POLITECNICO DI TORINO
Repository ISTITUZIONALE

Approximate Fault-Tolerant Neural Network Systems

Original
Approximate Fault-Tolerant Neural Network Systems / Traiola, M., Pappalardo, S., Piri, A., Ruospo, A., Deveautour, B., Sanchez, E., Bosio, A., Saeedi, S., Carpegna, A., Göebakan, A.B., Magliano, E., Savino, A.. - ELETTRONICO. - (2024), pp. 1-10. (IEEE European Test Symposium (ETS) 2024 Der Haag (NL) 20-24 May 2024) [10.1109/ets61313.2024.10567290].

Availability:
This version is available at: 11583/2991037 since: 2024-07-19T12:26:36Z

Publisher:
IEEE

Published
DOI:10.1109/ets61313.2024.10567290

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

27 November 2026

Approximate Fault-Tolerant Neural Network Systems

Marcello Traiola[1], Salvatore Pappalardo[2], Ali Piri[2], Annachiara Ruospo[3], Bastien Deveautour[4], Ernesto Sanchez[3], Alberto Bosio[2], Sepide Saeedi[3], Alessio Carpegna[3], Anıl Bayram Göğebakan[3], Enrico Magliano[3], Alessandro Savino[3]

[1]*Univ Rennes, CNRS, Inria, IRISA - UMR 6074, F-35000 Rennes, France*

[2,4]*Univ Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, 69130 Ecully, France*

[3]*Politecnico di Torino, Dip. di Automatica e Informatica, Torino, Italy*

[1]marcello.traiola@inria.fr, [2]{name.surname}@ec-lyon.fr, [3]{name.surname}@polito.it, [4]{name.surname}@cpe.fr

***Abstract*—This paper aims to comprehensively explore challenges and opportunities to design highly efficient Neural Network (NN) systems through Approximate Computing (AxC) techniques while ensuring fault tolerance properties. By highlighting the intrinsic conflicting goals of AxC and fault tolerance principles, the study aims to stimulate and contribute to a deeper understanding of how important it is to consider fault tolerance requirements while designing approximate-computing-based systems. This is key to developing highly efficient fault-tolerant architectures for Neural Networks.**

***Index Terms*—Approximate Computing, Reliability, Neural Networks, Fault Tolerance, Artificial Intelligence**

I. Introduction

In computing systems, resilience stands as a cornerstone principle, ensuring continuous operation and robustness in the face of diverse challenges, including hardware faults. Traditional approaches to strengthen system resilience often entail significant overheads regarding resources and energy consumption. However, a promising alternative has emerged in Approximate Computing (AxC) that offers an interesting opportunity to enhance resilience while mitigating the associated costs [1].

AxC, characterized by its relaxation of accuracy requirements in favor of efficiency gains, presents a novel paradigm to potentially improve system resilience. By leveraging the inherent trade-offs between accuracy and resources, AxC techniques can protect systems against various failure modes, enabling graceful degradation and adaptive response mechanisms [2]. As highlighted throughout the paper, this might seem counterintuitive; however, upon a careful design phase, AxC can help improve both the resilience and efficiency of Neural Network (NN) systems.

This paper investigates the conflicting requirements of approximate computing and reliability, highlighting the need for careful design to improve system resilience and efficiency. The exploration begins with examining the AxC techniques applied to NN systems over the years, discussing the impact of AxC in the NN domain and existing countermeasures to accuracy degradation. Furthermore, the paper explores the existing fault-tolerant strategies applied to NN systems. Then, the journey continues, investigating the delicate balance between approximation and reliability. We show the existing approaches to assess the reliability of approximate NN systems and how AxC has been used to improve NN fault tolerance. We focus on the techniques to improve the reliability of the NNs leveraging the accuracy variability and the intrinsic error masking of approximate techniques, as well as on the limitations to the reliability when approximate techniques are a mandatory design constraint. While approximate computing holds interesting promise for enhancing system resilience, it also poses challenges and considerations that must be addressed. Thus, we identify key aspects to consider when designing approximate fault-tolerant NN systems and highlight the need for careful *Approximation for Reliability (AfR)* design principles to enhance both the resilience and efficiency of such systems.

The rest of the paper is organized as follows. A background section discussing the fundamentals of AxC and resiliency concepts (Section II), an overview of state-of-the-art AxC techniques in NN systems (Section III), a review of reliability analysis approaches for NN systems (Section IV). Section V reviews existing approaches to balance approximation and reliability to finally highlight challenges, opportunities, and key aspects to address. Finally, Section VI provides concluding remarks, including perspectives and open challenges.

II. Background

The usage of AxC has been linked for many years with all applications that inherently possess the resilience to errors, meaning they can produce satisfactory outputs even when some of their computations are carried out in an approximate manner [3]. This is the case for many Machine Learning (ML) applications, where the classification results can be delivered with some error tolerance without compromising the task [4].

The concept of AxC encompasses a wide array of techniques that capitalize on the inherent resilience of applications, ultimately leading to improved efficiency across all computing stack layers, ranging from the fundamental transistor-level design to software implementations. These techniques can have varying impacts on both the hardware and the output

quality. Among them, one of the most common is employing approximate adders or multipliers [5]–[7] or resorting to data precision reduction to smaller and approximated representation, such as fixed point or integer quantization [8].

Error assessment usually involves simulating both the precise and approximate versions of applications [1] and measuring the deviation with standard metrics, such as Mean Squared Error (MSE), Mean Absolute Error (MAE) or more application-dependent ones, such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSI). However, the scientific literature has proposed alternative methods like Bayesian inference [9] or machine learning-based approaches [10].

Selecting the most suitable AxC techniques for an application is challenging, and many publications have proposed how to explore the design space to find the most suitable AxC techniques for an application [11]–[14]. Among various approaches, such as complex Design Space Exploration (DSE), including genetic algorithms and simulated annealing [15], ML ones such as Reinforcement Learning (RL) showed promising results in improving the DSE [16], [17], minimizing the number of designs to evaluate while maximizing the quality of the DSE model and reducing the exploration time [18].

Moving from approximate computing to Fault Injection (FI) [19], [20] and fault models can be a natural progression in exploring reliability and resilience in computing systems. While approximate computing intentionally introduces controlled inaccuracies to improve efficiency, FI techniques delve into unintended errors and their consequences. By injecting faults into hardware or software components, researchers can simulate real-world scenarios where errors occur due to manufacturing defects, environmental conditions, or malicious attacks. Fault models provide a systematic framework for characterizing different types of faults and their effects on system behavior, ranging from transient faults that cause temporary errors to permanent faults that result in persistent failures [21]. By studying fault injection and fault models in conjunction with approximate computing, researchers can gain insights into the interplay between intentional approximations and unintentional errors, leading to the development of more resilient and robust computing systems capable of withstanding a wide range of adverse conditions.

FI can be classified based on the mechanisms used to introduce the fault into the system artificially: (i) Physical FI aims at exposing the final implementation of the system to an external fault source (e.g., radiation-based fault injections), while (ii) Hardware-based FI exploits existing hardware interfaces to alter the behavior of the system (e.g., using debug interface to access to Central Processing Unit (CPU) internal registers), and (iii) Software-based FI instruments the software layer of the system to inject faults directly into memory locations. Finally, the (iv) model-based FI instruments the model of the system (e.g., Hardware Description Language (HDL) model or by exploiting micro-architectural description of the CPU/system) to inject faults during model simulation/emulation. Examples and details of the above techniques can be found in [19], [20].

Every injection could lead to different types of behavior in the system, classified as follows [22]:

- **OK:** the system continues working expectedly, without showing any appreciable difference concerning the golden run after the injection;
- **Crash**: whenever the system stop its activities. It includes forced reboots or shutdown of the system.
- **Hang**: despite being active, the system does not end after a time window of observation.
- **Silent Data Corruption (SDC)**: despite the system being able to end its task, the output of the computation is altered. To check if the computation results are correct, FI should follow a careful comparison of any system outcomes with a golden version obtained without injecting any error.

III. Approximate Computing for Neural Network systems

It appears that the use of AxC in ML dates back to 2014, when inexact multipliers were applied to a NN accelerator [23], achieving savings in energy, delay, and area at the cost of a moderate accuracy loss for a multi-layer perception. From there, numerous propositions appeared from the scientific community at different abstraction levels [24]. Starting with simple classifiers, scalable-effort classifiers were proposed to dynamically adjust computational effort based on input difficulty while maintaining accuracy [25]. ApproxANN [26] introduces an approximate computing framework for Artificial Neural Networks (ANNs). It applies approximation to computation and memory accesses by carefully characterizing the impact of neurons on output quality for maximum energy efficiency gain under a specified quality constraint.

Concerning Convolutional Neural Networkss (CNNs), the error resilience of parameters has been investigated [27] to determine the least critical ones to approximate. As a result, CNN filters with the least significance can be pruned, enabling trade-offs between accuracy and efficiency. Another approach is quantifying the error resilience of each neuron through theoretical analysis for CNNs [28] and Spiking Neural Networks (SNNs) [29], revealing significant variability across neurons. This enables dynamic adaptation of approximate bit-width and configurable approximate circuits depending on the error resilience of neurons.

Indeed, the use of approximate multipliers to enhance the performance of CNNs has been explored [30]–[32], showing that replacing part of or all exact multipliers with approximate ones has a low impact on the networks' accuracy while achieving significant performance gains in terms of power, area, and speed. In this sense, ALWANN [33] introduces a novel approach to select suitable approximate 8-bit multipliers for each computing element of a custom low-power accelerator by converting fully trained Deep Neural Networks (DNNs) to operate with 8-bit weights and multipliers. A multi-objective optimization algorithm is used to minimize classification error and energy consumption. Also, Neural Architecture Search (NAS) approaches have been proposed to incorporate approximate operations into CNNs and achieve power-efficient hardware implementations, based on Cartesian genetic programming [34] and differentiable architecture

search (DARTS) [35]. Approximate multipliers have also been used to improve the training performance of CNNs [36].

Concerning quantization, studies on developing CNNs with flexible data types have been conducted [37]. Various data types for inference and training procedures can be used, often layer-wise [38], along with incorporating approximate arithmetic operations into the CNN design. It has been shown that even 2-bit weights can lead to a reasonable accuracy after fine-tuning [37] and also Binary Neural Networks (BNNs) have been extensively studied [39]. Other approaches have been proposed to compress the storage requirements of NNs systems without compromising their accuracy. An example consists of applying pruning to remove unimportant connections, then the weights are quantized to enforce weight sharing, and finally, Huffman coding is used to compress the network representation [40].

Quantization and voltage scaling techniques have been used to enhance the energy efficiency in resource-constrained devices deploying CNNs [41]. Also, dedicated approximate HW accelerators have been designed based on conventional [42] or emerging computing paradigms, such as In-Memory Computing (IMC) and Near-Memory Computing (NMC) [43] and utilizing reconfigurable nanotechnologies such as Silicon Nanowire Field Effect Transistors (FETs) [44]. Finally, propositions to use distributed approximate systems for collaborative DNN have also been explored [45] to achieve energy-efficient inference for resource-constrained environments such as edge computing.

NN-based systems approximation works well thanks to their inherent tolerance. In fact, at some level, the NN purpose comes down to approximating a complex and incompletely specified function [24]. Through the training process, NNs imitate the behavior of such a function. However, the training process is usually performed under the fully-accurate computation assumption. Thus, when too much approximation is introduced, the system accuracy decreases. Therefore, the training process can be re-purposed to recover from the adverse effects of approximations.

The backpropagation algorithm has been used for quantifying approximation impact and incremental retraining to mitigate quality loss [46]. Post-approximation re-training and approximation-aware (re-)training are often used to recover from the introduced error [40]. By integrating hardware imperfections into the training phase, the accuracy of DNN models can be recovered significantly even with aggressive approximation [47]. Methodologies have been proposed for efficiently selecting approximate multipliers and applying retraining to minimize the approximation error [48]. Approaches to reduce the necessary retraining effort have also been used, such as knowledge distillation and gradient estimation [49]. Also, the behavioral simulation of various approximate multipliers to model the error introduced by approximate hardware in DNNs has been used to accelerate retraining [50].

Frameworks have been developed to simulate the introduction of approximation during training and inference. AxTrain [51] is a hardware-oriented training framework that actively searches for a network parameter distribution with high error tolerance and passively learns resilient weights by numerically incorporating the noise distributions of approximate hardware during training. ProxSim [52] and TFApprox [53] are Graphics Processing Unit (GPU)-accelerated simulation frameworks designed to optimize cross-layer approximate DNN. Approximate DNN inference and retraining are supported while introducing a hardware-aware regularization technique for DNNs optimization. TorchAxf [54] also includes SNN and supports different types of approximate adders and multipliers, as well as standard reduced precision floating-point formats like bfloat16 and user-customized precision representations. More recently, the development effort has been directed towards larger Artificial Intelligence (AI) models, such as Vision Transformers (ViTs). TransAxx framework [55] enables the use of approximate computing for ViTs. It uses sensitivity analysis to approximate multiplications, approximate-aware fine-tuning for accuracy restoration, and a methodology for generating approximate accelerators using a Monte Carlo Tree Search algorithm.

For a more complete overview of the existing literature on Approximate Computing for NNs systems, we refer the reader to existing surveys [5], [24], [56], [57]

IV. Reliability of Neural Network systems

The field of Reliability of NNs systems traces its origins to 2001 when one of the first studies [58] focused on fault tolerance in ANNs, particularly multilayered feedforward nets. The study aims to assess the real influence of faults on neural computation and evaluate the network's ability to survive faults. It adopts a high abstraction level based on neural graphs and introduces an error model to derive fault-induced errors' effects on neural outputs analytically. It also draws the conditions for complete compensation of errors through weight adjustment. The experimental evaluation indicated that even single errors significantly affect computation, and weight redistribution is insufficient for complete masking post-fault, thus suggesting architectural redundancy to achieve fault tolerance.

A. Reliability Assessment of NN-based Systems

Many studies have analyzed the phenomenon from different points of view. From the NN model perspective, the robustness of multi-layer neural networks has been explored on the hypothesis that neurons fail independently [59], providing bounds on the number of neurons that can fail without affecting computation. Since the NN system reliability competes with power efficiency and latency, methods for predicting the error resilience of neurons have been explored in DNNs [60], as well as in SNNs [61], for example, based on vulnerability value ranges [62]. The resilience estimations pave the way to selective protection for flexible trade-offs between reliability and efficiency.

From the hardware point of view, the error propagation in DNN accelerators has been investigated, highlighting that the resilience depends on factors like data types, values, data reuses, and layer types [63]. Different HW platforms have been used to accelerate DNNs. Thus, several studies assess the impact of permanent and transient faults on such platforms.

A variety of approaches have been proposed, from the most realistic neutron beam exposure [64]–[68] to combining gate-level fault simulation accuracy with software fault injection's speed and flexibility [69]–[72].

Many tools for resilience evaluation through FI have been developed throughout the years. SASSIFI [73] was designed for evaluating, through assembly-language instrumentation, the resilience of massively parallel applications running on modern NVIDIA GPUs to soft errors caused by high-energy particle strikes. A fault injection environment built on the darknet DNN framework has also been proposed [74] for evaluating the impact of permanent faults on CNNs used in automotive applications. High-level fault injection frameworks for both TensorFlow-based applications (TensorFI [75] and InjecTF2 [76]) and Pytorch-based applications (PyTorchFI [77] and TorchFI [78]) were proposed. They enable the injection of hardware and software faults into TensorFlow programs, allowing assessment of their resilience across various fault types. Fiji-FIN [79] was designed for evaluating the resiliency of IoT devices executing ML and Deep Learning (DL) models, mainly for security challenges like bit perturbation attacks and soft errors. Ways to provide a fast and accurate solution for injecting software-level faults in DNNs were embedded in approaches such as BinFI [80], Enpheeph [81], and SCI-FI [82]. While BinFI models the error propagation characteristics of a machine learning application as a monotonic function, the SCI-FI approach focuses on running the faulty DNN selectively by rapidly determining whether fault effects should be propagated throughout the inference. Enpheeph was designed specifically for fast injections on SNNs and compressed (sparse) DNNs. As mentioned, cross-layer fault-injection approaches [71] have been proposed to speed up the evaluation. Error models derived from accurate fault injection campaigns are used within error simulation engines, enabling fast and accurate assessments.

It is then clear that FI is a standard method for assessing the reliability of integrated systems. Due to the complexity of modern systems, only a subset of potential errors is randomly selected for practical experiments. A well-established method [83] provides an analytical way to obtain the necessary number of faults to inject to achieve a desired confidence level and error interval. Specific research for the particular case of NN systems has been conducted [4], providing a methodology to compute the statistically significant number of faults to be injected and the fault injection points to achieve statistically significant results.

B. Reliability Improvement of Neural Network systems

Over the years, many approaches have been proposed to improve the reliability of NN systems. Traditional resilience methods like Triple Modular Redundancy (TMR) incur high overheads and are unsuitable for DNN algorithms or accelerator architectures. Thus, the sensitivity to faults of different model and accelerator parts is assessed to enable cost-effective selective protection.

Many approaches target pre-trained networks, as the dataset to apply (re-)training-based approaches is not always available, and the cost is often prohibitive, especially for huge models. Specific methods aim to reduce worst-case failure rates after a bit-flip fault by employing online approaches based on selective TMR to counteract the effect of faults [84], [85]. Others modify the parameters of the NN to include Error Correction Codes (ECCs) [86]–[89], or use ML-based approaches to identify critical parameters and apply selective ECC-based memory protection [90], [91].

Approaches at the scheduling level have also been proposed to distribute the computation of critical neurons among the available processing elements and reduce the likelihood that a physical fault may impact multiple critical neurons [92]. Ensemble learning has also been used to enhance robustness against memory errors in embedded systems [93] while maintaining memory requirements. Approaches using security primitives to improve the reliability of NNs also exist [94].

When training data is available, and (re-)training is feasible, further enhancements are possible. Indeed, leveraging the training procedure enables the embedding of fault tolerance within the model. Some studies apply weight distribution fine-tuning [95] after carefully estimating neuron resilience. Fault-Aware Training (FAT) technique [96], [97] relies on injecting faults during the NN training to make it learn the possible faulty scenarios. Other approaches integrate ECC into the NN parameters during training [98], so the correcting code becomes part of the parameters. These approaches aim to improve the model fault tolerance without impacting the memory footprint.

For a more complete overview of the existing literature on the Reliability of NNs, we refer the reader to existing surveys [99]–[104].

V. Balancing Approximation and Reliability in NN systems: Challenges and Opportunities

Reliability and AxC are tightly related, as both rely on the NN inherent resiliency to errors. However, they are conflicting, as the goal of AxC to reduce the system redundancy - hence the overhead - goes against the inherent requirement for redundancy in reliability. Therefore, these two aspects should be jointly considered when designing NN systems to produce highly optimized and reliable solutions. In this section, we show - to the best of our knowledge - the existing literature studies in this direction.

A. State of the art on reliability assessment of approximate NN systems

First, let us consider studies about reliability assessment of approximate NN systems. Quantization is one of the most used AxC approaches to improve efficiency. Thus, the impact of using mixed-precision Floating-Point Operation (FLOP) on the reliability of DNNs executing on modern computing architectures - including Xilinx FPGAs, Intel Xeon Phis, and NVIDIA GPUs - has been explored. Results show that, while the number of wrong outputs in CNN execution tends to increase [105], the performance-reliability trade-off significantly improves when using lower precision data [106], as the Mean Executions Between Failures (MEBF) increase. Moreover,

fixed-point data representation offers the best trade-off between memory footprint reduction and CNN resilience [107].

Studies have been carried out on the impact of faults in approximate deep neural network accelerators, utilizing state-of-the-art approximate operations [6]. Faults can exacerbate accuracy loss in approximate accelerators compared to accurate DNN accelerators [108]. However, in some cases, approximate operations can help improve resilience [109]. We show such a phenomenon in the case study reported in Section V-C. On the one hand, this confirms once again that AxC and Reliability have conflicting goals. On the other hand, it shows that they are not necessarily incompatible, and careful and tailored approaches can help improve both fault resilience and efficiency [110]–[112].

Also, tools for exploring approximation and reliability trade-offs in DNN accelerators are starting to be proposed. As an example, DeepAxe [113] focuses on FPGA-based implementations. It enables selective approximation and provides different trade-offs in the design space considering accuracy, reliability, and hardware performance. Moreover, functional approximation and AxC errors have been used to emulate the effect of faults and speed up the resilience assessment [114], [115].

On the side of SNN, recent works are investigating both the impact of AxC as a tool for minimizing the power consumption and area footprint, as well as assessing the reliability fall out of such instruction. Authors in [116] propose a systematic way of exploring precision reduction on all the memory parameters of SNNs, resorting to Interval Arithmetic (IA) modeling of the error. Once the precision reduction is introduced in the system, it works as [117], investigates the fault-tolerance of the overall inference of the networks in terms of accuracy. Authors proposed an extension of the fault injection tool presented in [82] to target SNNs, and the main takeout of their analysis confirms that the bigger layers of the network, where the generation of spikes works in collaboration among neurons are still very resistant to faults. On the contrary, the fault tolerance reduces drastically when reaching the final layers, with only a few neurons partitioning the output space.

B. State of the art on reliability improvement of approximate NN systems

As supply voltage scaling has been used for energy-efficient DNN accelerator design, a sensitivity-based error resilience technique has been proposed in [118] to schedule more sensitive computations to more robust logic units. The in-memory architecture presented in [119] is based on Bit-line Computing and incorporates effective error detection and mitigation to enable aggressive voltage scaling while maintaining high CNN accuracy. The study in [120] jointly considers error resilience, efficiency, and performance optimization by using an evolutionary, multi-objective Neural Architecture Search (NAS) optimization technique for automatically designing hardware-optimized DNN architectures. To enable rapid evaluation of DNN architectures, the study derives computationally inexpensive objective functions to predict error resilience, energy consumption, latency, and required bandwidth of DNNs on hardware. AdAM [121], a novel adaptive fault-tolerant approximate multiplier for ASIC-based SNN accelerators, employs neutron beam exposure and fault injection adaptive adder utilizing the leading one position value of inputs for fault detection, optimizing unutilized adder resources. A lightweight fault mitigation technique sets detected faulty bits to zero. [122] propose a fault mitigation method to counter permanent faults on approximate DNN accelerators described in [108]. They propose a Fault-aware weight re-tuning, which updates weights to minimize the absolute difference between approximate and accurate multiplications of overall activations in the presence of faults.

While the studies mentioned so far investigate the reliability improvement of approximated NN systems, other studies explore the concept of *approximate fault-tolerance* for CNN accelerators to reduce performance overheads while ensuring reliability. The study in [123] investigates selective duplication and imprecise checking, demonstrating considerable reliability improvements with limited overhead. Finally, runtime adaptation of quantization has been used to achieve lightweight fault tolerance. Even in the presence of faults, a fail-degraded operating mode is proposed in [124], leveraging reduced precision computations to preserve compute capability rather than losing it entirely.

C. Case Study

Conventional fault-tolerant approaches leverage redundancy or reconfiguration to tolerate the presence of HW faults. Even if the overhead of those techniques is reduced thanks to the use of AxC, it can still be too high. In [109], the authors introduced the idea of only using the AxC paradigm to increase the resilience of DNN without any other redundancy techniques and thus avoiding extra costs. We extended the proposed idea by analyzing bit-width reduction and functional approximation.

Let us first define the adopted DNN hardware accelerator. It is based on an output stationary systolic array. Figure 1a shows the general architecture. The weights flow "vertically" while the activations flow "horizontally". Figure 1b shows a functional diagram of a Processing Element (PE). A PE performs a Multiply-accumulate Operation (MAC) operation; this means that for each clock cycle:

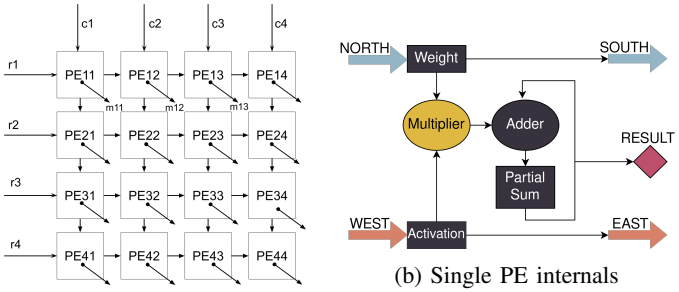
- `north` and `west` inputs are multiplied together,
- the result is added to the partial sum register,
- the value of `north` is put on the `south` output,
- the value of `west` is put in the `east` output.

These operations accumulate a series of multiplications and forward data to the neighbors. In this implementation, weights flow from north to south, and the activations from west to east are accumulated in the PE.

The systolic array has been implemented with two data representations: the INT 16-bit as the reference (i.e., precise) and the INT 8-bit as the approximated precision. Concerning the functional approximation technique, we conducted several experiments leveraging on different publicly available approximate multipliers from the EvoApproxLite [6].

TABLE I: Summary of Experimental Results.

#	Bit-width	Multiplier	Energy Reduction [%]	Accuracy [%]	Injected Faults [%]	Masked [%]	Tolerable [%]	Critical [%]
Baseline	16	Precise	–	99.07%	10%	47.58%	29,18%	23.24%
1	8	Precise	50%	99.05%	19%	64.65%	23.01%	12.34%
2	8	mul12s_2PT	50.3%	99.08%	19%	63.9%	23.79%	12.3%
3	8	mul12s_2QH	51.21%	99.1%	19%	38.92%	44.85%	16.23%
4	8	mul12s_2R5	52%	99.06%	19%	26.96%	55.69%	17.34%
5	8	mul12s_34P	55%	**98.24%**	19%	**74.16%**	**23.01%**	**2.83%**
6	8	mul12s_2TE	55.6%	9.8%	19%	3.94%	27.5%	68.76%



(a) Custom systolic array

(b) Single PE internals

Fig. 1: Systolic array architecture

*1) **Resilience Analysis**:* The resilience analysis is based on the framework published in [67]. We injected transient faults (i.e., modeled as single bit-flip) affecting the inputs of PEs in the systolic array. The faults' universe is then 3-dimensional, and its size depends on both the array size and the bit-width:

$$\text{FaultSpace} = 2 \times K \times M \tag{1}$$

where M is the number of systolic array's PE, and K is the bit-width of the PE (i.e., 8- or 16-bit). Please note that we injected faults in the PE input corresponding to the weights of the DNN. Each fault corresponds to an alteration of one bit, forcing its value to either '1' or '0'.

A statistical FI was configured by assuming a 1% margin of error, 50% probability of a fault failing, a cut-off point of 2.58 which corresponds to 99% confidence level [4].

Two outputs are collected for each input: the fault-free $\hat{Y}$ and the faulty output $\tilde{Y}$. They are vectors of 10 components, each corresponding to the probability of a class. The classification labels $\hat{y}$ and $\tilde{y}$ are obtained as $\hat{y} = argmax(\hat{Y})$ and $\tilde{y} = argmax(\tilde{Y})$. For the purpose of the resilience classification, $\hat{Y}$ and $\tilde{Y}$ are compared for each pair input-fault as follows:

- we classify a fault as **Masked** when $\hat{Y} = \tilde{Y}$,
- **Tolerable** when
  - $\hat{y} = \tilde{y}$ and $\frac{max(\tilde{Y})}{max(\hat{Y})} > 1$,
  - $\hat{y} = \tilde{y}$ and $0.95 < \frac{max(\tilde{Y})}{max(\hat{Y})} < 1$,
- **Critical** when
  - $\hat{y} = \tilde{y}$ and $\frac{max(\tilde{Y})}{max(\hat{Y})} < 0.95$,
  - $\hat{y} \neq \tilde{y}$.

In other words, we check whether the top-1 probability of the injected network $max(\tilde{Y})$ is greater or smaller than the golden top-1 probability $max(\hat{Y})$ and classify the fault accordingly. Note that **masked** faults do not produce any difference in the output, while only **critical** faults involve misclassification.

*2) **Results**:* The LeNet-5 CNN architecture, trained and tested on the MNIST dataset, has been used to validate the technique's effectiveness. It is composed of three convolutional layers and four fully connected. The training process was performed using the accurate version of the network (i.e., without quantization and functional approximation). The training parameters are: learning rate started at 0.05, with the decay of 5×10^{-4} every 375(*128) iterations, and the momentum was set to 0.9. The final accuracy of the model was equal to 99.05%. We then performed 16- and 8-bit post-training quantization through the following steps.

1) all weights are rescaled in the range $[-1.0, 1.0]$, and activations at each layer are rescaled in the range $[-1.0, 1.0]$ for signed outputs and $[0.0, 1.0]$ for unsigned outputs;
2) inputs, weights, biases and activations are quantized to the desired n_{bits} by converting $[-1.0, 1.0]$ and $[0.0, 1.0]$ to $[-2^{n_{bits}-1} - 1, 2^{n_{bits}-1} - 1]$ and $[0, 2^{n_{bits}-1} - 1]$

The target CNN is then deployed in our systolic array, sized as 28×28 in the configurations depicted in Table I. The Table reports the different systolic array configurations (numbered from 1 to 6 in the first column). Each configuration is characterized by its bit-width precision (16/8 bits) and the type of approximate multiplier (second and third columns). Columns four and five report the performance regarding energy reduction w.r.t the baseline and the classification accuracy. The last columns show the results of the FI campaign expressed in terms of fault percentage of injected faults w.r.t. the total one [4], and the FI outcomes (i.e., rate of Masked, Tolerable and Critical faults) found in every fault injection campaign.

As expected, the approximation positively impacts the energy reduction at a minimal accuracy degradation except for the configuration #6, for which the multiplier has very low precision. The interesting observation is the impact on resilience, represented here as injection outcomes of the approximation. Firstly, bit-width reduction reduces by a factor of 2 the amount of critical faults. Second, the approximate multiplier significantly impacts resilience, and the impact depends on the adopted multiplier. We can cite the case #5 for which we reduced 8x the number of critical faults without adding redundancy mechanisms.

D. Discussion

Throughout the paper, we highlighted that Reliability and AxC are conflicting but not necessarily incompatible. Careful

design approaches must be adopted for fault-resilient and efficient NN systems. We identified the following key aspects to be considered when designing NN systems to produce highly optimized and reliable solutions.

1) *Low-precision data*: quantization typically worsens the effect of a fault impacting the CNN execution, but the performance gains increase the Mean Executions Between Failures. A fault- and quantization-aware training process seems a promising direction to achieve resilient and efficient NN systems.
2) *Fault-tolerant approximate operators*: functional approximation has demonstrated very good results in achieving energy efficiency but often impacts fault tolerance negatively. Carefully adding fault tolerance features to approximate operators might allow them to be used consistently in safety-critical contexts.
3) *Lightweight fault tolerance approaches*: in some contexts, it might be enough to allow a controlled error until the system's normal operation can be restored. Lightweight monitors and mitigators can hence help in this direction.
4) *Models and tools*: integrated design solutions for approximate yet resilient NN systems for safety-critical applications are needed, paving the way to *Approximation for Reliability* design principles. To cope with the increasing complexity of systems, models, and tools to predict and control the impact of the approximation on efficiency and reliability should be designed to be embedded in new approximation-based design flows.

VI. Conclusion

This paper has explored the foundational aspects of the relationship between AxC and resiliency within NN systems, presenting a comprehensive overview of current techniques and methodologies. Through our analysis of state-of-the-art approaches, we have highlighted the overlapping aspects of reliability and AxC techniques, highlighting potential synergies and challenges. Addressing these challenges and exploring further opportunities for integration will be vital for enhancing the robustness and resilience of neural networks and coming to *Approximation for Reliability* design principles. Practical use cases presented in the paper demonstrate the inner resiliency of AxC techniques that, when tolerable by the application, can enhance the fault tolerance, helping the designer apply fault tolerance techniques only in critical parts of the systems, reducing the overall costs. We identified some key aspects to consider while designing NN systems for generating highly optimized and reliable solutions.

This paper serves as a foundation for future research in this critical domain, offering perspectives on future directions and highlighting open challenges to investigate further.

Acknowledgments

This work has been supported by the National Resilience and Recovery Plan (PNRR) through the National Center for HPC, Big Data and Quantum Computing, and from the APROPOS project in the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956090. This research was partially funded by the ANR-21-CE24-0015 "RE-TRUSTING" project and by the ANR France 2030 AdaptING project, "ANR-23-PEIA-0009"

References

[1] A. Bosio *et al.*, "Design, verification, test and in-field implications of approximate computing systems," in *2020 IEEE European Test Symposium (ETS)*, 2020, pp. 1–10.

[2] A. Savino *et al.*, "Approximate computing design exploration through data lifetime metrics," in *2019 IEEE European Test Symposium (ETS)*, 2019, pp. 1–7.

[3] V. K. Chippa *et al.*, "Analysis and characterization of inherent application resilience for approximate computing," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–9.

[4] A. Ruospo *et al.*, "Assessing convolutional neural networks reliability through statistical fault injections," in *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2023, pp. 1–6.

[5] H. Jiang *et al.*, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[6] V. Mrazek *et al.*, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 258–261.

[7] A. Piri *et al.*, "Input-Aware Approximate Computing," in *2022 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, May 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9801944

[8] M. Traiola *et al.*, "Predicting the impact of functional approximation: from component- to application-level," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, 2018, pp. 61–64.

[9] A. Savino *et al.*, "Efficient neural network approximation via bayesian reasoning," in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2021, pp. 45–50.

[10] V. Mrazek *et al.*, "autoax: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317781

[11] W. Hu *et al.*, "Exploring the design space of approximate arithmetic circuits using reinforcement learning," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 442–447.

[12] S. Barone *et al.*, "Multi-Objective Application-Driven Approximate Design Method," *IEEE Access*, vol. 9, pp. 86 975–86 993, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9449861

[13] M. Barbareschi *et al.*, "A Genetic-algorithm-based Approach to the Design of DCT Hardware Accelerators," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 18, no. 3, pp. 1–25, Jul. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3501772

[14] M. Barbareschi *et al.*, "Automatic Approximation of Computer Systems Through Multi-objective Optimization," in *Design and Applications of Emerging Computer Systems*, W. Liu *et al.*, Eds. Cham: Springer Nature Switzerland, 2024, pp. 383–420. [Online]. Available: https://doi.org/10.1007/978-3-031-42478-6_15

[15] M. Rizakis *et al.*, "Approximate fpga-based lstms under computation time constraints," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, N. Voros *et al.*, Eds. Cham: Springer International Publishing, 2018, pp. 3–15.

[16] Y. Wu *et al.*, "Ironman: Reinforcement learning based design space exploration for approximate computing," in *2021 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2021, pp. 1–8.

[17] S. Saeedi *et al.*, "Design space exploration of approximate computing techniques with a reinforcement learning approach," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2023, pp. 167–170.

[18] Q. Gautier *et al.*, "Sherlock: A multi-objective design space exploration framework," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 4, pp. 1–20, 2022.

[19] G. Di Natale *et al.*, Eds., *Cross-layer reliability of computing systems*. Place of publication not identified: INST OF ENGIN AND TECH, 2020, oCLC: 1191709900. [Online]. Available: http://public.eblib.com/choice/PublicFullRecord.aspx?p=6341977

[20] P. Bodmann *et al.*, "Soft error effects on arm microprocessors: Early estimations vs. chip measurements," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[21] M. Bushnell *et al.*, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer Publishing Company, Incorporated, 2013.

[22] A. Vallero *et al.*, "Cross-layer system reliability assessment framework for hardware faults," in *2016 IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–10.

[23] Z. Du *et al.*, "Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators," in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 201–206.

[24] J. Henkel *et al.*, "Approximate computing and the efficient machine learning expedition," in *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022, pp. 1–9.

[25] S. Venkataramani *et al.*, "Scalable-effort classifiers for energy-efficient machine learning," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2744769.2744904

[26] Q. Zhang *et al.*, "Approxann: An approximate computing framework for artificial neural network," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar*, pp. 701–706.

[27] M. Hanif *et al.*, "Error resilience analysis for systematically employing approximate computing in convolutional neural networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar*, pp. 913–916.

[28] C. Wu *et al.*, "ynamic adaptation of approximate bit-width for cnns based on quantitative error resilience," in *2019 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pp. 1–6.

[29] S. Sen *et al.*, "Approximate computing for spiking neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 193–198.

[30] I. Hammad *et al.*, "Impact of approximate multipliers on vgg deep learning network," *IEEE Access*, vol. 6, pp. 60 438–60 444,.

[31] M. Ansari *et al.*, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Trans. Very Large Scale Integr. VLSI Syst*, vol. 28, no. 2, pp. 317–328,.

[32] O. Spantidi *et al.*, "How much is too much error? analyzing the impact of approximate multipliers on dnns," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, 2022, pp. 1–6.

[33] V. Mrazek *et al.*, "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Nov*, pp. 1–8.

[34] M. Pinos *et al.*, "Evolutionary approximation and neural architecture search," *Genetic Programming and Evolvable Machines*, vol. 23, no. 3, p. 351–374, sep 2022. [Online]. Available: https://doi.org/10.1007/s10710-022-09441-z

[35] M. Pinos *et al.*, "Approxdarts: Differentiable neural architecture search with approximate multipliers," 2024.

[36] I. Hammad *et al.*, "Deep learning training with simulated approximate multipliers," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dec*, pp. 47–51.

[37] P. Rek *et al.*, "Typecnn: Cnn development framework with flexible data types," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar*, pp. 292–295.

[38] C. Parra *et al.*, "Exploiting resiliency for kernel-wise cnn approximation enabled by adaptive hardware design," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5.

[39] C. Yuan *et al.*, "A comprehensive review of Binary Neural Network," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12 949–13 013, Nov. 2023. [Online]. Available: http://arxiv.org/abs/2110.06804

[40] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding."

[41] B. Denkinger, "Impact of memory voltage scaling on accuracy and resilience of deep learning based edge devices," *IEEE Des. Test*, vol. 37, no. 2, pp. 84–92,.

[42] F. Kreß, "Atlas: An approximate time-series lstm accelerator for low-power iot applications," in *2023 26th Euromicro Conference on Digital System Design (DSD)*, pp. 569–576.

[43] F. Ponzina *et al.*, "Overflow-free compute memories for edge ai acceleration," *ACM Trans. Embed. Comput. Syst*, vol. 22, no. 5s, pp. 1–121 23,.

[44] R. Saravanan *et al.*, "Reconfigurable fet approximate computing-based accelerator for deep learning applications," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5.

[45] A. Das *et al.*, "Towards energy-efficient collaborative inference using multi-system approximations," *IEEE Internet Things J*, pp. 1–1,.

[46] S. Venkataramani *et al.*, "AxNN: energy-efficient neuromorphic systems using approximate computing," in *Proceedings of the 2014 international symposium on Low power electronics and design*, ser. ISLPED '14. New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 27–32. [Online]. Available: https://dl.acm.org/doi/10.1145/2627369.2627613

[47] Y. Fan *et al.*, "Axdnn: towards the cross-layer design of approximate dnns," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference, in ASPDAC '19*. New York, NY, USA: Association for Computing Machinery, pp. 317–322.

[48] C. Parra *et al.*, "Full approximation of deep neural networks through efficient optimization," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS*, pp. 1–5.

[49] C. Parra *et al.*, "Knowledge distillation and gradient estimation for active error compensation in approximate neural networks," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 679–684.

[50] C. Parra *et al.*, "Efficient accuracy recovery in approximate neural networks by systematic error modelling," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 365–371. [Online]. Available: https://ieeexplore.ieee.org/document/9371529

[51] X. He *et al.*, "Joint design of training and hardware towards efficient and accuracy-scalable neural network inference," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 810–821, 2018.

[52] C. Parra *et al.*, "Proxsim: Gpu-based simulation framework for cross-layer approximate dnn optimization," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar*, pp. 1193–1198.

[53] F. Vaverka *et al.*, "Tfapprox: Towards a fast emulation of dnn approximate hardware accelerators on gpu," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 294–297.

[54] M. Kwak *et al.*, "Torchaxf: Enabling rapid simulation of approximate dnn models using gpu-based floating-point computing framework," in *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 1–8.

[55] D. Danopoulos *et al.*, "Transaxx: Efficient transformers with approximate computing."

[56] G. Zervakis *et al.*, "Approximate computing for ml: State-of-the-art, challenges and visions," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021, pp. 189–196.

[57] G. Armeniakos *et al.*, "Hardware approximate techniques for deep neural network accelerators: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, nov 2022. [Online]. Available: https://doi.org/10.1145/3527156

[58] V. Piuri, "Analysis of Fault Tolerance in Artificial Neural Networks," *Journal of Parallel and Distributed Computing*, vol. 61, no. 1, pp. 18–48, Jan. 2001.

[59] E. Mhamdi *et al.*, "When neurons fail," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1028–1037.

[60] C. Schorn *et al.*, "Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar*, pp. 979–984.

[61] T. Spyrou *et al.*, "Neuron fault tolerance in spiking neural networks," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 743–748.

[62] M. Ahmadilivani *et al.*, "Deepvigor: Vulnerability value ranges and factors for dnns' reliability assessment," in *2023 IEEE European Test Symposium (ETS)*, pp. 1–6.

[63] G. Li, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, in SC '17*, vol. 8. New York, NY, USA: ACM, pp. 1–8 12.

[64] F. F. d. Santos *et al.*, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2019.

[65] F. F. Dos Santos *et al.*, "Experimental evaluation of neutron-induced errors on a multicore risc-v platform," in *2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2022, pp. 1–7.

[66] R. L. Rech *et al.*, "Reliability of google's tensor processing units for embedded applications," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 376–381.

[67] S. Pappalardo *et al.*, "A fault injection framework for ai hardware accelerators," in *2023 IEEE 24th Latin American Test Symposium (LATS)*, 2023, pp. 1–6.

[68] M. Traiola *et al.*, "Impact of high-level-synthesis on reliability of artificial neural network hardware accelerators," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2024.

[69] J. Condia *et al.*, "A multi-level approach to evaluate the impact of gpu permanent faults on cnn's reliability," in *2022 IEEE International Test Conference (ITC)*, pp. 278–287.

[70] F. F. Dos Santos *et al.*, "Understanding and improving gpus' reliability combining beam experiments with fault simulation," in *2023 IEEE International Test Conference (ITC)*, 2023, pp. 176–185.

[71] C. Bolchini *et al.*, "Fast and Accurate Error Simulation for CNNs Against Soft Errors," *IEEE Transactions on Computers*, vol. 72, no. 4, pp. 984–997, Apr. 2023.

[72] L. Roquet *et al.*, "Cross-Layer Reliability Evaluation and Efficient Hardening of Large Vision Transformers Models," in *Design Automation Conference (DAC)*, San Fracisco, United States, Jun. 2024. [Online]. Available: https://hal.science/hal-04456702

[73] S. K. S. Hari *et al.*, "Sassifi: An architecture-level fault injection tool for gpu application resilience evaluation," in *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2017, pp. 249–258.

[74] A. Bosio *et al.*, "A reliability analysis of a deep neural network," in *2019 IEEE Latin American Test Symposium (LATS), Mar*, pp. 1–6.

[75] Z. Chen *et al.*, "Tensorfi: A flexible fault injection framework for tensorflow applications," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 426–435.

[76] M. Beyer *et al.*, "Fault injectors for tensorflow: Evaluation of the impact of random hardware faults on deep cnns," 2020.

[77] A. Mahmoud *et al.*, "Pytorchfi: A runtime perturbation tool for dnns," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2020, pp. 25–31.

[78] B. Goldstein *et al.*, "Reliability evaluation of compressed deep learning models," in *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*, 2020.

[79] N. Khoshavi *et al.*, "Fiji-fin: A fault injection framework on quantized neural network inference accelerator," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec*, pp. 1139–1144.

[80] Z. Chen *et al.*, "Binfi: an efficient fault injector for safety-critical machine learning systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356177

[81] A. Colucci *et al.*, "enpheeph: A fault injection framework for spiking and compressed deep neural networks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 5155–5162.

[82] G. Gavarini *et al.*, "Sci-fi: a smart, accurate and unintrusive fault-injector for deep neural networks," in *2023 IEEE European Test Symposium (ETS)*. Venezia, Italy: IEEE, pp. 1–6.

[83] R. Leveugle *et al.*, "Statistical fault injection: Quantified error and confidence," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, 2009, pp. 502–506.

[84] N. Khoshavi *et al.*, "Shieldenn: Online accelerated framework for fault-tolerant deep neural network architectures," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6.

[85] A. Ruospo *et al.*, "Selective hardening of critical neurons in deep neural networks," in *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, 2022, pp. 136–141.

[86] M. Qin *et al.*, "Robustness of neural networks against storage media errors," 2017.

[87] H. Guan *et al.*, *In-place zero-space memory protection for CNN*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[88] S. Burel *et al.*, "Zero-overhead protection for cnn weights," in *2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2021, pp. 1–6.

[89] S.-S. Lee *et al.*, "Value-aware parity insertion ecc for fault-tolerant deep neural network," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 724–729.

[90] M. Traiola *et al.*, "A machine-learning-guided framework for fault-tolerant dnns," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–2.

[91] M. Traiola *et al.*, "hardnning: a machine-learning-based framework for fault tolerance assessment and protection of dnns," in *2023 IEEE European Test Symposium (ETS)*, pp. 1–6.

[92] A. Ruospo *et al.*, "On the reliability assessment of artificial neural networks running on ai-oriented mpsocs," *Appl. Sci*, vol. 11, no. 14, Art. no. 14.

[93] F. Ponzina *et al.*, "E2cnns: Ensembles of convolutional neural networks to improve robustness against memory errors in edge-computing devices," *IEEE Trans. Comput*, vol. 70, no. 8, pp. 1199–1212,.

[94] N. I. Deligiannis *et al.*, "Towards the integration of reliability and security mechanisms to enhance the fault resilience of neural networks," *IEEE Access*, vol. 9, pp. 155 998–156 012, 2021.

[95] C. Schorn *et al.*, "An efficient bit-flip resilience optimization method for deep neural networks," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Mar*, pp. 1507–1512.

[96] J. J. Zhang *et al.*, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, 2018, pp. 1–6.

[97] U. Zahid *et al.*, "Fat: Training neural networks for reliable inference under hardware faults," in *2020 IEEE International Test Conference (ITC)*, 2020, pp. 1–10.

[98] S. T. Ahmed *et al.*, "Nn-ecc: Embedding error correction codes in neural network weight memories using multi-task learning," in *2024 IEEE 42nd VLSI Test Symposium (VTS), IEEE, Apr*.

[99] S. Mittal, "A survey on modeling and improving reliability of dnn algorithms and accelerators," *J. Syst. Archit*, vol. 104, pp. 101 689,.

[100] A. Ruospo *et al.*, "A survey on deep learning resilience assessment methodologies," *Computer*, vol. 56, no. 2, pp. 57–66,.

[101] F. Su *et al.*, "Testability and dependability of ai hardware: Survey, trends, challenges, and perspectives," *IEEE Des. Test*, pp. 1–1,.

[102] C. Bolchini *et al.*, "Resilience of Deep Learning applications: a systematic survey of analysis and hardening techniques," Sep. 2023, arXiv:2309.16733 [cs]. [Online]. Available: http://arxiv.org/abs/2309.16733

[103] H.-G. Stratigopoulos *et al.*, "Testing and reliability of spiking neural networks: A review of the state-of-the-art," in *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2023, pp. 1–8.

[104] P. Rech, "Artificial neural networks for space and safety-critical applications: Reliability issues and potential solutions," *IEEE Transactions on Nuclear Science*, pp. 1–1, 2024.

[105] L. Luza, "Investigating the impact of radiation-induced soft errors on the reliability of approximate computing systems," in *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 1–6.

[106] F. Fernandes dos Santos *et al.*, "Reliability evaluation of mixed-precision architectures," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 238–249.

[107] A. Ruospo *et al.*, "Investigating data representation for efficient and reliable Convolutional Neural Networks," *Microprocessors and Microsystems*, vol. 86, p. 104318, Oct. 2021.

[108] A. Siddique *et al.*, "Exploring Fault-Energy Trade-offs in Approximate DNN Hardware Accelerators," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Apr. 2021, pp. 343–348.

[109] S. Pappalardo *et al.*, "Investigating the effect of approximate multipliers on the resilience of a systolic array dnn accelerator," in *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2023, pp. 1–6.

[110] G. Rodrigues *et al.*, "Survey on approximate computing and its intrinsic fault tolerance," *Electronics*, vol. 9, no. 4, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/4/557

[111] B. Deveautour *et al.*, "Qamr: an approximation-based fully reliable tmr alternative for area overhead reduction," in *2020 IEEE European Test Symposium (ETS)*, 2020, pp. 1–6.

[112] M. Traiola *et al.*, "Design space exploration of approximation-based quadruple modular redundancy circuits," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9.

[113] M. Taheri, "Deepaxe: A framework for exploration of approximation and reliability trade-offs in dnn accelerators," in *2023 24th International Symposium on Quality Electronic Design (ISQED), Apr*, pp. 1–8.

[114] M. Taheri *et al.*, "Appraiser: Dnn fault resilience analysis employing approximation errors," in *2023 26th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 124–127.

[115] M. H. Ahmadilivani *et al.*, "Special session: Approximation and fault resiliency of dnn accelerators," in *2023 IEEE 41st VLSI Test Symposium (VTS)*, 2023, pp. 1–10.

[116] S. Saeedi *et al.*, "Prediction of the impact of approximate computing on spiking neural networks via interval arithmetic," in *2022 IEEE 23rd Latin American Test Symposium (LATS)*, 2022, pp. 1–6.

[117] A. B. Gogebakan *et al.*, "Spikingjet: Enhancing fault injection for fully and convolutional spiking neural networks," 2024.

[118] W. Choi *et al.*, "Sensitivity based error resilient techniques for energy efficient deep neural network accelerators," in *2019 56th ACM/IEEE Design Automation Conference (DAC), Jun*, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8806833

[119] M. Rios *et al.*, "Error resilient in-memory computing architecture for cnn inference on the edge," in *Proceedings of the Great Lakes Symposium on VLSI 2022, in GLSVLSI '22*. New York, NY, USA: Association for Computing Machinery, pp. 249–254.

[120] C. Schorn *et al.*, "Automated design of error-resilient and hardware-efficient deep neural networks," *Neural Computing and Applications*, vol. 32, no. 24, pp. 18 327–18 345, Dec. 2020.

[121] M. Taheri *et al.*, "AdAM: Adaptive Fault-Tolerant Approximate Multiplier for Edge DNN Accelerators," in *2024 IEEE European Test Symposium (ETS)*. IEEE, May 2024.

[122] A. Siddique *et al.*, "Exposing reliability degradation and mitigation in approximate dnns under permanent faults," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 4, pp. 555–566, 2023.

[123] W. Wei *et al.*, "An Approximate Fault-Tolerance Design for a Convolutional Neural Network Accelerator," *IT Professional*, vol. 25, no. 4, pp. 85–90, Jul. 2023, conference Name: IT Professional.

[124] M. Beyer *et al.*, "Online Quantization Adaptation for Fault-Tolerant Neural Network Inference," in *Computer Safety, Reliability, and Security: 42nd International Conference, SAFECOMP 2023, Toulouse, France, September 20–22, 2023, Proceedings*. Berlin, Heidelberg: Springer-Verlag, Sep. 2023, pp. 243–256. [Online]. Available: https://doi.org/10.1007/978-3-031-40923-3_18