DNNs used for evaluation. (Adapted from [32])

Dataset	Network Architecture	Accuracy(%)
MNIST [18]	Fully-Connected (L1-L4): $784 \times 256 \times 256 \times 256 \times 10$	98.15
TIMIT [2]	Fully-Connected (L1-L4): $1845 \times 2000 \times 2000 \times 2000 \times 183$	73.91
ImageNet [5]	Convolutional (L1-L2): $(224, 224, 3) \times (27, 27, 64) \times (13, 13, 192)$	76.33 (Top-5)
	Convolutional (L3-L5): $(13, 13, 384) \times (13, 13, 256) \times (6, 6, 256)$	
	Fully-Connected (L6-L8): $4096 \times 4096 \times 1000$	

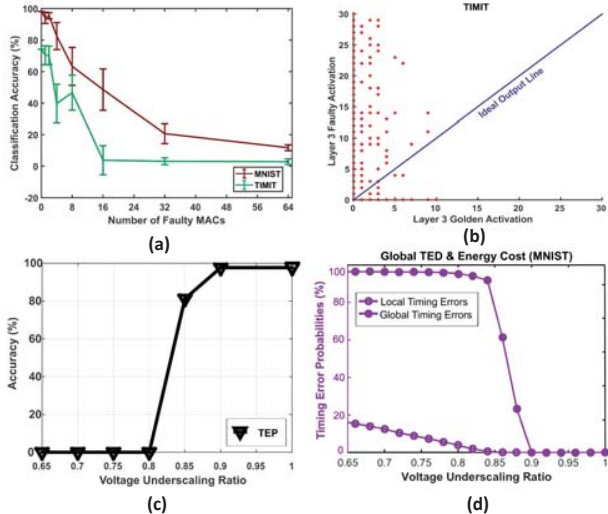


Figure 3: (a) Classification Accuracy in the Presence of Stuck-at-Faults in MAC units; (b) Impact of TPU Stuck-at-Faults on TIMIT’s activations; (c) Impact of timing error propagation on the classification accuracy on MNIST; (d) Local and Global timing error rates versus voltage underscaling in the TPU-based architecture. (Adapted from [32, 33])

which mainly arise because of process variations [22] and aging [28]. All the case-studies and analyses are performed using a baseline Tensor Processing Unit (TPU) [14] like accelerator. A block diagram of the accelerator is shown in Fig. 2(a). More details about the architecture can be found in [32].

2.1 Resilience of DNNs to Reliability Faults

Previous works have suggested that ML-based applications are inherently error-resilient [9]. Therefore, to study the impact of permanent and timing faults on the output of DNNs, we empirically evaluated the baseline hardware shown in Fig. 2(a) consisting of a large systolic array of multiplier-and-accumulate (MAC) units.

2.1.1 Permanent Faults. For this analysis, we synthesized an RTL implementation of the systolic array (shown in Fig. 2(a)) and generated its gate-level netlist. We then injected stuck-at-faults at the internal nodes in the netlist. Next, we mapped DNNs for two different classification tasks, i.e., digit classification using MNIST dataset and speech recognition using TIMIT dataset, on the “faulty” baseline TPU. Fig. 3(a) shows the impact of faulty MAC units on the DNN classification accuracy. For TIMIT, we observed that even with only 4 faulty MAC units (our baseline has 65 K MACs in total), the classification accuracy from the 74.13% to 39.69%.

The reason behind this drop in accuracy is that faults frequently affect higher order bits of the MAC output, which results in large errors in the matrix-matrix/matrix-vector product. This is illustrated with the help of Fig. 3(b) where the outputs computed using faulty hardware are (in most cases) larger than the expected output.

2.1.2 Timing Errors. For this analysis, we naively allowed timing errors to propagate (i.e., Timing Error Propagation (TEP) approach) to subsequent stages of computation [31]. Fig. 3(c) shows the classification accuracy for the MNIST benchmark as a function of the extent of voltage when errors are allowed to propagate through the systolic array. We note that as soon as timing errors start appearing, as shown in Fig. 3(d), the classification accuracy for TEP drops quickly.

2.2 Mitigation techniques

2.2.1 Permanent Fault Mitigation. We proposed two techniques in [33]: (1) *fault-aware pruning* (FAP); and (2) *fault-aware pruning plus retraining* (FAP+T). The techniques enable the baseline TPU to operate even at the fault rates upto 50% without significantly affecting the classification accuracy of the DNNs. The proposed techniques are based on the idea that a significant fraction of the overall connections in a DNN can be *pruned* away without significantly affecting the DNN accuracy. However, while the prior works use pruning to reduce the memory and the computational requirements of DNNs [11], we use pruning to enable hardware *fault tolerance*. FAP prunes all the network connections (network parameters) that map to faulty PEs by enabling simple bypass circuitry shown in Fig. 2(b). The bypass circuitry requires only minor modifications in the baseline accelerator and also does not have much overheads. In FAP+T we additionally *retrain* the DNN after pruning in order to regain the lost accuracy. One drawback of FAP+T is that the fault map of each TPU chip can be different, therefore, the retraining of a DNN has to be performed for each TPU chip which comes at the cost of increased “test time” per chip. Note that the FAP+T technique assumes static mapping policy.

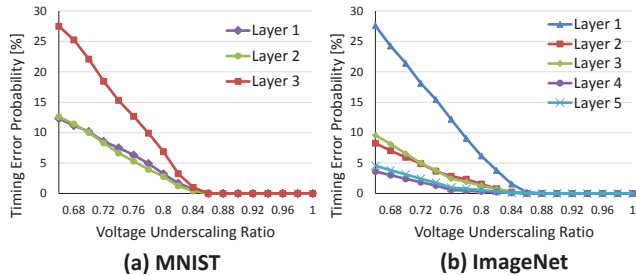


Figure 4: Timing error probabilities for each layer for two of the networks presented in Table 1. (Adapted from [32, 34])

i.e., one weight is mapped to only one specific PE while multiple weights can be mapped to the same PE.

2.2.2 Timing Fault Mitigation. We proposed a novel technique (TE-Drop) in [32] to mitigate timing errors in MAC units of a TPU-like accelerator. TE-Drop equips each MAC unit with Razor flip-flops [6] to detect timing errors, however, it mitigates the errors *without* re-executing erroneous computations.

TE-Drop is built on the observation that the contribution of each individual MAC computation to the output of a neuron is small [11]. When a MAC unit incurs a timing error, TE-Drop steals the next clock cycle from its successor MAC unit to correctly add its contribution to the partial sum, and *bypasses/drops* the subsequent MAC operation. To support this functionality, TE-Drop also requires a multiplexer (MUX) which is controlled by the error signal from the previous (upstream) MAC unit. If the previous MAC unit incurs a timing error, the MUX forwards the previous MAC’s correctly computed partial sum to the next MAC unit; otherwise, the current MAC unit updates the partial sum and forwards its output to the next MAC. As shown in Fig. 2(c), TE-Drop requires minimal hardware changes in the baseline TPU.

Fig. 4 illustrates the timing error rate for each layer of two DNNs as a function of the voltage underscaling ratio. In both the networks, we observe that the timing errors vary significantly across layers of a network. Therefore, based on this analysis, we proposed a *per-layer voltage underscaling scheme* in [32] which distributed the available timing error budget equally among layers to achieve high energy efficiency.

3 SECURE MACHINE LEARNING

Similar to the traditional computing systems, DNN-based systems are also vulnerable to several security threats, i.e., data manipulation [12, 16], model/IP stealing [30], and denial-of-service attacks [12]. However, to handle these vulnerabilities in such relatively complicated systems raises its own challenges. Therefore, in the subsequent subsections, we present the different security attacks and their respective countermeasures.

3.1 Security Attacks on DDNs

DNN-based classifiers are vulnerable to specially crafted adversarial noise patterns in inputs. These noise patterns can either perform targeted or untargeted misclassification or confidence reduction (which can make DNNs even more vulnerable to reliability threats). Two of the most dangerous attacks are adversarial perturbations and backdooring attacks.



Figure 5: Original images of MNIST digit "7" and US "stop sign", and their respective backdoored images.

3.1.1 Adversarial attacks. These attacks targets inference of DNNs and assume that an attacker does not have access to the training process including the dataset but have access to the trained DNN in both black-box and white-box settings. Therefore, these attacks generate imperceptible noise either by exploiting the gradient (e.g., Fast Gradient Sign (FGS), Iterative Fast Gradient Sign (IFGS) and Jacobian-based saliency map attack (JSMA) methods [21]) or using more complex optimization procedures (Limited Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [26] and Carlini & Wagner (CW) [3]). For example, a recently proposed attack, TriSec [17], exploits gradient information and optimization of back-propagating effects of the targeted label to the inputs. However, generating the attack image using traditional adversarial attack methodology requires larger number of inferences and output probabilities, therefore, these attacks may fail in resource-constrained DNN-based systems, i.e., autonomous vehicles. To address these limitations, resource-efficient decision-based (RED-Attack [16]) attacks have been proposed, which leverages the half-interval search algorithm to find an appropriate attack image along the learned classification boundary.

3.1.2 Backdoor or neural Trojan attacks. Unlike the adversarial attacks, these attacks targets the outsourced ML training where attacker has the access to training procedure and train the DNN for a well crafted high density perceptible noise pattern (act as a hidden backdoor) while maintaining the validation accuracy and performing well on its intended task. So, whenever this *backdoor trigger* is present in the input DNN performs targeted or random misclassifications [10][19]. In targeted misclassification this backdoor is associated with a target label but in random misclassification it reduces the confidence or classification accuracy for backdoored inputs. An illustration of this attack on clean MNIST digit and the US stop sign and respective backdoor images is shown in Fig. 5.

To achieve her goal, attacker can make modifications in the training procedure, like augmenting the training data with target samples and labels (*training set poisoning* [13]), modifying the configuration settings of the training algorithm, i.e., learning rate or the batch size, or even directly modifying the trained network parameters (Θ).

Moreover, DNNs are also vulnerable to model stealing attacks [30] that extract or estimate the behavior of the DNN IP (trained IP) to breach IP copyrights and eventually leads to financial loss. Due to recent advancement in ML-based systems, several novel and efficient DNN-algorithms, i.e., spiking neural networks and capsule nets, have been proposed. However, these NNs are also vulnerable to adversarial perturbations, as shown in the analysis in [20].

3.2 Defenses Against Security Attacks on DDNs

To address the above-mentioned security attacks, i.e., adversarial perturbation and backdooring attacks, several countermeasures have been proposed and some of them are discussed below.













	Case I	Case II	Case III	Case IV	
L-BFGS					Case I: Classification of the Original Samples (without any attacks)
	99.47%	85.68%	72.74%	78.64%	Case II: Classification of the Perturbed Images without preprocessing filters or attacker has the access to output of the preprocessing filters
FGSM					Case III: Classification of the Perturbed Images when attacker do not have the access to output of the preprocessing filters
	99.47%	75.68%	78.45%	68.45%	Case IV: Classification of the Perturbed Images (after incorporating the preprocessing) when attacker do not have access to the preprocessing filters
BIM					Confidence: Stop → Stop Sign
	99.47%	89.68%	70.39%	85.64%	Confidence: Stop → Speed 60 km/h

Figure 6: Impact of the preprocessing filtering on the adversarial attacks (i.e., L-BFGS, FGSM, BIM) with different attack models with and without the access of filters. (adapted from [15])

3.2.1 Defenses against Adversarial attacks. Typically these attacks exploit the gradient, therefore to protect the gradient of DNN, some countermeasures like *DNN masking*, *gradient masking* and *adversarial training* based defenses have been proposed but they are either limited to known attacks or can be breached by modifying the optimization function. Another defense is to perform *preprocessing* (e.g., *quantization* [1], *filtering* [27]) of the CNN inputs [4] which can increase the perceptibility of attack noise or reduces it overall effects. For examples, analysis in [15] shows that *low-pass pre-processing filters can nullify adversarial attacks if unknown to the attacker*.

3.2.2 Defenses against Backdoor Attacks. Typically, since backdoors exploit the spare capacity in DNNs [10], *pruning* is a natural defense. The pruning-based defense reduces the size of the backdoored network by eliminating dormant neurons, that either limits or disables the backdoor behaviour. Although the pruning defense successfully nullifies all naive backdoor attacks, a “pruning-aware” attack (under the assumption that attacker is aware of the pruning) can break the pruning-based defense by mapping the clean and backdoor behaviour onto the same set of neurons which remain active in most of the cases. Finally, to defend against such pruning-aware attacks, a defender can perform the local pruning-aware retraining on a small clean training dataset (*fine-tuning*) which effectively disables backdoors.

4 FUTURE RESEARCH DIRECTIONS

Frameworks for Studying Error-Resilience: Empirical analysis, although provides significantly accurate results, takes a considerable amount of time to analyze the impact of specific reliability vulnerabilities. Therefore, there is a dire need to develop frameworks which can precisely and efficiently simulate the effects of reliability threats. These frameworks are envisioned to be highly useful for studying the impact of concurrent reliability threats.

Design Methods for Building Robust DNNs: Significant research has been carried out in developing energy- and performance-efficient DNNs. However, only a limited number of methods are available which allow to cater the effects of reliability threats. Towards this, analysis of different network architectures is required, and methods have to be developed which can build inherently resilient DNNs.

Robust ML Training: Most of the traditional ML training algorithms focus on achieving high accuracy for specific application/dataset which make the ML-based systems vulnerable to several security and reliability threats. Although, several algorithms have been developed to assist the training with respect to adversarial perturbation, but their

scope is limited the known adversarial perturbations. Therefore, such ML-training algorithms are required which can ensure the robustness, in terms of safety, security, privacy and reliability.

ACKNOWLEDGEMENT

A part of this work was funded by NSF Grant 1801495.

REFERENCES

- [1] Hassan Ali et al. 2018. QuSecNets: Quantization-based Defense Mechanism for Securing Deep Neural Network against Adversarial Attacks. *arXiv:1811.01437* (2018).
- [2] Jimmy Ba et al. 2014. Do Deep Nets Really Need to be Deep? In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2654–2662.
- [3] Nicholas Carlini et al. 2016. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644* (2016).
- [4] Anirban Chakraborty et al. 2018. Adversarial Attacks and Defences: A Survey. *arXiv:1810.00069* (2018).
- [5] J. Deng et al. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*. 248–255. <https://doi.org/10.1109/CVPR.2009.5206848>
- [6] Dan Ernst et al. 2004. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro* 24, 6 (2004), 10–20.
- [7] Andre Esteva et al. 2019. A guide to deep learning in healthcare. *Nature medicine* 25, 1 (2019), 24.
- [8] Maximilian Fink et al. 2019. Deep Learning-Based Multi-scale Multi-object Detection and Classification for Autonomous Driving. In *Fahrerassistenzsysteme*. Springer.
- [9] Anteneh Gebregiorgis et al. 2017. Error propagation aware timing relaxation for approximate near threshold computing. In *DAC*. ACM, 77.
- [10] Tianyu Gu et al. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv:1708.06733* (2017).
- [11] Song Han et al. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv:1510.00149* (2015).
- [12] Muhammad Abdullah Hanif et al. 2018. Robust Machine Learning Systems: Reliability and Security for Deep Neural Networks. In *IOLTS*. IEEE, 257–260.
- [13] Ling Huang et al. 2011. Adversarial machine learning. In *AISEC*. ACM, 43–58.
- [14] Norman P Jouppi et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *ISCA*. IEEE, 1–12.
- [15] Faiq Khalid et al. 2019. FAdEML: Understanding the Impact of Pre-Processing Noise Filtering on Adversarial Machine Learning. In *DATE*. IEEE.
- [16] Faiq Khalid et al. 2019. RED-Attack: Resource Efficient Decision based Attack for Machine Learning. *arXiv:1901.10258* (2019).
- [17] Faiq Khalid et al. 2019. TriSec: Training Data-Unaware Imperceptible Security Attacks on Deep Neural Networks. *arXiv:1811.01031* (2019).
- [18] Y. Lecun et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov 1998), 2278–2324.
- [19] Kang Liu et al. 2018. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *RAID*. Springer, 273–294.
- [20] Alberto Marchisio et al. 2019. SNN under Attack: are Spiking Deep Belief Networks vulnerable to Adversarial Examples? *arXiv:1902.01147* (2019).
- [21] Nicolas Papernot et al. 2016. The limitations of deep learning in adversarial settings. In *EuroS&P*. IEEE, 372–387.
- [22] Bharathwaj Raghunathan et al. 2013. Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In *DATE*. IEEE, 39–44.
- [23] Semeen Rehman, Muhammad Shafique, and Jörg Henkel. 2016. *Reliable Software for Unreliable Hardware: A Cross Layer Perspective*. Springer.
- [24] Muhammad Shafique et al. 2014. The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives. In *DAC*. ACM, 1–6.
- [25] Vivienne Sze et al. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.
- [26] Christian Szegedy et al. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
- [27] Hammad Tariq et al. 2018. SSCNets: A Selective Sobel Convolution-based Technique to Enhance the Robustness of Deep Neural Networks against Security Attacks. *arXiv:1811.01443* (2018).
- [28] Abhishek Tiwari et al. 2008. Facelift: Hiding and slowing down aging in multicores. In *MICRO*. IEEE Computer Society, 129–140.
- [29] Ramakrishna Vadlamani et al. 2010. Multicore soft error rate stabilization using adaptive dual modular redundancy. In *DATE*. IEEE, 27–32.
- [30] Binghui Wang and Neil Zhenqiang Gong. [n. d.]. Stealing hyperparameters in machine learning. In *IEEE S&P*.
- [31] Paul N Whatmough et al. 2013. Circuit-level timing error tolerance for low-power DSP filters and transforms. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 6 (2013), 989–999.
- [32] Jeff Zhang et al. 2018. Thundervolt: Enabling Aggressive Voltage Underscaling and Timing Error Resilience for Energy Efficient Deep Learning Accelerators. In *DAC*.
- [33] Jeff Jun Zhang et al. 2018. Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In *VTS*. IEEE, 1–6.
- [34] Jeff Jun Zhang et al. 2018. FATE: fast and accurate timing error prediction framework for low power DNN accelerator design. In *ICCAD*. ACM.