

Towards Energy-Efficient and Secure Edge AI: A Cross-Layer Framework

ICCAD Special Session Paper

Muhammad Shafique*, Alberto Marchisio†, Rachmad Vidya Wicaksana Putra†, Muhammad Abdullah Hanif†*

*New York University Abu Dhabi (NYUAD), Abu Dhabi, United Arab Emirates

†Technische Universität Wien (TU Wien), Vienna, Austria

muhammad.shafique@nyu.edu, {alberto.marchisio, rachmad.putra}@tuwien.ac.at, mh6117@nyu.edu

Abstract—The security and privacy concerns along with the amount of data that is required to be processed on regular basis has pushed processing to the edge of the computing systems. Deploying advanced Neural Networks (NN), such as deep neural networks (DNNs) and spiking neural networks (SNNs), that offer state-of-the-art results on resource-constrained edge devices is challenging due to the stringent memory and power/energy constraints. Moreover, these systems are required to maintain correct functionality under diverse security and reliability threats. This paper first discusses existing approaches to address energy efficiency, reliability, and security issues at different system layers, i.e., hardware (HW) and software (SW). Afterward, we discuss how to further improve the performance (latency) and the energy efficiency of Edge AI systems through HW/SW-level optimizations, such as pruning, quantization, and approximation. To address reliability threats (like permanent and transient faults), we highlight cost-effective mitigation techniques, like fault-aware training and mapping. Moreover, we briefly discuss effective detection and protection techniques to address security threats (like model and data corruption). Towards the end, we discuss how these techniques can be combined in an integrated cross-layer framework for realizing robust and energy-efficient Edge AI systems.

Index Terms—Artificial intelligence, machine learning, Edge AI, deep neural networks, spiking neural networks, accuracy, latency, energy efficiency, reliability, security, robustness, Edge computing, tinyML.

I. INTRODUCTION

The NN-based AI systems have achieved state-of-the-art accuracy for various applications such as image classification, object recognition, healthcare, automotive, and robotics [1]. However, current trends show that the accuracy is improved at the cost of increasing complexity of NN models (e.g., larger model size and complex operations) [2] [3]. This increased complexity hinders the deployment of advanced NNs (DNNs and SNNs) on resource-constrained edge devices [4]. Therefore, optimizations at different system layers (i.e., HW and SW) are necessary to enable the use of advanced NNs at the edge [2]. Besides performance and energy efficiency, reliability and security aspects are also important to ensure the correct functionality under diverse operating conditions (e.g., in the presence of HW faults and security threats), especially for safety-critical applications like autonomous driving and healthcare [5] [6]. Therefore, *the important design metrics for enabling Edge AI include performance (i.e., latency), energy efficiency, reliability, and security.*

A. Key Challenges for Energy-Efficient and Secure Edge AI

We introduce the key challenges for developing Edge AI systems in the following text (see Fig. 1 for an overview of the challenges).

- **Performance:** Edge AI systems are expected to have high performance to provide real-time response. However, due to memory- and compute-intensive nature of NNs, achieving high performance is not trivial. Moreover, edge devices have limited compute and memory resources, which makes it challenging to map the full NN computations simultaneously on an accelerator fabric [7].

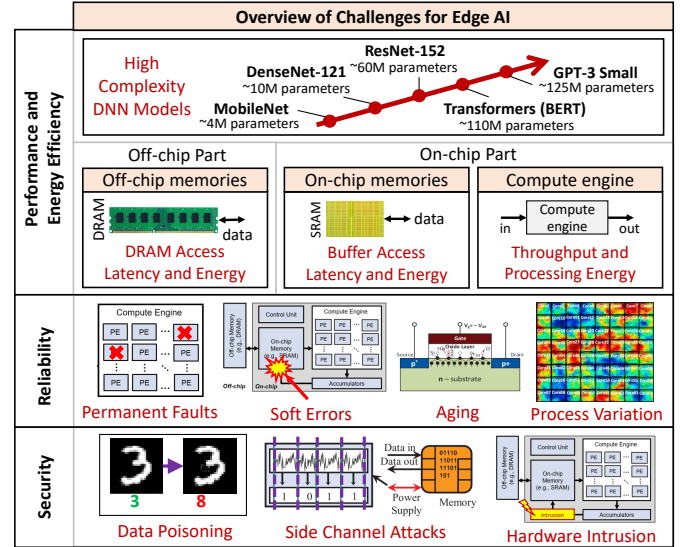


Fig. 1. Overview of challenges for energy-efficient and secure Edge AI.

- **Energy Efficiency:** Edge AI systems should also have high energy efficiency to ensure complete processing within a restricted energy budget, especially in the case of battery-powered devices. Therefore, the energy consumption in both the off-chip and on-chip parts should be minimized. The off-chip part includes the DRAM-based off-chip memory accesses [7], while the on-chip part includes (1) the on-chip memory accesses, and (2) the neural operations like multiply-and-accumulation (MAC) [8].
- **Reliability:** Edge AI systems should produce correct outputs even in the presence of different types of reliability threats [9]. The main reliability threats are as follows.
 - **Process variations** are the result of imprecisions in the fabrication process, as manufacturing billions of nano-scale transistors with identical electrical properties is difficult to impossible. This causes variations in the leakage power and frequency in the same chip, across different chips in the same wafer, and even across different wafers [10].
 - **Soft errors** are caused by high-energy particle strikes, manifest as bit-flips at the HW layer, and can propagate all the way to the application layer and may cause incorrect outputs [11].
 - **Aging** is gradual degradation of the processing circuits over time [12]. It occurs due to physical phenomena like Hot Carrier Injection (HCI), Time-Dependent Dielectric Breakdown (TDDB), and Negative/Positive Bias Temperature Instability (NBTI/PBTI).
- **Security:** Edge AI systems should offer high resilience against security vulnerabilities such as side channels and HW intrusions [9]. Moreover, NN algorithms (e.g., DNNs) have other

security vulnerabilities as well that can be exploited through data poisoning to cause confidence reduction or misclassification [13].

The above discussion highlights different possible challenges for developing energy-efficient and secure Edge AI systems. To address each challenge individually, various techniques have been proposed at different layers of the computing stack. *However, systematic integration of the most effective techniques from both the hardware and software levels is important to achieve ultra-efficient and secure Edge AI.*

B. Our Contributions

In the light of the above discussion, the **contributions** of this paper are the following.

- We present an overview of different challenges and state-of-the-art techniques for improving performance and energy efficiency of Edge AI systems (**Section II**).
- We present an overview of different challenges and state-of-the-art techniques for reliability and security of Edge AI (**Section III**).
- We present a cross-layer framework that systematically integrates the most effective techniques for improving the energy efficiency and robustness of Edge AI (**Section IV**).
- We discuss the challenges and recent advances in neuromorphic computing considering SNNs (**Section V**).

II. PERFORMANCE AND ENERGY-EFFICIENT EDGE AI

In the quest of achieving higher accuracy, the evolution of DNNs has seen a dramatic increase in the complexity with respect to model size and operations, i.e., from simple Multi-Layer Perceptron (MLP) to deep and complex networks like Convolutional Neural Networks (CNNs), Transformers, and Capsule Networks (CapsNet) [14]. Hence, *the advanced DNNs require specialized hardware accelerators and optimization frameworks to enable efficient and real-time data processing at the edge*. To address this, a significant amount of work has been carried out in the literature. In this section, we discuss different state-of-the-art techniques for improving performance and energy efficiency of Edge AI (see overview in Fig. 2).

A. Optimizations for DNN Models

The edge platforms typically have limited memory and power/energy budgets, hence small-sized DNN models with limited number of operations are desired for Edge AI applications. Model compression techniques such as pruning (i.e., structured [15] [16] or unstructured [17]–[19]) and quantization [19]–[22] are considered to be highly effective for reducing the memory footprint of the models as well as for reducing the number of computations required per inference. Structured pruning [15] can achieve about 4x weight memory compression, while class-blind unstructured pruning (i.e., PruNet [18]) achieves up to 190x memory compression. Quantization when combined with pruning can further improve the compression rate. For instance, quantization in the Deep Compression [19] improves the compression rate by about 3x for the AlexNet and the VGG-16 models. The Q-CapsNets framework [22] shows that quantization is highly effective for complex DNNs such as CapsNets as well. It reduces the memory requirement of the CapsNet [14] by 6.2x with a negligible accuracy degradation of 0.15%. Since model compression techniques may result in a sub-optimal accuracy-efficiency trade-off (due to lack of information of the underlying hardware architecture used for DNN execution), HW-aware model generation and compression techniques have emerged as a potential

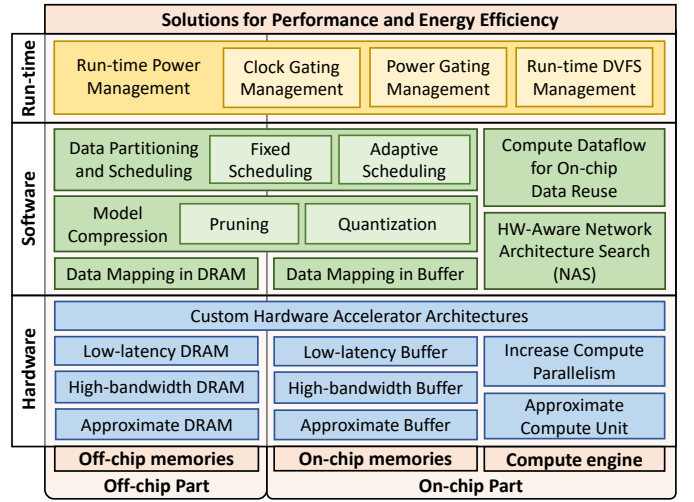


Fig. 2. Overview of the techniques at different system layers for improving the performance and energy efficiency of Edge AI.

solution. Many Neural Architecture Search (NAS) techniques [23]–[29] have been proposed to generate high accuracy and efficient models. The state-of-the-art NAS like the APNAS framework [28] employs an analytical model and a reinforcement learning engine to quickly find DNNs with good accuracy-efficiency trade-offs for the targeted systolic array-based HW accelerators [30] [31]. It reduces the compute cycles by 53% on average with negligible accuracy degradation (avg. 3%) compared to the state-of-the-art techniques. Therefore, it is suitable for generating DNNs for resource-constrained applications. Meanwhile, the NASCaps framework [29] employs an NSGA-II algorithm to find Pareto-optimal DNN models by leveraging the trade-off between different hardware characteristics (i.e., memory, latency, and energy) of a given HW accelerator. Compared to manually designed state-of-the-art CapsNets (i.e., DeepCaps), the NASCaps achieves 79% latency reduction, 88% energy reduction, 63% memory reduction, with only 1% accuracy reduction.

B. Optimizations for DNN Accelerators

To efficiently run the generated DNN models on accelerator fabric, optimizations should be applied across the HW architecture, i.e., in the off-chip memory, on-chip memory, and on-chip compute engine.

The Off-chip Memory (DRAM): The main challenges arise from the fact that a full DNN model usually cannot be mapped and processed at once on the accelerator fabric due to limited on-chip memory. Therefore, redundant accesses for the same data to DRAM are required, which restricts the systems from achieving high performance and energy efficiency gains, as DRAM access latency and energy are significantly higher than other operations [32]. Toward this, previous works have proposed (1) model compression through pruning [15]–[19] and quantization [19] [20]–[22], and (2) data partitioning and scheduling schemes [33]–[36]. However, they do not study the impact of DRAM accesses which dominate the total system latency and energy, and do not minimize redundant accesses for overlapping data in convolutional operations. To address these limitations, several SW-level techniques have been proposed (the ROMANet [7] and DRMap [37] methodologies). Our ROMANet [7] minimizes the DRAM energy consumption through a design space exploration (DSE) that finds the most effective data partitioning and scheduling while considering redundant access optimization. It minimizes the average DRAM energy-per-access by avoiding row buffer conflicts and misses through an effective DRAM mapping,

as shown in Fig. 3. Our DRMap [37] further improves the DRAM latency and energy for DNN processing considering different DRAM architectures such as the low-latency DRAM with subarray-level parallelism (i.e., SALP [38]). It employs a DSE with a generic DRAM data mapping policy that maximizes DRAM row buffer hits, bank- and subarray-level parallelism to obtain minimum energy-delay-product (EDP) of DRAM accesses for the given DRAM architectures and DNN data partitioning and scheduling (see Fig. 4).

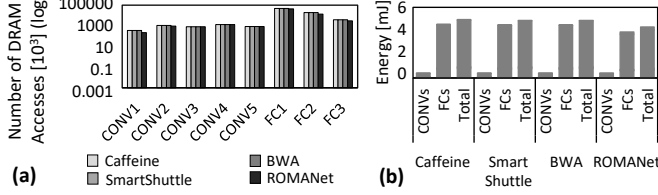


Fig. 3. Experimental results of (a) the number of DRAM accesses, and (b) the DRAM access energy for the AlexNet. The ROMANet [7] decreases the DRAM accesses and the DRAM energy compared to the state-of-the-art (i.e., the Caffeine [34], the SmartShuttle [35], and the BWA [36]).

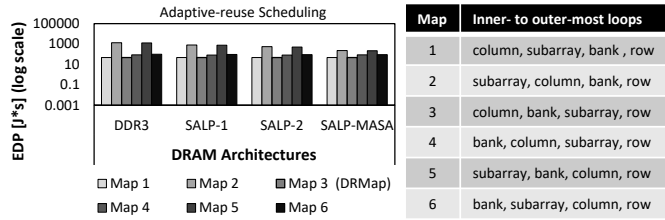


Fig. 4. The EDP of DRAM accesses for the AlexNet across different DRAM architectures (i.e., DDR3, SALP-1, SALP-2, and SALP-MASA) and different DRAM mapping policies, which have different orders of DRAM mapping loops. The results show that the DRMap mapping (i.e., Map 3) consistently obtains the lowest EDP [37].

The On-chip Memory (Buffer): To efficiently shuttle data between the DRAM and the on-chip fabric, specialized on-chip buffer design and access management are important. Here, the scratchpad memory (SPM) design is commonly used due to its low latency and power characteristics [7] [39]. For optimizing buffer access latency and energy, several SW-level techniques have been proposed (such as ROMANet [7] and DESCNet [8]). Our ROMANet framework [7] exploits the bank-level parallelism in the buffer to minimize latency and energy of the given buffer access requests, as shown in Fig. 5. Meanwhile, our DESCNet framework [8] searches for different on-chip memory architectures to reduce the energy consumption, and performs run-time memory management to power-gate the unnecessary memory blocks for non-memory-intensive operations. These optimizations provide up to 79% energy savings for CapsNet inference.

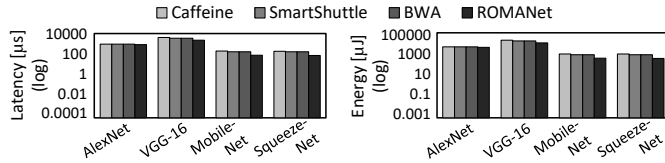


Fig. 5. Experimental results of the buffer access latency and energy across different optimization techniques and different networks. The ROMANet [7] effectively reduces the buffer latency and energy over the state-of-the-art (i.e., the Caffeine [34], SmartShuttle [35], and BWA [36] techniques).

The Compute Engine (Computational Units): The state-of-the-art HW-level optimization techniques (e.g., approximate computing) can provide significant area, performance and energy efficiency improvements, but at the cost of output quality degradation, which cannot be tolerated in safety-critical applications. Toward this, we

proposed the concept of curable approximations in [40], which ensures minimal accuracy degradation by employing approximations in a way that approximation errors from one stage are compensated in the subsequent stage/s of the pipeline. When used for improving the efficiency of compute engine with cascaded processing elements (PEs), like the systolic array in the TPU [31], it reduces the Power-Delay Product (PDP) of the array by about 46% and 38% compared to the conventional and approximate systolic array design, respectively. To efficiently employ approximations in applications that can tolerate minor quality degradation, a systematic error analysis is necessary to identify the approximation knobs and the degree to which each type of approximation can be employed. Toward this, several methodologies have been proposed to analyze the error resilience of CNNs [41] and CapsNets (i.e., ReD-CaNe [42]). By modeling the effects of approximations, it is possible to identify the optimal approximate components (e.g., adders and multipliers) that offer the best accuracy-efficiency trade-off while meeting the user-defined constraints. Compared to having accurate hardware, an efficient design that employs a layer-wise selection of approximate multipliers can achieve 28% energy reduction [42]. Furthermore, to find the configurations that offer good accuracy-energy trade-offs, the ALWANN framework [43] performs a DSE with a multi-objective NSGA-II algorithm.

Run-time Optimizations: Several run-time power management techniques can be employed to further boost the efficiency, e.g., the run-time clock gating, power gating, and dynamic voltage and frequency scaling (DVFS) techniques. For instance, the DESCNet technique [8] partitions the SPM into multiple sectors, and performs sector-level power-gating based on the characteristics of CapsNet workload to get high energy savings at run time during inference. Compared to the standard memory designs, the application-driven memory organizations equipped with memory power management unit in the DESCNet can save up to 79% energy and 47% area.

III. IMPROVING RELIABILITY AND SECURITY FOR EDGE AI

The Edge AI systems need to continuously produce correct outputs under diverse operating conditions. This requirement is especially important for safety-critical applications such as medical data analysis and autonomous driving. There are mainly two categories of vulnerability issues that threaten the Edge AI: (1) *reliability* and (2) *security*. In this section, we discuss the state-of-the-art techniques for improving the reliability and security of Edge AI (see an overview in Fig. 6).

A. Reliability Threats and Mitigation Techniques

Reliability threats may come from various sources like *process variation*, *soft errors*, and *aging*. They can manifest as *permanent faults* (faults that remain in the system permanently and do not disappear), *transient faults* (faults that occur once and can disappear), or *performance degradation* (e.g., in the form of delay/timing errors). To address these threats, conventional fault-mitigation techniques for VLSI can be employed, e.g., Dual Modular Redundancy (DMR) [44], Triple Modular Redundancy (TMR) [45], and Error Correction Code (ECC) [46]. However, these techniques incur huge overheads due to redundant hardware or execution. Hence, *cost-effective techniques are required to mitigate the reliability threats in the Edge AI*.

Permanent Faults: To mitigate permanent faults in DNN accelerators, recent works have proposed techniques like fault-aware pruning (FAP) [47] and fault-aware training (FAT) [47] [48]. They aim at making DNNs resilient to the faults by incorporating the information of faults in the optimization/training process. These

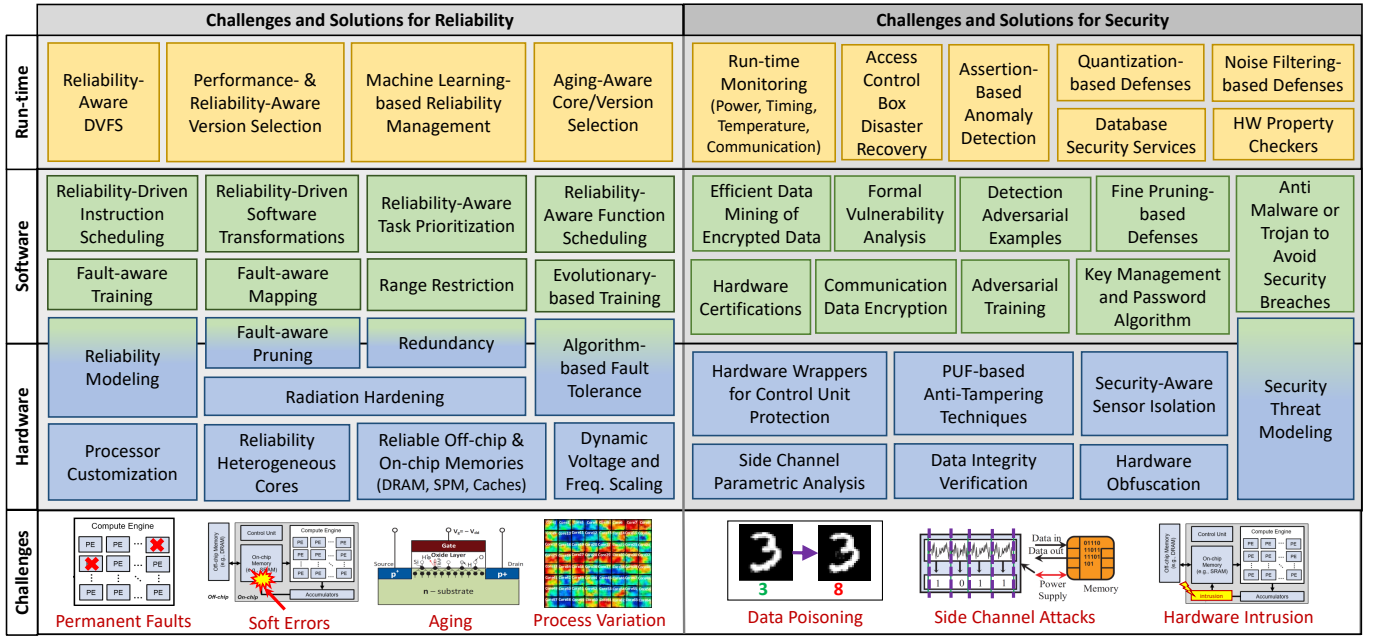


Fig. 6. Overview of challenges for reliability and security aspects, and the respective solutions on different system layers.

techniques usually require minor modifications at the hardware level (i.e., additional circuitry) to bypass/disconnect the faulty components, which results in minor run-time overheads. The key limitation of FAT is that it incurs a huge retraining cost, specifically for the cases in which retraining has to be performed for a large number of faulty chips. Moreover, FAT cannot be employed if the training dataset is not available to the user. To address these limitations, we proposed SalvageDNN [49] that enables us to mitigate permanent faults in DNN accelerators without retraining. It achieves this through a significance-driven fault-aware mapping (FAM) strategy, and shuffling of parameters at the software level to avoid additional memory operations. Techniques like FT-ClipAct [50] and Ranger [51] employ range restriction functions to block large (abnormal) activation values using pre-computed thresholds. Range restriction is realized using clipped activation functions that map out of the range values to pre-specified values within the range that have the least impact on the output. FT-ClipAct [50] shows that such techniques can improve the accuracy of the VGG-16 by 68.92% (on average) at 10^{-5} fault rate compared to the no fault mitigation case.

Transient Faults (Soft Errors): Soft error rates have been increasing in HW systems [52]. To mitigate their negative impact, several techniques have been proposed [51], [53]–[57]. Some of these techniques only cover limited faults [55] and/or incur significant overheads [53] [56]. For instance, techniques in [53] employ a separate network to detect the anomaly in the output. Other state-of-the-art techniques employ online SW-level range restriction functions, like Ranger [51] that rectifies the faulty outputs of DNN operations without re-computation by restricting the value ranges.

Aging: Aging may result in timing errors, and techniques like ThUnderVolt [58] and GreenTPU [59] can be employed for mitigating the effects of timing errors that occur in the computational units of DNN accelerators. Meanwhile, aging in the on-chip memory (6T-SRAM), one of the key component in DNN accelerators, has been addressed by techniques like the fixed aging balancing [60], adaptive aging balancing [61], and additional circuitry [62] [63]. However, these techniques are designed for specific data distribution and/or

applications, or require additional circuitry in each SRAM cell. To address this challenge, we proposed DNN-Life framework [64] that employs novel memory-write (and read) transducers to achieve an optimal duty-cycle at run time in each cell of the on-chip weight memory to mitigate NBTI aging.

Besides the HW-induced reliability threats (i.e., permanent faults, soft errors, and aging), other works have analyzed the resilience of DNNs against other threats (e.g., input noise). For instance, the FANNet methodology [65] analyzes the DNN noise tolerance using model checking techniques for formal analysis of DNNs under different ranges of input noise. The key idea is to investigate the impact of training bias on accuracy, and study the input node sensitivity under noise.

B. Secure ML: Attacks and Defenses

Security threats may come from different types of attacks, such as *side-channel attacks*, *data poisoning*, and *hardware intrusion*. These attacks can cause confidence reduction in classification accuracy, random or targeted misclassification, and IP stealing. To systematically identify the possible security attacks and defense mechanisms for Edge AI, a *threat model* (which defines the capabilities and goals of the attacker under realistic assumptions) is required [9]. The attacks can be categorized based on the Edge AI design cycle, i.e., during *training*, *HW design or implementation*, and *inference* [9] (the overview is shown in Fig. 7).

- **Training:** The attacker can manipulate the DNN model, training dataset or tools, to attack the system [66].
- **HW Implementation:** The attacker can steal the DNN IP through side-channel attacks, or hardware intrusion [66].
- **Inference:** The attacker can perform side-channel attacks for IP stealing, or manipulate the input data to achieve random or targeted misclassification [66].

Therefore, *effective defense mechanisms are required to secure Edge AI from possible attacks*. Toward this, both attacks and defenses need to be explored. In this section, we discuss different security attacks and some possible defenses (countermeasures) against these attacks.

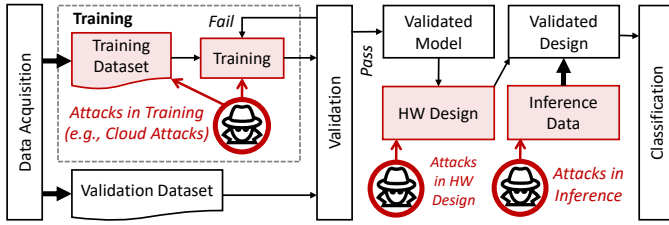


Fig. 7. An overview of security threats (attacks) and in the training, HW design/implementation, and inference phases.

Data Poisoning/Manipulation: Data poisoning aims at producing incorrect output (i.e., misclassification), and it can be performed by adding crafted noise to the DNN inputs (i.e., training or test data). Toward this, SW-level methodologies (e.g., TrISec [67], FaDec [68], and CapsAttacks [69]) have been proposed to explore the impacts of different data poisoning attacks. For instance, TrISec [67] generates imperceptible attack images as the test inputs by leveraging the backpropagation algorithm on the trained DNNs without knowledge of the training dataset. The generated attacks have close correlation and structural similarity index with the clean input, thereby making them difficult to notice in both subjective and objective tests. FaDec [68] generates imperceptible decision-based attack images as the test inputs by employing fast estimation of the classification boundary and adversarial noise optimization. It results in a fast and imperceptible attack, i.e., 16x faster than the state-of-the-art decision-based attacks. Meanwhile, CapsAttacks [69] performs analysis to study the vulnerabilities of the CapsNet by adding perturbations to the test inputs. The results show that, compared to traditional DNNs with similar width and depth, the CapsNets are more robust to affine transformations and adversarial attacks. *All these works demonstrated that DNNs are vulnerable to data poisoning attacks (which can be imperceptible), thereby the effective countermeasures are required.* Previous works have proposed several SW-level defense mechanisms. One idea is to employ encryption for protecting the training data [70]–[73]. Another idea is to employ noise filters, as the FadeML methodology [74] demonstrates that the existing adversarial attacks can be nullified using noise filters, like the Local Average with Neighborhood Pixels (LAP) and Local Average with Radius (LAR) techniques. Meanwhile, the QuSecNets methodology [75] employs quantization to eliminate the attacks in the input images. It has two quantization mechanisms, i.e., *constant quantization*, which quantizes the intensities of input pixels based on fixed quantization levels; and *trainable quantization*, which learns the quantization levels during the training phase to provide a stronger protection. This technique increases the accuracy of CNNs by 50%-96% and by 10%-50% for the perturbed images from the MNIST and the CIFAR-10, respectively.

Side Channel Attacks: These attacks aim at extracting confidential information (e.g., for data sniffing and IP stealing) without interfering with the functionality or the operation of the devices by monitoring and manipulating the side channel parameters (e.g., timing, power, temperature, etc.). The potential countermeasures are the obfuscation techniques, which target at concealing or obscuring the functional behavior or specific information [76]. For instance, the processing HW can be designed so that the power signals of the operation are independent to the processed data values, thereby concealing the secret information [77]. Meanwhile, to protect the devices from timing attacks, designers can (1) randomize the execution delay of different operations, or (2) enforce the same execution delay for all operations, thereby obscuring the underlying operation [77].

Hardware Intrusion: HW intrusion means that the attacker inserts malware or trojan (typically in the form of circuitry modification) in the processing HW for performing attacks such as confidence reduction and misclassification. The potential countermeasures are the typical HW security techniques, like the built-in self-test (BIST) to verify the functionality of the processing HW, the side channel analysis-based monitoring [78]–[80] to detect and identify anomalous side channel signals, the formal method analysis to quickly and comprehensively analyze the behavior of the processing HW (e.g., using property checker [78], mathematical model [81], SAT solver [82], and SMT solver [83]).

IV. A CROSS-LAYER FRAMEWORK FOR ENERGY-EFFICIENT AND ROBUST EDGE AI

To develop energy-efficient and robust Edge AI systems, different aspects related to performance and energy efficiency, reliability, and security should be collectively addressed. Toward this, we propose a cross-layer framework that combines different techniques from different layers of the computing stack for achieving energy-efficient and secure Edge AI systems (see the overview in Fig. 8). Our integrated framework employs the following steps.

DNN Model Creation with Secure Training: DNNs for Edge AI have to meet the design constraints (e.g., accuracy, memory, power, and energy). This can be achieved through two different ways, i.e., by employing (1) model compression through pruning [18] and quantization [22] of the pre-trained DNN model, and (2) multi-objective neural architecture search (NAS) similar to the APNAS [28] and NASCaps [29] frameworks. APNAS [28] searches for a model that has good accuracy and performance considering a systolic array-based DNN accelerator [30] through reinforcement learning. Meanwhile, NASCaps [29] optimizes the accuracy and the hardware efficiency of a given accelerator for CapsNet inference. To ensure that the generated model can be trusted, the training process should be protected from attacks. To do this, several countermeasures can be employed, e.g., by comparing the redundant trained models [84], by performing local training [85] to identify if the trained model has been attacked, or by encrypting the training dataset [70]–[73] to remove data poisoning attacks (see ❶ in Fig. 8).

Efficient Edge AI Design: Once a trusted model is generated, further performance and energy optimizations are performed (see ❷ in Fig. 8). At design time, DRAM latency and energy can be improved using techniques like ROMANet [7] and DRMap [37]. Meanwhile, the buffer latency and energy can be optimized using ROMANet [7] and DESCNet [8], and the compute latency and energy can be optimized using approximation methodologies like CANN [40], ALWANN [43], and ReD-CaNe [42]. Moreover, efficiency gains of the systems can be improved at run time using run-time power management techniques like clock gating [5], power gating [8], and DVFS [5]. Furthermore, this step should ensure that the employed techniques do not violate the design specifications, thereby providing efficient Edge AI.

Resilient Edge AI Design: To improve the resiliency of Edge AI against the reliability threats, effective mitigation techniques are required (see ❸ in Fig. 8). Toward this, the characteristics of DNN resiliency under the targeted reliability threats are evaluated. Recent works have studied the DNN resiliency in the presence of approximation errors [41] [42] and permanent faults [49]. Based on this information, appropriate fault mitigation techniques can be identified and deployed. At design time, several techniques can be employed, such as fault-aware training (e.g., FAP [47] and FAT

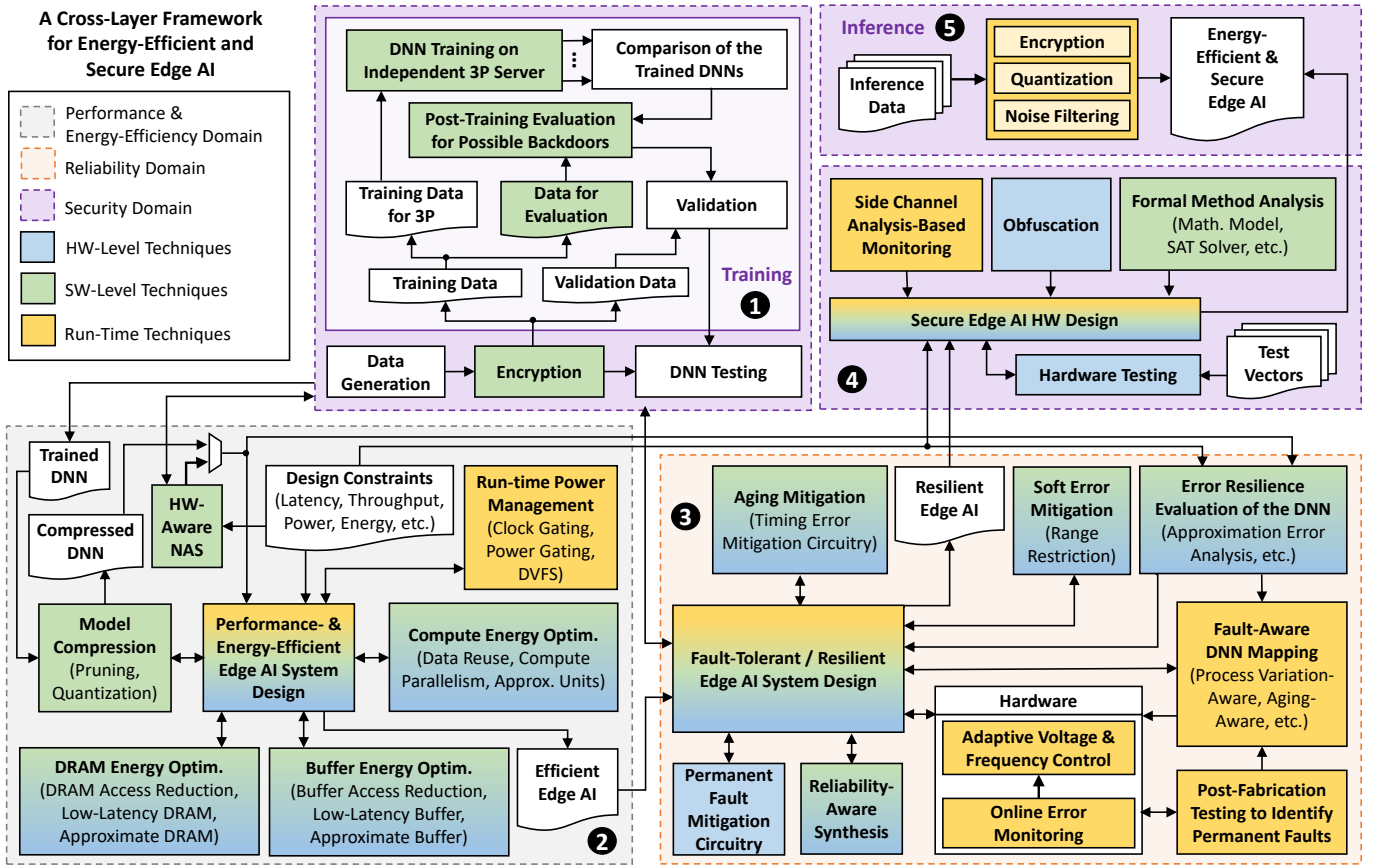


Fig. 8. Overview of our cross-layer framework for energy-efficient and secure Edge AI systems.

[48]), range restriction (e.g., FT-ClipAct [50]), and aging-aware timing error mitigation (e.g., ThUnderVolt [58] and GreenTPU [59]). Meanwhile, the fault-aware mapping (e.g., SalvageDNN [49]), the range restriction (e.g., Ranger [51]), online error monitoring and adaptive DVFS can be performed to improve the system's resiliency at run time. Furthermore, this step needs to ensure that the employed techniques do not lead to any violation of the design constraints, thereby resulting in a resilient and energy-efficient Edge AI system.

Secure HW Design/Implementation: Since the HW side also has vulnerability issues, the HW design/implementation process should be protected. Toward this, the existing HW security techniques can be employed (see ④ in Fig. 8). For instance, the side-channel analysis-based monitoring [78]–[80] can monitor the side-channel signals that attackers can exploit. Then, we can leverage the information to devise defense mechanisms that block the exploitation. Another idea is to obscure the HW information from the attacker using obfuscation techniques [86] [87]. The other techniques leverage the formal method-based analysis [78] [81]–[83] to quickly identify all possible security vulnerabilities and the corresponding defense mechanisms. To evaluate the efficacy of the applied defense mechanisms, HW testing is performed. Furthermore, this step also needs to ensure that the employed defense techniques still meet the design constraints, thereby resulting in a secure HW design.

Secure Inference: Since the security attacks can also target the inference phase, a secure inference is required (see ⑤ in Fig. 8). Most of the attacks come in the form of data manipulation. Hence, we can perform data encryption to block the insertion of perturbations into the input data. Another idea is to mitigate the input data-based attacks by employing quantization-based defenses such as, QuSecNets [75]

and by noise filters like in the FadeML methodology [74].

Note that all the proposed steps can jointly provide an end-to-end cross-layer framework that performs HW- and SW-level optimizations at the design-time and run-time. *Our proposed framework ensures that the Edge AI systems have high performance and energy efficiency, while providing correct output under diverse reliability and security threats.*

V. NEUROMORPHIC RESEARCH CONSIDERING SNNs

SNNs are considered as the third generation of NN models, which employ spike-encoded information and computation [88]. Due to their bio-inspired operations, SNNs have a high potential to provide energy-efficient computation. Recent works have been actively exploring two research directions, i.e., SNNs with a localized learning rule like the spike-timing-dependent plasticity (STDP) [3], and SNNs obtained from DNN conversions [89].

A. Improving the Energy Efficiency of SNNs

To improve the energy efficiency of SNNs, several HW- and SW-level optimizations have been proposed. For HW-level techniques, SNN accelerators have been designed, such as TrueNorth [90], SpiNNaker [91], PEASE [92], Loihi [93], and ODIN [94]. Recent work (i.e., the SparkXD framework [95]) optimizes the DRAM access latency and energy for SNN inference by employing the reduced-voltage DRAM operations and effective DRAM mapping, leading to DRAM energy saving by up to 40% (see Fig. 9). For SW-level techniques, the FSpINN framework [3] improves the energy efficiency of SNN processing in the training (avg. 3.5x) and the inference (avg. 1.8x) through the optimization of neural operations

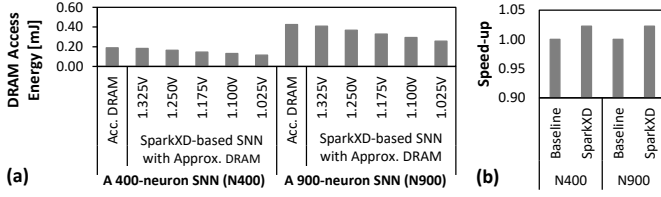


Fig. 9. (a) The DRAM energy in an SNN inference on MNIST incurred by an SNN with accurate DRAM (the Baseline) and a SparkXD-based SNN with approximate DRAM. (b) The speed-up achieved by the SparkXD.

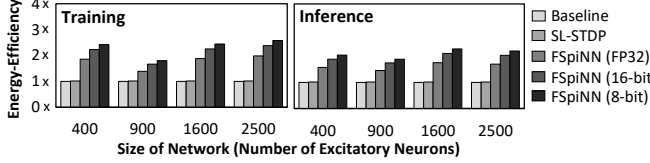


Fig. 10. The FSpINN improves the energy efficiency compared to the standard unsupervised SNN (Baseline) [98] and the SL-STDP [99] across different network sizes for both training and inference phases on the MNIST workload.

and quantization, without accuracy loss (see Fig. 10). The Q-SpiNN [96] explores different precision levels, rounding schemes, and quantization schemes (i.e., post- and in-training quantization) to maximize memory savings for both weights and neuron parameters (which occupy considerable amount of memory in the accelerator fabric). The other techniques target at mapping and running the SNN applications (e.g., DVS Gesture Recognition [89] and Autonomous Cars [97]) on neuromorphic hardware (i.e., Loihi) to improve the energy efficiency of their processing compared to running them on conventional platforms (e.g., CPUs, GPUs). As shown in Fig. 11, the CarSNN [97] improves by 2% the N-CARS accuracy, compared to the related works, while consuming only 315 mW on the Loihi Neuromorphic Chip, thus making a step forward towards ultra-low power event-based vision for autonomous cars.

B. Improving the Reliability of SNNs

In recent years, the SNN reliability aspect starts gaining attention because it is crucial to ensure the functionality of SNN systems. Moreover, the reliability issues may come from various sources (e.g., manufacturing defects, optimization techniques, etc.). For instance, employing the reduced-voltage DRAM in SNN accelerators can offer energy savings, but at the cost of increased DRAM errors which may alter the weight values and reduce the accuracy. Toward this, the SparkXD framework [95] improves the SNN reliability (preserving the high accuracy) by incorporating the information of faults (i.e., fault map and fault rate) in the retraining process, i.e., so-called the fault-aware training (FAT). Furthermore, the ReSpawn framework [103] mitigates the negative impact of permanent and approximation-induced faults in the off-chip and on-chip memories of SNN HW accelerators through a cost-effective fault-aware mapping (FAM). It places the weight bits with higher significance on the non-faulty memory cells, which enhances the reliability of SNNs without retraining, and achieves up to 70% accuracy improvement from the baseline, as shown in Fig. 12. In this manner, the ReSpawn can also improve the yield and reduce the per-unit-cost of SNN chips. Besides the HW-induced faults, the SNN systems may encounter dynamically changing environments, which cause the offline-learned knowledge to obsolete at run-time. Toward this, the SpikeDyn framework [104] employs an unsupervised continual learning mechanism by leveraging the internal characteristics of neural dynamics and weight decay function to enable an online learning scenario.

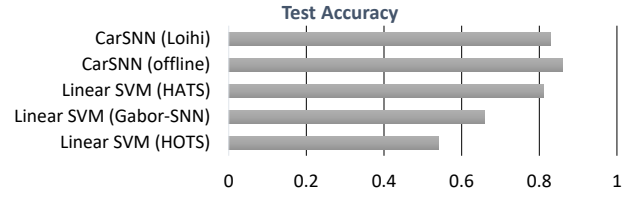


Fig. 11. The CarSNN [97], although being more energy-efficient, achieves higher accuracy for the N-CARS dataset than the related works like the HATS [100], Gabor-SNN [101], and HOTS [102] techniques.

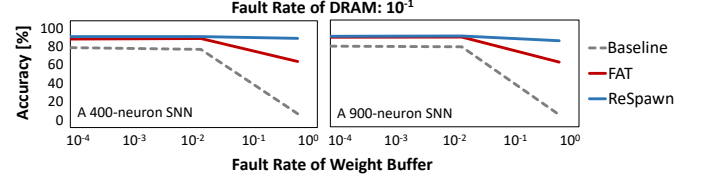


Fig. 12. The ReSpawn maintains higher accuracy than the fault-aware training (FAT), across different network sizes and different fault rates in memories.

C. Improving the Security of SNNs

Previous works have studied that SNNs are vulnerable to security attacks, like data poisoning attacks on traditional image classification datasets like the MNIST [105] and on event-based datasets [106], showing different behavior under attack, compared to the non-spiking DNNs. Furthermore, SNNs are also vulnerable to externally triggered bit-flip attacks. The experiments conducted in [107] show that only 4 bit-flips at the most sensitive weight memory cells are sufficient for fooling SNNs on the CIFAR10 dataset. Once these memory locations are found, the attacker can trigger the malicious hardware that generates bit-flips by inserting a specific pattern in the input images. To address the security problem, several defense techniques have been proposed. One technique is exploiting the structural network parameters, e.g., threshold voltage and time window, to improve the SNN robustness [108]. By fine-tuning such parameters, the SNNs can be up to 85% more robust than non-spiking DNNs. Meanwhile, the R-SNN methodology [109] employs noise filtering to remove the adversarial attacks in the DVS inputs. The experiments demonstrate that such noise filtering slightly affects the SNN outputs for clean event sequences, while a wide range of filter parameters can increase the robustness of the SNN under attack by up to 90%.

VI. CONCLUSION

The use of Edge AI and tinyML systems is expected to grow fast in the coming years. Therefore, ensuring their high energy efficiency and robustness is important. This paper provides an overview of challenges and potential solutions for improving performance, energy efficiency, and robustness (i.e., reliability and security) of Edge AI. It shows that HW/SW co-design and co-optimization techniques at the design- and run-time can be combined through a cross-layer framework to efficiently address these challenges.

ACKNOWLEDGMENTS

This work was partly supported by Intel Corporation through Gift funding for the project "Cost-Effective Dependability for Deep Neural Networks and Spiking Neural Networks".

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [2] M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, vol. 8, pp. 225 134–225 180, 2020.

- [3] R. V. W. Putra and M. Shafique, "Fspinn: An optimization framework for memory-efficient and energy-efficient spiking neural networks," *IEEE TCAD*, vol. 39, no. 11, pp. 3601–3613, 2020.
- [4] A. Marchisio, M. A. Hanif, F. Khalid, G. Plastiras, C. Kyrkou, T. Theocharides, and M. Shafique, "Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges," in *Proc. of ISVLSI*, 2019, pp. 553–559.
- [5] F. Kriebel, S. Rehman, M. A. Hanif, F. Khalid, and M. Shafique, "Robustness for smart cyber physical systems and internet-of-things: From adaptive robustness methods to reliability and security for machine learning," in *Proc. of ISVLSI*, 2018, pp. 581–586.
- [6] M. Shafique, M. Naseer, T. Theocharides, C. Kyrkou, O. Mutlu, L. Orosa, and J. Choi, "Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead," *IEEE Design and Test*, vol. 37, no. 2, pp. 30–57, 2020.
- [7] R. V. W. Putra, M. A. Hanif, and M. Shafique, "Romanet: Fine-grained reuse-driven off-chip memory access management and data organization for deep neural network accelerators," *IEEE TVLSI*, vol. 29, no. 4, pp. 702–715, 2021.
- [8] A. Marchisio, V. Mrazek, M. A. Hanif, and M. Shafique, "Descnet: Developing efficient scratchpad memories for capsule network hardware," *IEEE TCAD*, pp. 1–1, 2020.
- [9] M. A. Hanif, F. Khalid, R. V. W. Putra, S. Rehman, and M. Shafique, "Robust machine learning systems: Reliability and security for deep neural networks," in *Proc. of IOLTS*, 2018, pp. 257–260.
- [10] B. Raghunathan, Y. Turakhia, S. Garg, and D. Marculescu, "Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors," in *Proc. of DATE*, 2013, pp. 39–44.
- [11] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE TDMR*, vol. 5, no. 3, 2005.
- [12] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The eda challenges in the dark silicon era: Temperature, reliability, and variability perspectives," in *Proc. of DAC*, 2014, pp. 1–6.
- [13] M. Shafique, T. Theocharides, C.-S. Bouganis, M. A. Hanif, F. Khalid, R. Hafiz, and S. Rehman, "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the iot era," in *Proc. of DATE*, 2018, pp. 827–832.
- [14] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. of NIPS*, 2017, p. 3859–3869.
- [15] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM JETC*, vol. 13, no. 3, Feb. 2017.
- [16] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proc. of ECCV*, 2018, pp. 784–800.
- [17] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.
- [18] A. Marchisio, M. A. Hanif, M. Martina, and M. Shafique, "Prunet: Class-blind pruning method for deep neural networks," in *Proc. of IJCNN*, 2018, pp. 1–8.
- [19] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.
- [20] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. of ICML*, 2015.
- [21] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of CVPR*, 2018.
- [22] A. Marchisio, B. Bussolino, A. Colucci, M. Martina, G. Masera, and M. Shafique, "Q-capsnets: A specialized framework for quantizing capsule networks," in *Proc. of DAC*, 2020, pp. 1–6.
- [23] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
- [24] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.
- [25] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*, 2018, pp. 4095–4104.
- [26] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proc. of ECCV*, 2018, pp. 19–34.
- [27] M. Tan *et al.*, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proc. of CVPR*, 2019, pp. 2820–2828.
- [28] P. Acharariti, M. A. Hanif, R. V. W. Putra, M. Shafique, and Y. Hara-Azumi, "Apnas: Accuracy-and-performance-aware neural architecture search for neural hardware accelerators," *IEEE Access*, vol. 8, pp. 165 319–165 334, 2020.
- [29] A. Marchisio, A. Massa, V. Mrazek, B. Bussolino, M. Martina, and M. Shafique, "Nascaps: A framework for neural architecture search to optimize the accuracy and hardware efficiency of convolutional capsule networks," in *Proc. of ICCAD*, 2020, pp. 1–9.
- [30] M. A. Hanif, R. V. W. Putra, M. Tanvir, R. Hafiz, S. Rehman, and M. Shafique, "Mpna: A massively-parallel neural array accelerator with dataflow optimization for convolutional neural networks," *arXiv preprint arXiv:1810.12910*, 2018.
- [31] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. of ISCA*, 2017, pp. 1–12.
- [32] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.
- [33] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proc. of FPGAs*, 2015.
- [34] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong, "Caffeine: Toward uniformed representation and acceleration for deep convolutional neural networks," *IEEE TCAD*, vol. 38, 2018.
- [35] J. Li *et al.*, "Smartshuttle: Optimizing off-chip memory accesses for deep learning accelerators," in *Proc. of DATE*, 2018, pp. 343–348.
- [36] S. Tewari, A. Kumar, and K. Paul, "Bus width aware off-chip memory access minimization for cnn accelerators," in *Proc. of ISVLSI*, 2020.
- [37] R. V. W. Putra, M. A. Hanif, and M. Shafique, "Drmap: A generic dram data mapping policy for energy-efficient processing of convolutional neural networks," in *Proc. of DAC*, 2020, pp. 1–6.
- [38] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (salp) in dram," in *Proc. of ISCA*, 2012.
- [39] D. Kang, D. Kang, and S. Ha, "Multi-bank on-chip memory management techniques for cnn accelerators," *IEEE TC*, 2021.
- [40] M. A. Hanif, F. Khalid, and M. Shafique, "Cann: Curable approximations for high-performance deep neural network accelerators," in *Proc. of DAC*, 2019, pp. 1–6.
- [41] M. A. Hanif, R. Hafiz, and M. Shafique, "Error resilience analysis for systematically employing approximate computing in convolutional neural networks," in *Proc. of DATE*, 2018, pp. 913–916.
- [42] A. Marchisio, V. Mrazek, M. A. Hanif, and M. Shafique, "Red-cane: A systematic methodology for resilience analysis and design of capsule networks under approximations," in *Proc. of DATE*, 2020.
- [43] V. Mrazek, Z. Vasicek, L. Sekanina, M. A. Hanif, and M. Shafique, "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," in *Proc. of ICCAD*, 2019, pp. 1–8.
- [44] R. Vadlamani, J. Zhao, W. Burleson, and R. Tessier, "Multicore soft error rate stabilization using adaptive dual modular redundancy," in *Proc. of DATE*, 2010, pp. 27–32.
- [45] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Res. Dev.*, vol. 6, no. 2, 1962.
- [46] H. Y. Sze, "Circuit and method for rapid checking of error correction codes using cyclic redundancy check," Jul. 18 2000, uS Patent 6,092,231.
- [47] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *Proc. of VTS*, 2018, pp. 1–6.
- [48] S. Kim *et al.*, "Energy-efficient neural network acceleration in the presence of bit-level memory errors," *IEEE TCAS*, vol. 65, no. 12, pp. 4285–4298, 2018.
- [49] M. A. Hanif and M. Shafique, "Salvagednn: salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping," *RSTA*, vol. 378, no. 2164, 2020.
- [50] 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. of DATE*, 2020, pp. 1241–1246.
- [51] Z. Chen, G. Li, and K. Pattabiraman, "A low-cost fault corrector for deep neural networks through range restriction," in *Proc. of DSN*, 2021.
- [52] G. Li *et al.*, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proc. of SC*, 2017.
- [53] C. Schorn, A. Guntoro, and G. Ascheid, "Efficient on-line error detection and mitigation for deep neural network accelerators," in *Proc. of SAFECOMP*, 2018, pp. 205–219.
- [54] E. Ozen and A. Orailoglu, "Sanity-check: Boosting the reliability of safety-critical deep neural network applications," in *Proc. of ATS*, 2019.

- [55] S. Hong *et al.*, "Terminal brain damage: Exposing the graceful degradation in deep neural networks under hardware fault attacks," in *Proc. of USENIX*, 2019.
- [56] A. Mahmoud *et al.*, "Hardnn: Feature map vulnerability evaluation in cnns," *arXiv preprint arXiv:2002.09786*, 2020.
- [57] K. Zhao *et al.*, "Ft-cnn: Algorithm-based fault tolerance for convolutional neural networks," *IEEE TPDS*, vol. 32, no. 7, 2021.
- [58] J. Zhang *et al.*, "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proc. of DAC*, 2018, pp. 1–6.
- [59] P. Pandey *et al.*, "Greentpu: Improving timing error resilience of a near-threshold tensor processing unit," in *Proc. of DAC*, 2019, pp. 1–6.
- [60] S. V. Kumar, K. Kim, and S. S. Sapatnekar, "Impact of nbt on sram read stability and design for reliability," in *Proc. of ISQED*, 2006.
- [61] M. Shafique, M. U. K. Khan, O. Tüfek, and J. Henkel, "Enaam: Energy-efficient anti-aging for on-chip video memories," in *Proc. of DAC*, 2015.
- [62] T. Siddiqua and S. Gurumurthi, "Enhancing nbt recovery in sram arrays through recovery boosting," *IEEE TVLSI*, vol. 20, no. 4, 2011.
- [63] B. Zatt *et al.*, "A low-power memory architecture with application-aware power management for motion & disparity estimation in multiview video coding," in *Proc. of ICCAD*, 2011, pp. 40–47.
- [64] M. A. Hanif and M. Shafique, "Dnn-life: An energy-efficient aging mitigation framework for improving the lifetime of on-chip weight memories in deep neural network hardware architectures," in *Proc. of DATE*, 2021, pp. 729–734.
- [65] M. Naseer, M. F. Minhas, F. Khalid, M. A. Hanif, O. Hasan, and M. Shafique, "Fannet: Formal analysis of noise tolerance, training bias and input sensitivity in neural networks," in *Proc. of DATE*, 2020.
- [66] N. Papernot *et al.*, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.
- [67] F. Khalid, M. A. Hanif, S. Rehman, R. Ahmed, and M. Shafique, "Trisec: Training data-unaware imperceptible security attacks on deep neural networks," in *Proc. of IOLTS*, 2019, pp. 188–193.
- [68] F. Khalid, H. Ali, M. Abdullah Hanif, S. Rehman, R. Ahmed, and M. Shafique, "Fadec: A fast decision-based attack for adversarial machine learning," in *Proc. of IJCNN*, 2020, pp. 1–8.
- [69] A. Marchisio, G. Nanfa, F. Khalid, M. A. Hanif, M. Martina, and M. Shafique, "Capsattacks: Robust and imperceptible adversarial attacks on capsule networks," *CoRR*, vol. abs/1901.09878, 2019.
- [70] E. Hesamifard, H. Takabi, and M. Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv preprint arXiv:1711.05189*, 2017.
- [71] F.-J. González-Serrano, A. Amor-Martín, and J. Casamayón-Antón, "Supervised machine learning using encrypted training data," *Int. J. of Information Security*, vol. 17, no. 4, pp. 365–377, 2018.
- [72] J. Gao, W. Wang, M. Zhang, G. Chen, H. Jagadish, G. Li, T. K. Ng, B. C. Ooi, S. Wang, and J. Zhou, "Panda: facilitating usable ai development," *arXiv preprint arXiv:1804.09997*, 2018.
- [73] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, 2018.
- [74] F. Khalid, M. A. Hanif, S. Rehman, J. Qadir, and M. Shafique, "Fademi: Understanding the impact of pre-processing noise filtering on adversarial machine learning," in *Proc. of DATE*, 2019, pp. 902–907.
- [75] F. Khalid, H. Ali, H. Tariq, M. A. Hanif, S. Rehman, R. Ahmed, and M. Shafique, "Qusecnets: Quantization-based defense mechanism for securing deep neural network against adversarial attacks," in *Proc. of IOLTS*, 2019, pp. 182–187.
- [76] T. Hoque, R. S. Chakraborty, and S. Bhunia, "Hardware obfuscation and logic locking: A tutorial introduction," *IEEE Design and Test*, vol. 37, no. 3, pp. 59–77, 2020.
- [77] S. Bhunia and M. Tehranipoor, "Chapter 8 - side-channel attacks," in *Hardware Security*, S. Bhunia and M. Tehranipoor, Eds. Morgan Kaufmann, 2019, pp. 193–218.
- [78] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Proc. of CAV*, 2017, pp. 3–29.
- [79] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel, "Chiron: Privacy-preserving machine learning as a service," *arXiv preprint arXiv:1803.05961*, 2018.
- [80] L. Wei *et al.*, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proc. of ACSAC*, 2018.
- [81] D. Gopinath *et al.*, "Deepsafe: A data-driven approach for assessing robustness of neural networks," in *Proc. of ATVA*, 2018, pp. 3–19.
- [82] L. Kuper *et al.*, "Toward scalable verification for safety-critical deep networks," *arXiv preprint arXiv:1801.05950*, 2018.
- [83] G. Katz and pthers, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Proc. of CAV*, 2017, pp. 97–117.
- [84] T. Li *et al.*, "Outsourced privacy-preserving classification service over encrypted data," *J. of Network and Computer Appl.*, vol. 106, 2018.
- [85] Y. Liu, J. Chen, and H. Chen, "Less is more: Culling the training set to improve robustness of deep neural networks," in *Proc. of GameSec*, 2018, pp. 102–114.
- [86] I. Rosenberg, G. Sicard, and E. O. David, "Deepapt: nation-state apt attribution using end-to-end deep neural networks," in *Proc. of ICANN*, 2017, pp. 91–99.
- [87] Q. Sun *et al.*, "Natural and effective obfuscation by head inpainting," in *Proc. of CVPR*, 2018, pp. 5050–5059.
- [88] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [89] R. Massa, A. Marchisio, M. Martina, and M. Shafique, "An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor," in *Proc. of IJCNN*, 2020, pp. 1–9.
- [90] F. Akopyan *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE TCAD*, vol. 34, no. 10, pp. 1537–1557, 2015.
- [91] E. Painkras *et al.*, "Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation," *IEEE JSSC*, vol. 48, no. 8, pp. 1943–1953, 2013.
- [92] A. Roy, S. Venkataramani, N. Gala, S. Sen, K. Veezhinathan, and A. Raghunathan, "A programmable event-driven architecture for evaluating spiking neural networks," in *Proc. of ISLPED*, 2017.
- [93] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [94] C. Frenkel *et al.*, "A 0.086-mm² 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos," *IEEE TBCAS*, vol. 13, no. 1, pp. 145–158, 2018.
- [95] R. V. W. Putra, M. A. Hanif, and M. Shafique, "Sparkxd: A framework for resilient and energy-efficient spiking neural network inference using approximate dram," in *Proc. of DAC*, 2021, pp. 1–6.
- [96] R. V. W. Putra and M. Shafique, "Q-spinn: A framework for quantizing spiking neural networks," in *Proc. of IJCNN*, 2021, pp. 1–8.
- [97] A. Viale, A. Marchisio, M. Martina, G. Masera, and M. Shafique, "Carsnn: An efficient spiking neural network for event-based autonomous cars on the loihi neuromorphic research processor," in *Proc. of IJCNN*, 2021.
- [98] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *FNCOM*, vol. 9, p. 99, 2015.
- [99] G. Srinivasan *et al.*, "Spike timing dependent plasticity based enhanced self-learning for efficient pattern recognition in spiking neural networks," in *Proc. of IJCNN*, 2017.
- [100] A. Sironi *et al.*, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *Proc. of CVPR*, 2018.
- [101] A. Bovik, M. Clark, and W. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE TPAMI*, vol. 12, no. 1, 1990.
- [102] X. Lagore *et al.*, "Hots: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE TPAMI*, vol. 39, no. 7, pp. 1346–1359, 2017.
- [103] R. V. W. Putra, M. A. Hanif, and M. Shafique, "Respawn: Energy-efficient fault-tolerance for spiking neural networks considering unreliable memories," in *Proc. of ICCAD*, 2021, pp. 1–8.
- [104] R. V. W. Putra and M. Shafique, "Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments," in *Proc. of DAC*, 2021.
- [105] A. Marchisio, G. Nanfa, F. Khalid, M. A. Hanif, M. Martina, and M. Shafique, "Is spiking secure? a comparative study on the security vulnerabilities of spiking and deep neural networks," in *Proc. of IJCNN*, 2020, pp. 1–8.
- [106] A. Marchisio, G. Pira, M. Martina, G. Masera, and M. Shafique, "Dvs-attacks: Adversarial attacks on dynamic vision sensors for spiking neural networks," in *Proc. of IJCNN*, 2021.
- [107] V. Venceslai *et al.*, "Neuroattack: Undermining spiking neural networks security through externally triggered bit-flips," in *Proc. of IJCNN*, 2020.
- [108] R. El-Allami *et al.*, "Securing deep spiking neural networks against adversarial attacks through inherent structural parameters," in *Proc. of DATE*, 2021, pp. 774–779.
- [109] A. Marchisio *et al.*, "R-snn: An analysis and design methodology for robustifying spiking neural networks against adversarial attacks through noise filters for dynamic vision sensors," in *Proc. of IROS*, 2021.