26th IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS)

Dependable Deep Learning: Towards Cost-Efficient Resilience of Deep Neural Network Accelerators against Soft Errors and Permanent Faults

Muhammad Abdullah Hanif, Muhammad Shafique
Technische Universität Wien (TU Wien), Vienna, Austria
{muhammad.hanif, muhammad.shafique}@tuwien.ac.at

***Abstract*—Deep Learning has enabled machines to learn computational models (i.e., Deep Neural Networks – DNNs) that can perform certain complex tasks with claims to be close to human-level precision. This state-of-the-art performance offered by DNNs in many Artificial Intelligence (AI) applications has paved their way to being used in several safety-critical applications where even a single failure can lead to catastrophic results. Therefore, improving the robustness of these models to hardware-induced faults (such as soft errors, aging, and manufacturing defects) is of significant importance to avoid any disastrous event. Traditional redundancy-based fault mitigation techniques cannot be employed in a wide of applications due to their high overheads, which, when coupled with the compute-intensive nature of DNNs, lead to undesirable resource consumption. In this article, we present an overview of different low-cost fault-mitigation techniques that exploit the intrinsic characteristics of DNNs to limit their overheads. We discuss how each technique can contribute to the overall resilience of a DNN-based system, and how they can be integrated together to offer resilience against multiple diverse hardware-induced reliability threats. Towards the end, we highlight several key future directions that are envisioned to help in achieving highly dependable DL-based systems.**

***Index Terms*—Reliability, Dependability, Cost, Efficiency, Deep Learning, DL, Deep Neural Networks, DNNs, Faults, Soft Errors, Aging, Permanent Faults, Manufacturing Defects, Yield, Resilience, Robustness, Systems, Architecture, Accelerator.**

I. Introduction

Deep Learning (DL) enables computational models that are composed of multiple layers to learn hierarchical features directly from raw data [1]. These models (Deep Neural Networks - DNNs) offer state-of-the-art accuracy in many Artificial Intelligence (AI) applications, such as image classification, image segmentation, object detection and speech recognition [2] [3] [4], which in some scenarios have been claimed to surpass human-level precision [5]. This has laid the foundation for these models being used in several safety-critical applications like autonomous driving [6] [7] [8] and medical diagnosis [9] [10] [11]. In these applications, the system is required to analyze the input generated by sensors in real-time. Driven by the compute-intensive nature of DNNs, a significant amount of research has been carried out in developing DNN hardware accelerators that can offer performance- and energy-efficient inference for DNN-based applications [12]. A few examples are TPU [13], MAERI [14] and Eyeriss [15].

Alongside resource-efficient execution, these systems are expected to generate highly reliable output. However, besides the input- and environment-induced errors, the hardware-induced faults (such as soft errors, device aging and manufacturing defects) lead to an increased number of false positives and false negatives, and even system failures (see Fig. 1). This leads to a significant degradation in the performance of the system and also raises various safety concerns in critical applications. For example, in the context of autonomous vehicles, detecting obstacles in the environment when nothing is actually there (false positives) can lead to abrupt braking, and reduced speed while missing the detection of an obstacle (false negative) can lead to a fatal accident. Moreover, these systems have to meet standards like ISO 26262 [16] that allow for only a limited number of errors in the entire system's lifetime. Therefore, it is vital to ensure the robustness of DNNs against a wide range of dependability threats.

Traditional fault-mitigation techniques that are based on redundancy (e.g., Dual Modular Redundancy (DMR) [17] and Triple Modular Redundancy (TMR) [18]) are not effective to be employed for DNN-based applications due to their huge overheads arising from redundant hardware/execution of compute-intensive DNNs and synchronization issues. Other techniques such as Instruction Duplication (ID) [19] [20] and use of Error-Correcting Codes (ECC) also pose similar issues

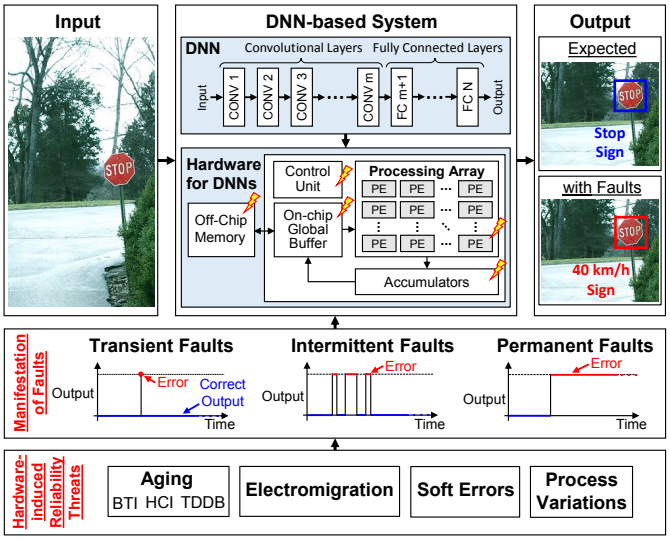

Fig. 1. Reliability threats, their manifestation, and impact on a DNN-based system's output. (used "Stop" sign picture is from the COCO dataset [21])

that lead to noticeable degradation in the system's performance and power/energy efficiency. Therefore, *alternate techniques are being developed that exploit the intrinsic characteristics of DNNs to boost their error-resilience at low cost*. These techniques are being developed with the objective that they can be integrated in the current state-of-the-art systems without drastically reducing their benefits.

Contributions of This Paper:

1) We provide an overview of different methodologies for enhancing the resilience of DNN-based systems against different reliability vulnerabilities, and also highlight their key characteristics.
2) We discuss how each technique can contribute to the overall resilience of a DNN-based system, and how can they be integrated in a cross-layer framework to offer protection against a wide spectrum of hardware-induced reliability threats while incurring low design-time and run-time overheads.
3) We highlight the key challenges and open research problems on the road ahead towards developing dependable AI systems that offer an optimal level of protection against reliability threats.

II. Hardware-Induced Reliability Threats

Continuous scaling of CMOS technology, mainly for improved performance and efficiency, has raised various reliability concerns for nano-scale devices. Fig. 1 shows the main types of threats and how they manifest to affect the functionality of systems.

- **Soft Errors** are transient faults that manifest as bit flips and result in data corruption. These are mainly caused by extrinsic sources like alpha particles emitted from the impurities in packaging materials or by neutrons from cosmic radiations when they strike the chip [33]. Temperature is another factor that can result in an increase in the Soft Error Rate (SER).
- **Aging** of electronic circuits occurs due to various physical phenomena such as Bias Temperature Instability (BTI), Hot Carrier

978-1-7281-8187-5/20/$31.00 ©2020 IEEE

TABLE I
AN OVERVIEW OF DIFFERENT HARDWARE (HW) AND SOFTWARE (SW) FAULT-MITIGATION TECHNIQUES FOR MACHINE LEARNING SYSTEMS.

Technique	Abstraction Layer	Related Work	Brief Description	Cost	Targeted Faults	Other Dependencies
Fault-Aware Training	SW	[22] [23]	It incorporates the information of hardware-level faults in the training process to enable the network to adapt accordingly and produce correct output using faulty hardware during inference.	Design-time: High	Permanent faults in memory	Fault map extraction
Fault-Aware Pruning	HW Architecture + SW	[24] [23] [25]	This technique leverages the inherent redundancy in DNNs and propose to drop the faulty computations to maintain the accuracy of the provided DNN. In case of higher number of faults, this technique is coupled with fault-aware training to offer optimal performance.	Run-time: Low	Permanent faults in computational array	Modifications in HW architecture + fault map extraction
Fault-Aware Mapping	HW Architecture + SW	[24]	Fault-aware mapping leverages the fact that different computations/parameters have different level of significance and pruning the least significant set of computations/parameters during fault-aware pruning can have the least impact on the accuracy of the DNN-based system. Therefore, it proposes to find the least significant set and, with the help of mapping, map them to the faulty components	Design-time: Low	Permanent faults in computational array	Fault map extraction
Range Restriction	SW	[26] [27]	In DNNs, faults in sensitive computations usually result in abnormal output at an intermediate stage in the network that can propagate to the output to produce incorrect result. Range restriction techniques propose to determine ranges of intermediate outputs and identify any output that does not fall in the defined range as faulty and map them to some in the range using some pre-defined mechanism.	Design-time: Low Run-time: Low	Transient Faults (i.e., Soft Errors)	Modifications in HW architecture in case of specialized HW
Algorithm-based Fault Tolerance	SW	[28]	Algorithm-based Fault Tolerance (ABFT) techniques that are based on checksum computation are commonly used to protect matrix-matrix multiplications. These techniques have been extended to offer cost-effective error detection and correction for convolution operations as well.	Run-time: Low	Transient faults in computational units and data corruption	No
Radiation Hardening	Circuit	[29] [30]	This approach enables us to make the hardware less prone to radiation-induced faults, using stronger cells or through designing the hardware components such that they are biased towards a specific type of faults towards which DNNs are more resilient. For example, designing SRAM cells such that, in case of radiation-induced faults, they result in a '0' more often then a '1'.	Run-time: Low	Transient Faults (i.e., Soft Errors)	No
Redundancy	HW Architecture + SW	[31] [32]	It introduces selective spatial and temporal redundancy to detect and rectify errors in critical computations during inference.	Run-time: Moderate to High	Transient, intermittent and permanent faults	No

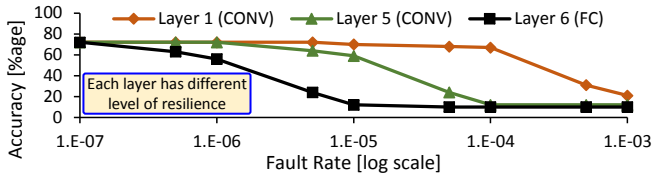

Fig. 2. Resilience analysis of different layers of the AlexNet trained for the CIFAR-10 dataset [27].

Injection (HCI), Time-Dependent Dielectric Breakdown (TDDB) and Electromigration (EM). It typically results in circuits becoming slower with time, e.g., by increasing the threshold voltage (V_{TH}) of the circuits [34], or breakdown of dielectric and wires. The aging faults manifest as timing errors during the early stages, and later can transform into permanent faults. Alongside various other factors, the rate of aging usually increases with temperature.

- **Process Variations** is an issue caused by the variations introduced during the manufacturing process. This issue arises due to the fact that it is difficult to manufacture transistors with the same properties, such as channel length, oxide thickness and doping levels, at the nano-scale. This results in inter-chip and intra-chip variations, which affect the performance (e.g., in the form of reduced operating frequency) and power efficiency, as well as the yield of the manufacturing process (in case of permanent faults).

III. COST-EFFECTIVE RESILIENCE AGAINST HARDWARE-INDUCED RELIABILITY THREATS

Table I summarizes several prominent techniques designed to improve the resilience of DNN-based systems against hardware-induced faults. The techniques are developed based on the inherent characteristics of the DNNs to limit the overhead cost. In the following subsections, we discuss the techniques used for mitigating different types of hardware-induced faults, and how they can be integrated together to offer protection against multiple types of reliability threats.

A. Soft Error Mitigation

Soft errors manifest as random bit-flips in a system that can propagate to the application layer and may lead to incorrect output or system failure. Although DNNs are considered highly resilient, these faults, when occur at critical locations, can result in undesirable results and may have severe consequences, specifically in applications like autonomous driving, banking, healthcare, and aviation. Different parts of a DNN show a different level of resilience to faults (see Fig. 2). Frameworks like BinFI [35] and Ares [36] have been proposed to compute the resilience of each part of a given DNN and identify the critical bits/computations. This analysis helps in introducing selective protection and limit the overhead cost of redundancy.

Various techniques have been proposed to mitigate the effects of soft errors. Radiation hardening of particular nodes [37] (identified using resilience analysis) can significantly boost the fault tolerance of a system at a low cost. Based on the fact that a bit-flip from 1-to-0 is relatively less critical [38], modifications have been proposed for SRAM cells that force the cells to 0s in case of radiation-induced errors [30]. However, these techniques require modifications in all/most of the SRAM cells and, therefore, can result in a high-to-moderate level of overheads. Recently, range restriction-based techniques have emerged, which locally detect the faults that can influence the output and restrict them from propagating into the network [26] [27]. The flow of these techniques is illustrated in Fig. 3. Note that, as these techniques do not require additional training or hardware, they incur low overheads. Fig. 4 presents our analysis that shows the effectiveness of these techniques when applied to the AlexNet trained for the CIFAR-10 dataset.

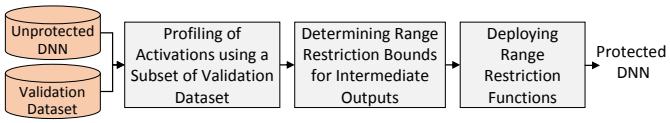

Fig. 3. Overview of range restriction-based fault-mitigation techniques

B. Permanent Fault Mitigation

Manufacturing-induced defects / permanent faults are static in nature and can be detected through post-fabrication testing. The extracted fault maps/statistics can then be used to improve the resilience of the DNNs, thereby enabling reliable execution on a faulty hardware. One of the promising techniques towards this is fault-aware training [22] [23], where the training process takes into consideration the information of faults and train the network accordingly. The flow of fault-aware training is shown in Fig. 5. Although fault-aware training offers a significant improvement in the resilience of a DNN, it cannot be used in all possible circumstances. For example, if the complete training

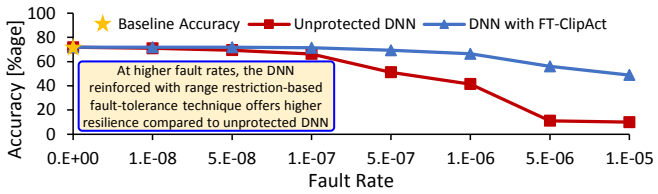

Fig. 4. Resilience of the AlexNet (trained for the CIFAR-10 dataset) with and without range restriction-based fault mitigation, based on our analysis in [27].

dataset is not available, even after training, the network will provide sub-optimal results. Moreover, in the case of a large number of faulty chips, where each chip has a different fault map, the training overhead can be significant. Techniques like fault-aware pruning [23] can be used without retraining in such scenarios. This technique is mainly designed to mitigate permanent faults in the computational units of a systolic array-based DNN hardware. Fig. 8a shows the hardware architecture of a systolic array [13], and Fig. 8b shows the design-time architectural modifications required in the processing elements to enable fault-aware pruning at run-time. These techniques offer limited improvement in the resilience of a system. Fault-aware mapping [24] has recently been proposed to overcome these limitations and to enable the system to maintain its baseline accuracy by adapting the mapping of the computations based on hardware faults. Fig. 6 shows an overview of the flow of our fault-aware mapping technique [24]. Its working principle is illustrated in Fig. 7. Fig. 7b shows the default mapping of four 2x2 filters (shown in Fig. 7a) onto an un-faulty 4x4 computational array. In case there are faulty MAC units in the array, the default mapping strategy might map critical weights (i.e., weights having higher magnitude) on those units (see Fig. 7c), which are then pruned due to fault-aware pruning. Fig. 7d shows that, by reshuffling the filters, the relatively ineffectual weights can be mapped on the fault MAC units to reduce the accuracy drop in case of fault-aware pruning. Fig. 9 shows the effectiveness of fault-aware mapping + pruning compared to a simple fault-aware pruning technique. Note that these techniques are orthogonal to the ones discussed for soft error mitigation and, therefore, can be combined with them to offer a wider coverage of reliability threats.

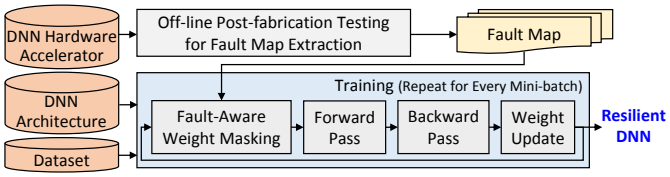

Fig. 5. Overview of the flow of fault-aware training process.

C. Timing Error Mitigation

In the earlier years, the aging of a circuit results in timing errors. Zhang et al. in [25] proposed ThunderVolt, a technique to detect and mitigate these errors in the MAC units of the processing array. The technique uses Razor flip-flops to detect the errors. In case an error is detected, it steals a cycle from the subsequent processing element by dropping its computation to discard the erroneous output and pass-on the approximately correct output. Fig. 8c shows the hardware modification required in the processing elements to realize the technique. Note that ThunderVolt is mainly developed to improve the energy efficiency of a DNN-based system through voltage scaling. However, as it mainly mitigates the effects of timing errors, it can be used to mitigate aging-induced faults.

IV. Challenges on the Road Ahead

Following are some of the key challenges and future directions that are envisioned to play a vital role in realizing dependable DL-based systems.

1) **Frameworks for effective integration of fault-mitigation techniques:** Different systems show a different level of resilience to each type of hardware-induced faults. Therefore, each system may require a different set of fault-mitigation techniques to offer reliable functionality. Towards this, frameworks need to be developed that can efficiently analyze the resilience of DNN-based systems to different types of reliability threats (preferably, in a synergistic way) and propose a sub-set of the available techniques that can be integrated together to offer optimal / cost-effective resilience while meeting the pre-defined resource constraints.
2) **Design Methods for Building Reliable DNNs:** Similar to the research in building resource-efficient DNNs, there is a need for building automated design methodologies that can generate robust DNN models. Towards this, a thorough design space exploration is required to identify the key design principles and building blocks that help in improving the resilience of DNNs. Afterward, methodologies have to be developed that can employ these principles to build reliable and resource-efficient DNNs.
3) **Test and Verification:** The increasing use of DNNs in safety-critical applications has raised the need for "complete" verification of these models. Frameworks are being developed that can formally verify their functionality. However, most of the frameworks focus on small-scale DNNs, or do approximate or incomplete verification. A considerable amount of effort is required in this direction to build frameworks that can efficiently verify the functionality of large-scale DNNs with realistic assumptions. One of the recent studies in this direction is [39], for more details see [40]. An alternate direction towards the verification of ML systems is *Interpretable Machine Learning / Explainable AI*, which focuses on developing models that can be interpreted by humans.
4) **Reliability for SNNs:** Inspired by the spiking-based event-driven information processing in the brain, the Spiking Neural Networks (SNNs) promise to offer low power consumption and high-speed inference when processed using massively parallel neuromorphic hardware [41] [42]. The underlying functions in SNNs and the need for a different type of hardware architecture make them distinct from DNNs. These distinctions ask for novel techniques to analyze and mitigate the effects of hardware-induced reliability threats in SNN-based systems.

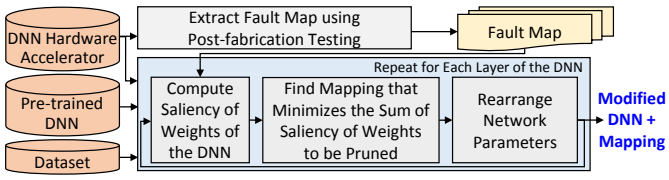

Fig. 6. Overview of the flow of our fault-aware mapping process.

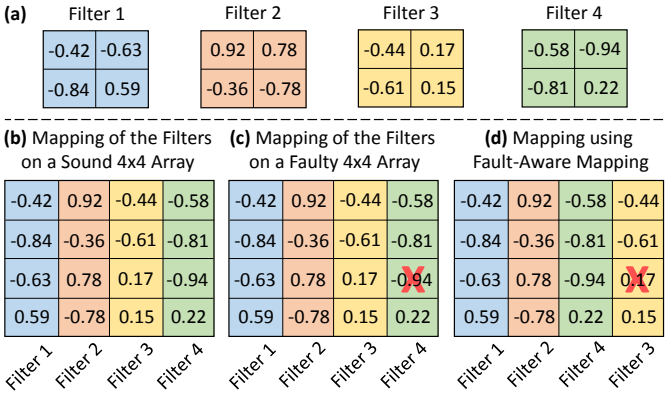

Fig. 7. An example illustrating the working principle of fault-aware mapping.

Acknowledgement

We would like to acknowledge Intel's Grant for the project "*Cost-Effective Dependability for Deep Neural Networks and Spiking Neural Networks*".

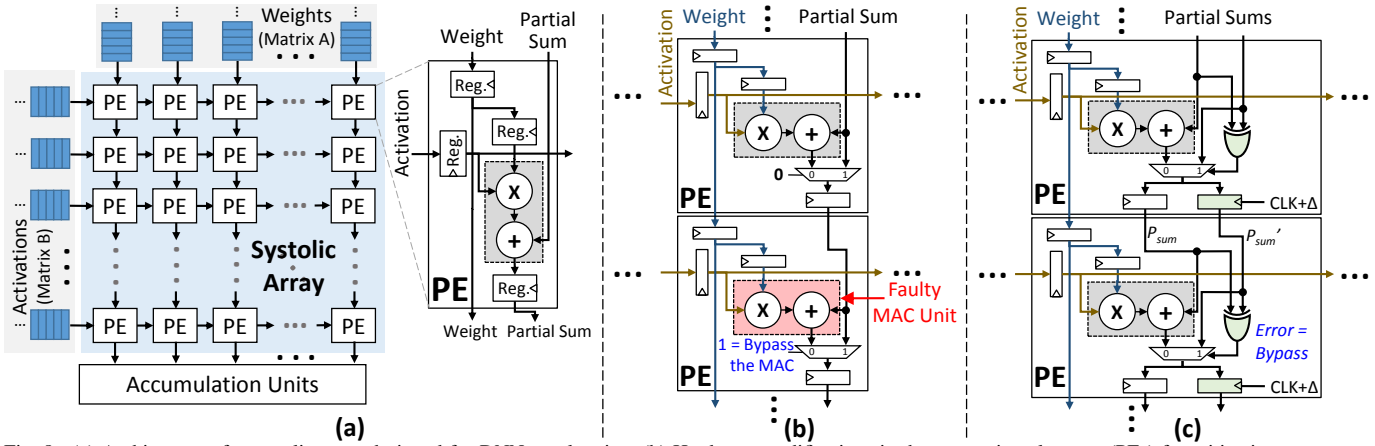

Fig. 8. (a) Architecture of a systolic array designed for DNN acceleration. (b) Hardware modifications in the processing elements (PEs) for mitigating permanent faults. (c) Hardware modifications in the PEs for mitigating timing faults. (Adapted from [23]–[25])

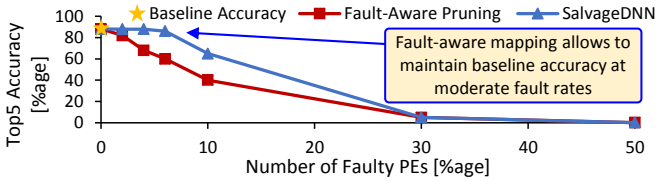

Fig. 9. Comparison between fault-aware pruning and fault-aware mapping + pruning (i.e., SalvageDNN) using VGG11 trained for the ImageNet dataset and deployed on a 256x256 faulty systolic array-based DNN hardware [24].

References

[1] Y. LeCun et al., "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[2] Y. Guo et al., "Deep learning for visual understanding: A review," *Elsevier Neurocomputing*, vol. 187, pp. 27–48, 2016.

[3] H. I. Fawaz et al., "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[4] A. Marchisio et al., "Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges," in *IEEE **ISVLSI***, 2019, pp. 553–559.

[5] D. S. et al., "Mastering the game of go without human knowledge," ***Nature***, vol. 550, no. 7676, pp. 354–359, 2017.

[6] M. Al-Qizwini et al., "Deep learning algorithm for autonomous driving using googlenet," in *IEEE **IV** Symposium*, 2017, pp. 89–96.

[7] B. Huval et al., "An empirical evaluation of deep learning on highway driving," *arXiv preprint arXiv:1504.01716*, 2015.

[8] S. Mozaffari et al., "Deep learning-based vehicle behaviour prediction for autonomous driving applications: A review," *arXiv preprint arXiv:1912.11676*, 2019.

[9] R. Miotto et al., "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[10] A. Esteva et al., "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, p. 24, 2019.

[11] B. S. Prabakaran et al., "Emap: A cloud-edge hybrid framework for eeg monitoring and cross-correlation based real-time anomaly prediction," *arXiv preprint arXiv:2004.10491*, 2020.

[12] V. Sze et al., "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[13] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *ACM/IEEE **ISCA***, 2017, pp. 1–12.

[14] H. Kwon et al., "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects," in *ACM **ASPLOS***, 2018, pp. 461–475.

[15] Y. Chen et al., "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2019.

[16] "Road vehicles – Functional Safety," International Organization for Standardization, Geneva, CH, Standard, 2011.

[17] R. Vadlamani et al., "Multicore soft error rate stabilization using adaptive dual modular redundancy," in *IEEE **DATE***, 2010, pp. 27–32.

[18] R. E. Lyons et al., "The use of triple-modular redundancy to improve computer reliability," *IBM journal of research and development*, vol. 6, no. 2, pp. 200–209, 1962.

[19] X. Vera et al., "Selective replication: A lightweight technique for soft errors," *ACM **TOCS***, vol. 27, no. 4, pp. 1–30, 2010.

[20] M. Shafique et al., "Exploiting program-level masking and error propagation for constrained reliability optimization," in *ACM/IEEE **DAC***, 2013, pp. 1–9.

[21] T. Lin et al., "Microsoft coco: Common objects in context," in ***ECCV***. Springer, 2014, pp. 740–755.

[22] S. Kim et al., "Energy-efficient neural network acceleration in the presence of bit-level memory errors," *IEEE **TCAS-I***, vol. 65, no. 12, pp. 4285–4298, 2018.

[23] J. J. Zhang et al., "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *IEEE **VTS***. IEEE, 2018, pp. 1–6.

[24] M. Hanif at al., "Salvagednn: salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping," *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2164, 2020.

[25] J. Zhang et al., "Thundervolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *ACM/IEEE **DAC***, 2018, pp. 1–6.

[26] Z. Chen et al., "Ranger: Boosting error resilience of deep neural networks through range restriction," *arXiv preprint arXiv:2003.13874*, 2020.

[27] L. Hoang et al., "Ft-clipact: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation," *IEEE **DATE***, 2020.

[28] K. Zhao et al., "Algorithm-based fault tolerance for convolutional neural networks," *arXiv preprint arXiv:2003.12203*, 2020.

[29] J. Guo et al., "Novel low-power and highly reliable radiation hardened memory cell for 65 nm cmos technology," *IEEE **TCAS-I***, vol. 61, no. 7, pp. 1994–2001, 2014.

[30] A. Azizimazreah et al., "Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs," in *IEEE **NAS***, 2018, pp. 1–10.

[31] L.-C. Chu et al., "Fault tolerant neural networks with hybrid redundancy," in *IEEE **IJCNN***. IEEE, 1990, pp. 639–649.

[32] S. Shankland, "Meet Tesla's self-driving car computer and its two AI brains," https://www.cnet.com/news/meet-tesla-self-driving-car-computer-and-its-two-ai-brains/.

[33] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE **T-DMR***, vol. 5, no. 3, pp. 305–316, 2005.

[34] K. Kang et al., "Nbti induced performance degradation in logic and memory circuits: How effectively can we approach a reliability solution?" in *ACM/IEEE **ASP-DAC***, 2008, pp. 726–731.

[35] Z. Chen et al., "Binfi: an efficient fault injector for safety-critical machine learning systems," in *ACM **HPCA***, 2019, p. 69.

[36] B. Reagen et al., "Ares: A framework for quantifying the resilience of deep neural networks," in *ACM/IEEE **DAC***, 2018, pp. 17:1–17:6.

[37] D. B. Limbrick et al., "Reliability-aware synthesis of combinational logic with minimal performance penalty," *IEEE Transactions on nuclear science*, vol. 60, no. 4, pp. 2776–2781, 2013.

[38] M. A. Hanif et al., "Robust machine learning systems: Reliability and security for deep neural networks," in *IEEE **IOLTS***, 2018, pp. 257–260.

[39] M. Naseer et al., "Fannet: Formal analysis of noise tolerance, training bias and input sensitivity in neural networks," *IEEE **DATE***, 2020.

[40] M. Shafique et al., "Robust machine learning systems: Challenges,current trends, perspectives, and the road ahead," *IEEE **D&T***, vol. 37, no. 2, pp. 30–57, 2020.

[41] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE **Micro***, vol. 38, no. 1, pp. 82–99, 2018.

[42] R. Massa et al., "An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor," *IEEE **IJCNN***, 2020.