Resiliency approaches in Convolutional, Photonic, and Spiking Neural Networks

Resiliency Approaches in Convolutional, Photonic, and Spiking Neural Networks

A. Bosio[1], M. Gomes[1], F. Pavanello[4], A. Porsia[2], A. Ruospo[2], E. Sanchez[2], E. I. Vatajelu[3]
[1]*Ecole Centrale de Lyon, CPE Lyon, INL, Ecully, France*
[2]*Politecnico di Torino, Dip. di Automatica e Informatica, Torino, Italy*
[3]*Univ. Grenoble Alpes, CNRS, Grenoble INP*, TIMA, 38000 Grenoble, France*
[4]*Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, Grenoble INP, CROMA, 38000 Grenoble, France*
Email: [1]{name.surname}@ec-lyon.fr, [2]{name.surname}@polito.it, [3]{name.surname}@univ-grenoble-alpes.fr

***Abstract*—This study presents a comparative examination of state-of-the-art resiliency approaches of Convolutional, Spiking, and Photonic neural networks (CNNs, SNNs, PNNs), their fault and error models, and the main fault tolerance techniques.**

***Index Terms*—Convolutional Neural Networks, Spiking Neural Networks, Photonic Neural Networks, Reliability, Fault Injection, Functional Safety, AI safety**

I. Introduction

Ensuring the dependability of today's systems presents major problems due to the integration of multiple hardware technologies and the implementation of more complex algorithms. In fact, more systems are using artificial intelligence (AI)-based algorithms in conjunction with the smaller technology nodes to meet their ever expanding processing needs. In this context, it's important to highlight the growing trend of several industries, including automotive, robotics, avionics, etc., incorporating Artificial Neural Networks (ANNs) into their systems. Given their close to human computational powers, this turns out to be a highly interesting solution, but it's crucial to emphasize that using such solutions brings up new and complicated reliability challenges. Actually, delegating important decisions to ANN-based systems can be risky because, as predictive models, they are not flawless and may give an incorrect response even in fault/error-free situations. Therefore, it starts to be crucial to deeply investigate the reliability and the behaviours of those AI-based systems in faulty scenarios.

In the last decades, different neural network architectures have been designed to address specific types of problems and data structures more effectively. One of the key properties that influence the choice of the architectural implementations consists in the data characteristics. Architectures like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Graph Neural Networks (GNNs) are tailored to handle spatial relationships in images, sequential patterns in time-series data, and irregular structures in graphs effectively. Transformer architectures excel at parallel processing of sequences but may be less efficient for spatial data. Some architectures are specifically designed for specialized tasks, such as Generative Adversarial Networks (GANs) for generating synthetic data. Some other architectures are very well suited to process time-series data or sensory processing like the Spiking Neural Networks (SNNs).

*Institut National Polytechnique Grenoble Alpes

However, besides the task complexity, important aspects to consider are the power, energy, and computational efficiency, especially when they operate in memory and resource constrained devices, like the **TinyML** and IoT world. In areas like image recognition and speech processing, the energy and computational requirements during both training and deployment of Deep Neural Networks (DNNs) are growing at an unsustainable rate in the push for greater accuracy [1]. It is even more common to map neurons and weight values to a smaller value range using the *quantization* process [2], [3], e.g., from FP32 to INT8. The effect will be increased energy efficiency and performance.

Some innovative approaches have been suggested as an alternative to the well-established Deep Neural Networks (DNNs) and their architecture optimization. These new solutions focus on either bio-inspired computing, such as Spiking Neural Networks (SNNs), more energy-efficient architectures like in-memory computing on non-volatile memory cores, or even the adoption of entirely different materials, such as photonics.

Spiking Neural Networks (SNNs) have functional similarities to biological neural networks, allowing them to utilize sparsity and temporal coding. Although SNNs are currently not as performant as DNNs, the gap is narrowing in certain tasks, while SNNs generally require much less energy for operation [4]. In addition, by taking advantage of emerging memory devices, such as memristors, their energy efficiency can be pushed even further. However, training SNNs remains a challenge due to the complex dynamics of neurons and the non-differentiable nature of spike operations. In this context, the bio-inspired training, such as spike-timing-dependent plasticity (STDP) can be seen as a viable solution, even more so since this approach can help mitigate hardware imperfections, increase energy efficiency and allow for network versaltility over different tasks (reconfiguration by retraining).

Given this context, exploring the use of alternative hardware might enable further improvements in AI systems, by moving away from traditional electronic implementations. These emerging technologies are often not subject to the same limitations of their electronic counterparts, and thus, might offer more efficient alternatives for certain applications [5]. Photonic Neural Networks (PNNs) are hardware implementations of AI that perform computations on optical signals, rather than on electronic ones. By using light, they leverage several of its properties to potentially enable high parallelization, low latency, and reduced power consumption [6]. For instance, PNNs have been demonstrated to perform sub-

nanosecond image classification [7] and to achieve up to 10^{12} multiply and accumulate (MAC) operations per second [8].

The variety of neural network architectures and hardware implementations mirrors the complexity of the problems they seek to solve, underscoring the necessity for customized solutions to proficiently tackle the complexities present in real-world data and tasks. This diversity has led to the development of various resiliency approaches, as detailed in the literature.

The main intent of this work is to highlight the state-of-the-art solutions to assess and increase the reliability of Convolutional (Section II), Spiking (Section III), and Photonic (Section IV) neural networks.

II. Resiliency Approaches in CNNs

This section provides an overview of the main resilience assessment approaches for CNNs (Section II-A), then it presents image-based fault detection techniques (Section II-B). Importance is placed on the experimental demonstration that a test image can not only excite hardware-induced faults, but also propagate them through the output of the CNN.

A. Reliability Assessments

CNNs' resiliency investigations and improvements can be conducted at various levels of detail or granularity:

- *Network-level resilience (NLR):* the evaluation of the resiliency is performed considering the entire neural network, and the improvement is achieved by means of network-tailored or full redundancy solutions, e.g., the entire CNN is duplicated or tripled. Examples are [9], [10]. Clearly, the overhead associated to this solution is non-negligible; to address this issue, the following classes offer more selective approaches.
- *Layer-level resilience (LLR):* the resilience of individual layers is investigated [9]–[11] and selective mitigation techniques are proposed to trade-off between reliability, execution time, and time and/or resource redundancy [12]–[14].
- *Feature-map-level resilience (FLR):* feature maps in CNNs exhibit different vulnerability; this class identifies and statically protects the most vulnerable feature maps in a CNN. For instance, the authors in [15] propose to duplicate only filters corresponding to vulnerable feature maps. This duplication decision is made before the model is deployed.
- *Inference-level resilience (ILR):* it selectively reruns vulnerable inferences by analysing their output. It is a time redundancy approach and is invoked based on inference output (e.g., [15]).
- *Kernel-level resilience (KLR):* the resilience of individual parts (i.e., kernels) of the source code executed on a GPU is investigated. Based on the different kernel vulnerability factor (KVF), i.e., the probability of faults in a kernel to affect the model's computations, specific vulnerable kernels are hardened. As an example, the authors in [16] propose a Triple Modular Redundant (TMR) technique to tune the trade-off between the model's reliability and its performance through hardening the carefully-selected kernels.
- *Weights-level resilience (WLR):* the resilience of individual weights is investigated to assess the different weights' fault tolerance, depending on the layer's position, the data type representation (e.g., [10], [11]).
- *Neurons or activation-level resilience (ActLR):* the vulnerability of each individual artificial neuron is investigated (e.g., [10], [17]–[19]).
- *Bit-level resilience (BLR):* the resilience of individual bit positions is assessed. This granularity allows to better study the vulnerabilities of selected data type representations (e.g., floating-point weights vs. integer ones).

Even though this classification is tailored to CNNs, it can be applied to other types of architectures, integrating or adapting specific internal units that are not considered in CNNs. Depending on the above classification, several fault mitigation approaches have been proposed to raise the resiliency level of state-of-the-art CNNs.

The most used faults and error models in CNNs can be grouped in two big categories, that involve the corruption of (i) parameters (e.g., synaptic weights) or (ii) artificial neurons, also known as activation values. A fault in a parameter (synaptic weights, biases) can be implemented, for example, as a stuck-at-*value*, where the value is within the full domain of the adopted data representation. This fault model is reproduced as the effect of a single faulty bit within the weights' binary range. Indeed, one of the most used fault models consists of stuck-at-0 or stuck-at-1 affecting individual bits in CNN's weights. They are typically implemented as bit flips, and, the resulting faulty value will be within the entire range available for the specific data representation. In the last few years, a great effort has been made to investigate the sensitivity of each bit position in networks using different data types. This is due to the fact that the values represented by a data type depend on the bit positions involved, as different data types interpret each bit differently [11], [20], [21].

Errors in activation values (neurons) are mainly implemented as *crashed/dead* neurons and *byzantine* neurons. The first class includes all those faults that completely stop their activity. A crashed neuron is modelled by purposely setting its output to zero. The second class includes neurons that keep their activity but produce arbitrary values, within their bounded transmission capacity.

B. Fault Detection Techniques

To meet today's functional safety standards, deploying AI algorithms in safety-critical systems requires detecting in the field the occurrence of permanent and transient faults happening during the operational phase of devices. To avoid integrating additional hardware, software-based solutions are the more appealing option among the possibilities that exist.

Current state-of-the-art functional methods for detecting faults that may affect AI-powered devices are often based on Software Test Libraries (STLs), a set of assembly programs that can excite and detect faults in the underlying computing device by applying carefully crafted test patterns. However, since STLs require a significant manual effort to develop and may affect the performance of the AI task [22], alternative solutions have been proposed that leverage directly the computations of AI models, such as CNNs. In particular, they involve constructing suitable input data that, when fed to the CNN, can excite faults simply by flowing through the layers of the network and propagate them up to the output layer.

This image-based fault detection strategy relies on the adoption of carefully created test images.

Image-based fault detection: To the best of our knowledge, the first research that exploits images for testing purposes is presented in [23]. The authors propose a method to test a memristor-based Resistive Random Access Memory (ReRAM) hosting a CNN using test images. In particular, the authors leverage Adversarial Examples, a technique that consists in adding a small quantity of noise to an image so that it looks indistinguishable from the original to the human eye, but causes the CNN to output a completely wrong prediction. While this technique is originally used as an offensive method to disrupt the operations of a CNN, the authors use the fault sensitivity of Adversarial Examples to excite and detect ReRAM faults with high probability. Inspired by these testing property of images, in [24], a novel technique that also leverages test images to detect faults was presented: the *Image Test Libraries (ITLs)*. Similarly to STLs, ITLs deliver test patterns to a functional unit in order to detect faults, but the difference lies in how these patterns are conveyed to the functional unit of interest. While STLs use assembly instructions to control the inputs to the functional unit under test, ITLs exploit existing CNN routines that make use of the functional unit of interest. Since convolutions account for more than 90% of the operations of a CNN, convolution kernels are an optimal target for this technique. Test patterns generated using Automatic Test Pattern Generation (ATPG) are carefully placed into a set of images so that they may reach the functional unit of interest when the convolution is performed. The main advantage offered by ITLs is the possibility to leave the CNN running without modifying its structure and/or weights, which are in fact used as input constraints for the ATPG process. In particular, ITLs have been used to perform in-field functional testing of single-precision floating-point multipliers in a GPU by leveraging the weights of the first convolutional layer of a CNN. The process includes reconstructing a dataflow algorithm capable of mapping each GPU core to the list of multiplications it performs in the form of weight-input index pairs, depending on the convolution algorithm and the GPU scheduling policies. Once the mapping is known, it is possible to place test patterns in one or more images so that each multiplier may receive the correct ones. Fault simulation of the ITL reported a single stuck-at test coverage of around 95% for each multiplier of the targeted GPU.

ITL fault propagation analysis: ITLs have been shown to be able to propagate hardware faults up to the software level. To experimentally demonstrate the efficacy of ITLs in exciting faults, in [24] faulty images have been created. To this end, architectural-level fault simulations have been performed and then, for each injected fault, it has been checked if the fault was propagated to the software level, which in the case of a CNN corresponds to the *output of the first layer*. The fault injection technique employed for this purpose is a mixture of architectural and software-level fault injection that combines the accuracy of gate-level microarchitectural simulation with the speed of software fault injections. The main idea behind it is that a multiplication performed by a faulty multiplier, therefore producing a faulty output $\widehat{O}$, is equivalent to a multiplication performed by a golden multiplier with a faulty input. In mathematical terms, if a faulty multiplier produces a faulty output $\widehat{O} = I \cdot W$, the same $\widehat{O}$ can be produced by a golden multiplier that performs the same operation, but with a faulty input, i.e., $\widehat{I} \cdot W = I \cdot \widehat{W} = \widehat{O}$. Given $\widehat{O}$, I and W, it is possible to calculate $\widehat{I}$ and $\widehat{W}$ as:

$$\widehat{I} = \frac{\widehat{O}}{W} \qquad \widehat{W} = \frac{\widehat{O}}{I}$$

Fault injections are performed by applying to the CNN faulty inputs $\widehat{I}$ corresponding to specific hardware faults internal to the targeted multiplier. Costly simulations are replaced by the inference of faulty images corresponding to a precise hardware fault. The construction of faulty images consists of the following steps:

1) Inject a fault into a multiplier
2) For each multiplication $I \cdot W$ performed by the multiplier during the inference of the ITL images, extract the weight W and the faulty output $\widehat{O}$
3) Compute $\widehat{I} = \frac{\widehat{O}}{W}$
4) Replace the original input I with the faulty input $\widehat{I}$

Given the nature of the convolution operation, a single element of the input concurs in the computation of several elements of the output, including some that may not be computed by the targeted multiplier. It follows that launching the inference of a faulty image results in some output elements being wrong even if they should not be. To solve this problem, a mask M is applied to the output. M is a tensor having the same size of the output that contains a 1 in positions corresponding to output elements that are actually computed by the faulty multiplier, and 0 in all the others. Given a fault f, a multiplier c, a layer L, an input image I and the corresponding faulty image(s) I_f, the correct faulty output is calculated as:

$$L_f(I) = L(I) \odot (\mathbf{1} - M_c) + L(I_f) \odot M_c \tag{1}$$

This procedure has been shown to allow the observation of the fault propagation to the output of the *first* layer. However, a more realistic approach would involve trying to observe the fault on the output of the last layer. In this work, the procedure has been extended in order to apply the same hardware fault to *all* convolutional layers, with the only difference (compared to the first layer) that operations involving the input image must be performed using the output of the previous layer. Equation 1 can be generalized for each convolutional layer i as:

$$L_{i,f}(O_{i-1}) = L_i(O_{i-1}) \odot (\mathbf{1} - M_{i,c}) + L_i(O_{i-1,f}) \odot M_{i,c}$$

where O_{i-1} is the output feature map of the $i-1$-th layer and $O_{i-1,f}$ is the corresponding faulty feature map. By applying this procedure to each convolutional layer, the propagation of the fault can be observed up to the output layer. Figure 1 shows a graphical representation of this fault injection method. The input image I on the left is passed through the L_1 convolutional layer, where a fault has been injected on the multiplier MUL_0. The faulty multiplications performed during the convolution are then collected and the corresponding faulty inputs $\widehat{I}$ are computed. The mask M_{1,MUL_0}, relative to layer L_1 and multiplier MUL_0, contains a 1 in positions corresponding to outputs computed by the faulty multiplier, 0 everywhere else. The faulty inputs (represented in purple) are then applied to the input image, producing the faulty

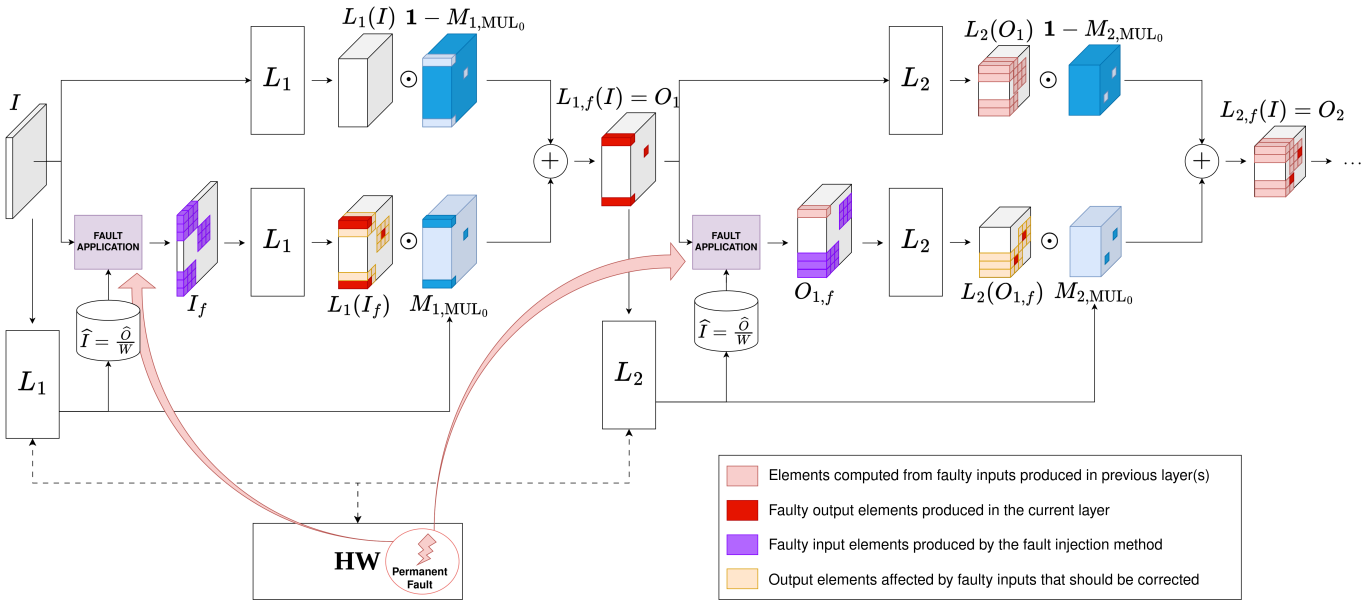

Fig. 1: Schematic of the software-level fault injection framework employed to experimentally show that ITLs can propagate the fault to the output of the CNN.

input image I_f. Both I and I_f are then passed to L_1 *without injecting the fault*, producing the clean output feature map $L_1(I)$ and the faulty output feature map $L_1(I_f)$, respectively. $L_1(I_f)$ contains the output elements computed by MUL_0 (in red), as well as some incorrect output elements (in light orange) computed as a byproduct of the fault injection method. In fact, in a convolution a single input concurs in the computation of more than one output elements, hence changing an input results in a change of several outputs. To remove the incorrect output elements, $L_1(I_f)$ is multiplied elementwise by the mask M_{1,MUL_0} to single out the faulty output elements and leave out the others. The clean output $L_1(I)$ is multiplied elementwise by $\mathbf{1} - M_{1,\mathrm{MUL}_0}$ to single out the output elements that were not computed by MUL_0. The two resulting tensors are then summed to obtain the faulty output feature map $L_{1,f}(I) = O_1$. The procedure is then repeated (second half of figure 1) by substituting I with O_1. Note that in the second iteration, the faulty output elements produced in L_1 concur in the computation of output elements (in light red) even when executing L_2 without injecting a fault on MUL_0. Given the nature of the convolution operation, wrong inputs produced in L_1 are processed also by clean multipliers in L_2, effectively propagating the fault on more output elements. These faulty outputs are then supplemented by new faulty outputs produced by MUL_0 when computing the output of layer L_2.

Preliminary results obtained with the above method indicate that a permanent stuck-at fault in a multiplier can be excited by the ITL in the first convolutional layer and propagated up to the CNN's output layer. For the sake of completeness, an additional experimental analysis has been performed. The six test images developed in [24] for the in-field testing of GPU's multiplier running the ResNet20 CNN have been adopted. The intent of the analysis was to experimentally verify the ability of the considered ITL in propagating faults up to the output of the CNN. To do so, the software-level framework shown in Figure 1 has been

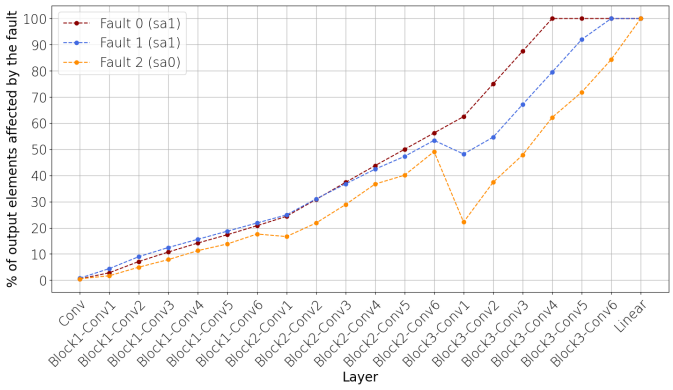

Fig. 2: Percentage of faulty output elements for each convolutional and fully connected layer of ResNet20.

developed with the intent of turning gate-level faults (i.e., stuck-at faults affecting the multiplier) into test images. Results for three hardware faults injected on a targeted multiplier (two stuck-at-1s, one stuck-at-0) are shown in Figure 2. The graph reports the percentage of output elements that change due to the faulty multiplier at every convolutional and fully connected layer of the CNN under study (ResNet20 targeting CIFAR-10). It can be observed that the percentage of different values almost always increases, leading to the output of the model being completely different from the expected one. For the sake of clarity, only Fault 0 leads to a failure. The drops at the beginning of each block correspond to strided convolutions, in which some elements of the input feature map are skipped and the size of the output is reduced. These results can be explained by the fact that in a convolution with a $f \times f$ filter, a change in a single input element affects $\left(\frac{f}{2}\right)^2$ to f^2 output elements, including some that may be computed by different, fault-free multipliers. Due to this

property of the convolution operation, output elements computed using clean multipliers may be wrong, not because of an internal hardware fault, but because one of the inputs is wrong. Besides, new faulty outputs are produced in each convolution by the same faulty multiplier, since it is assumed that the fault is permanent. When performing several convolutions one after the other, this behavior results in an avalanche effect that causes large chunks of the feature maps to be wrong with respect to the feature maps obtained by performing the same chain of convolutions with a clean input and a correctly functioning multiplier.

III. Resiliency Approaches in SNNs

The Spiking Neural Networks (SNN) are widely studied nowadays due to the high level of realism they bring to neural simulation, their energy efficiency and their ability for on-line learning. In SNNs, neurons communicate with each other through discrete-time spikes, rather than continuous-valued activations. The related bio- inspired learning rule is known as STDP (Spike Based Dependent Plasticity) and is applied on each synapse independently of the global state of the network. In return, the synapse must be doted of computation capabilities. A hardware implementation of an SNN requites architectural co-localization of the processing and memory (non-Von Neumann architecture). The circuits solutions used to implement silicon neurons are application depended, but the vast majority are built with a temporal integration block, a spike generation block, a refractory period mechanism, and a spike adaptation block [25]. Synapses are required to exhibit plasticity (i.e., modulation in their efficacy) and to support online learning algorithms, that manifest in changes in their strengths. Emerging memory devices can be used as synaptic elements thanks to their tunable conductivity, compatibility with advanced CMOS fabrication process, low power consumption, non-volatility and scalability. The synaptic conductance modulation can be emulated using: (i) the analog approach (cumulative decrease and increase of resistance), where multiple resistance states emulate long-term potentiation and depression; or (ii) the binary approach, uses two distinct resistance states per device associated with a probabilistic programming scheme [26].

In this context, we are concerned with the use of emerging memory technologies (memristors and spintronic devices) in a non-Von Neumann context and the dependability issues they face especially within hardware implementations of bio-inspired neural networks (Spiking Neural Networks).

The emerging memory technologies favor increasing system complexity and performance, opening the scientific community to great improvements in state-of-the-art computing but also to new applications and computation paradigms (such as in-memory computing or neuromorphic computing) which had been unfeasible a few years back due to technological limitations. This work mainly focusses on the use of memristors as artificial synapses for bio-inspired computing architectures. In this context, they have double functionality: memory (to save the values of the synaptic weights) and computation (to facilitate the on-line learning process, i.e., update synaptic weights) [27].

Due to the aggressive technology scaling and the introduction of new steps at back-end-of-line for the fabrication of the storage element, both components of the memory cell (access device and storage element) suffer from fabrication-induced variability. Moreover, due to intrinsic properties of these technologies, they are more susceptible to variations and defects, so there is a need for high quality test and fault tolerance. In addition, because of the different operation modes (analog storage and computing) compared to traditional digital memories, they require fundamentally new testing schemes.

The strong restrictions on the size of embedded Spiking Neural Network architectures (limited silicon area and interconnectivity ability) require minimization of the network redundancy which in turn reduces its the intrinsic fault tolerance, therefore there is an acute need to evaluate the reliability and perform manufacturing test of the neuromorphic hardware architectures to guarantee their correct operation and robustness. In this section we introduce a method for analyzing faults in Spiking Neural Networks (SNNs), examine the impact of hardware imprecision on the accuracy of the network, and assess how the training of the network influences its ability to withstand faults.

To analyse the faults that can occur in a Spiking Neural Network (SNN) and to asses the network robustness the development of pertinent fault models and methodologies for conducting fault injection campaigns are essential. In this context, we propose two analyse faults from two different angles, i.e., from bottom-up and from top-down. In the **bottom-up approach** the effect of fabrication-induced defects and variability on the operation of the neuron and synapse is to be evaluated and a fault modeling campaign should be conducted. The resulting faults can then be injected at system level and the robustness of the SNN evaluated. In previous work we showed that fabrication- induced parameter variations affect the neuron/synaptic behavior, which in turn affects the robustness of the SNN [28]. In this work, we complete our previous study by evaluating the accuracy loss of an SNN with leaky-integrate and fire (LIF) neurons and memristive synapses trained on the NMNIST data-set [29] by using different training approches: Shadow Training ST (with a base accuracy of 96%), (ii) BackPropagation Through Time BPTT (with a base accuracy of 98%), (iii) STDP (with a base accuracy of 94%). The results are illustrated in Fig. 3. As expected, the robustness of the network is influenced by the training procedure. In fact, when using shadow training, the network's fault tolerance is lower than in the other two cases, which is not entirely surprising since this type of training does not take into account any of the unique features of an SNN. On the other hand, when using STDP training, the fault tolerance is maximized, as the non-idealities are considered during the training process.

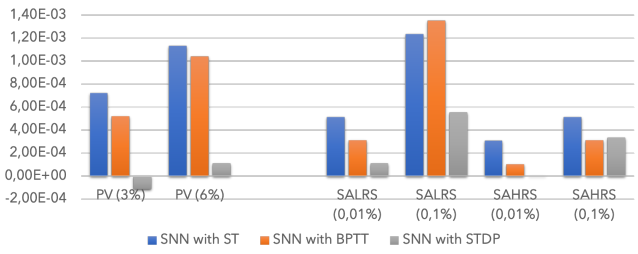

Fig. 3: Relative accuracy loss under synaptic nonideality (process variability-PV and defects leading to stuck-at faults-SA, i.e, memristor does not switch)

For a complete picture of SNN robustness, in further work we will analyze the impact of device degradation/malfunctioning and disturb effects on the performance of the neuron/synapse module and on the entire network. We will evaluate all pertinent reliability concerns like voltage noise, temperature noise, electromagnetic noise, aging degradation, cycle-to-cycle variation, etc. The identified possible (pertinent) faults will be injected at architecture level and mapped at behavioral level.

In the **top-down approach**, the correctness of the SNN algorithm needs to be evaluated under the effect of different faults, and the results to be validated by functional evaluation of the SNN architecture. Starting from the behavioral model of the SNN under study, we have investigated how the network behaves under different issues such as defective neurons or defective synapses. We have also evaluated the functional accuracy of the SNN during inference and learning. The results of this analysis provide insights into the following aspects of Neural Network (NN) functionality: the impact of defective neurons versus defective synapses on NN performance; the number of critical components that must fail for the entire network to fail; he significance of defect location, whether in the inner or outer layer of a NN. In a first approximation, a fault is defined as an undesired behaviour of a network element (neuron or synapse). The first fault models we have considered are dead neuron fault (DNF) and dead synapse fault (DNF) [29]. A DNF is defined as a neuron that does not fire under any conditions. A DSF is defined as a synaptic connection that does not allow information transfer from input to output neuron. We have perform a fault injection campaign, assuming different scenarios of fault occurrence assuming clustered faults or unclustered (randomly or deterministically distributed) faults. The functionality of the network has been evaluated and the severity of the fault occurrence related to its frequency and location is assessed. The DNFs and DSFs are the worst-case fault models, as they illustrate the situation where all functionality of (faulty) neuron and synapse is lost. We have also analysed the faults caused by the reduced functionality of neuron and/or synapse such as: modified synaptic weight, faulty signal integration by the neuron, modified firing threshold of the neuron, delayed firing of the neuron, etc. All these studies were conducted on an SNN with leaky-integrate and fire (LIF) neurons, trained on the NMNIST dataset [30] by using different training approches: Shadow Training ST (with a base accuracy of 96%), (ii) BackPropagation Through Time BPTT (with a base accuracy of 98%), (iii) STDP (with a base accuracy of 94%) and the results are illustrated in Fig. 4. It is worth noting that the impact of DSF on the network accuracy is very similar to the impact of memristor SA faults illustrated in 3. Moreover, the presence of DNFs has a more severe effect on the network trained by STDP compared to the network trained by backpropagation, particularly when the number of DNFs is large. This is primarily because in STDP-trained networks, each output neuron responds to a specific pattern due to the local training procedure, whereas in networks trained using backpropagation, the all neurons respond to averaged patterns due to global training.

It is clear that even with on-line training, the presence of faults reduces the resilience of the Spiking Neural Network (SNN), making it necessary to develop resilience approaches for SNNs. The goal of these approaches is to ensure that SNNs can continue

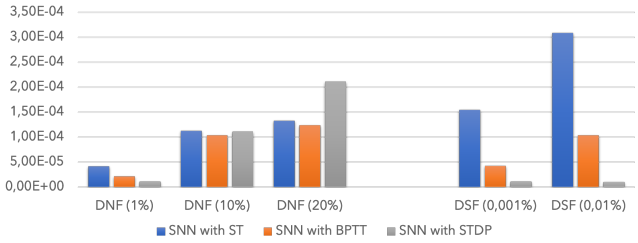

Fig. 4: Relative accuracy loss under Dead Neuron and Dead Synapse Faults (DNF, DSF)

to function correctly even in the presence of various types of perturbations, such as noise, neuron or synapse failure. Some examples of resilience approaches in spiking neural networks include but are not limited to:

- Noise Robustness: Introducing noise during training can help SNNs become more resilient to noise during inference. This can be achieved through techniques such as stochastic gradient descent, dropout, or adding noise to synaptic weights.
- Adaptation: Adjusting the network's parameters or connections in response to changes in the environment or input can help maintain its performance and stability.
- Adaptive Plasticity: Implementing adaptive plasticity mechanisms allows SNNs to dynamically adjust their synaptic weights and connectivity in response to changes in the environment or network conditions. This enables the network to adapt and compensate for variations or disturbances.
- Homeostasis: Maintaining the network's internal state within a stable range can help it resist perturbations and maintain its function.
- Continual Learning: Implementing continual learning strategies allows SNNs to continuously update their operation and adapt to new information without catastrophic forgetting. This ensures that the network remains resilient to changes in the input distribution or environment over time.

Implementing resilience approaches in Spiking Neural Networks (SNNs) will come at a cost, primarily in terms of reduced power efficiency, increased hardware complexity, and larger area footprint. To achieve a resilient SNN design that maintains its advantages over classical Deep Neural Networks (DNNs), it is important to carefully balance all of these aspects.

IV. Resiliency Approaches in PNNs

In this section, we give an overview of the resiliency of PNNs, highlighting how these hardware implementations of NN architectures demand the use of both traditional and novel approaches.

While the reliability of photonic devices in the context of data transmission is established [31], their resiliency within computing applications remains less explored, indicating possibilities for further research. Nonetheless, moving computations to the optical domain could address the needs of AI applications that demand low-latency responses and intensive computational power, such as in many safety-critical scenarios.

A. Photonic Circuits and Basic Devices

Photonic computing uses optical signals, rather than electrical ones, to carry out computations. This analog computing approach

involves modulating the amplitude and phase of light to represent data, often using its wavelength, polarization or mode to parallelize computations. Photonic circuits then process this data by modifying, splitting and combining different signals, ultimately detecting them to retrieve the processed information [32].

Despite design differences, certain components are commonly found in photonic computing. Some of these are used to guide and direct light within the circuit, such as waveguides and beam splitters, while others modify their proprieties, as phase shifters and modulators. Additionally, we find functional blocks made from basic components, including interferometers, that enable the interference of different signals, and ring resonators which may serve as bandpass filters, permitting the transmission of specific light wavelengths that resonate within it, while attenuating others.

The characteristics of these components are defined during fabrication. However, many of them can be electronically controlled, which allows for a dynamic reprogramming of circuits at the cost of additional control circuitry [33]. For example, beam splitters have their splitting ratio determined by the device dimensions and geometry, and phase shifters can be controlled by adjusting the index of refraction of a small portion of a waveguide by, e.g., using waveguide heaters or injecting free cariers [34]. The same controlling strategy allows for other adjustments, such as selecting the resonant wavelength of a ring resonator or modifying the interference between signals in interferometers.

B. PNN Implementations

PNNs, as their name suggests, are NN implementations that use photonic hardware to perform computations, which allows for improvements in speed and energy efficiency due to the unique characteristics of light [35]. In the optical domain, linear transformations are achievable using passive components [36], matrix-vector operations are performed in reduced computational time [37], and data is parallelized to be processed simultaneously [8].

While no single photonic component acts as an artificial neuron, a photonic circuit can be designed to perform the mathematical operations of a given NN. Several implementations of PNNs have been suggested and demonstrated experimentally [38]. Since optical signals have multiple degrees of freedom for encoding data and computing, circuits can vary significantly in how they process and detect information. They are broadly categorized by how different inputs are distinguished, whether through space, wavelength, or time domains [39].

PNNs that use spatial differentiation of inputs, for example, assign a separate physical input to each optical signal. In these networks, weight matrix multiplications are achieved by modifying the phase of different inputs and making them interfere, most notably, using meshes of interferometers [36], [40]. Activation functions, on the other hand, can be implemented by using any of the devices and circuits that exhibit optical non-linearity [41], [42]. Aside from feed-forward NNs, photonic circuits also implement CNNs [8], [43], SNNs [44], reservoir computing systems [45], among others.

However, just as electronic systems, photonic systems are susceptible to hardware faults that may lead to errors, and must also be designed with robustness in mind if they are to be used in safety-critical applications. Most PNN implementations rely on electronic devices to some extent, thus, when considering their reliability, one must pay attention to threats that may affect both the photonic and electronic circuitry of these systems.

C. Photonic Devices Threats

Photonic components and signals used in PNNs face several threats. These might impact the circuit itself, resulting in wrong operations being performed, or the optical signals that propagate within it, changing the data being processed. Both impact light's behavior, leading to deviations in the expected outputs.

Since photonic circuits often rely on the interference of light, they are highly sensitive to changes in the dimensions of fabricated devices and the index of refraction of their materials [46]. Fabrication Process Variations (FPVs) represent a major concern in this context, as inconsistencies during the manufacturing process can modify the physical dimensions of devices [47], [48]. This leads to a range of issues, including unintended phase shifts, deviations on expected splitting ratios, or the shift of resonant wavelengths, all of which compromises the accuracy of PNNs. Beam splitters, for instance, were previously shown to have a $1 - 2\%$ deviation from 50:50 splitting ratio due to FPVs [49].

Temperature fluctuations, further impact photonic circuits by altering the index of refraction of their materials, leading to phase shifts and delays in the propagation of light. Temperature can also be an issue when using waveguide heaters as phase shifters, due to thermal crosstalk, i.e., when a heater affects unintentionally the behavior of neighbouring components [50]. Additionally, aging in photonic circuits must also be taken into account, although its impact is arguably lower compared to electronic systems, due to the physical nature of the photonic hardware [51]. Interestingly, unlike in electronics, optical devices and circuits have been shown to be resistant to threats resulting from radiation [52].

As signals propagate within the circuit, they experience losses and noise that might originate from absorption in different devices and crosstalk from tightly packed signals, among other sources [53]. The analog nature of computations in PNNs makes them particularly susceptible to these degradations in signal integrity, that at a first glance may be small at the device level, but accumulate throughout the circuit, leading to wrong interpretations of data at the detection stage, affecting the overall performance of PNNs.

D. Electronic Devices Threats

Despite the nature of PNNs, they still rely on electronic components for certain functionalities, such as storing information, controlling optical devices, and generating and detecting optical signals. These devices lie at the periphery of the photonic circuits, yet they pose vulnerabilities for the whole system, since they are also susceptible to threats. These threats are common to digital computers and have been extensively researched [54].

As an example, consider the act of controlling a phase shifter, needed to implement the trained weights into PNNs. This process starts with the retrieval of phase settings, i.e. the parameters of the network, from a digital memory. These settings are then translated into voltage levels by a digital-to-analog converter (DAC), which in turn adjusts the phase shifter according to the desired configuration. Problems might arise in many stages of this process, ranging from data alteration in the memory to deviations in the DAC's output, or even a lack of bit precision to implement the required phase shift, leading to incorrectly applied phase shifts.

Additionally, failures in these electronic components might result in a loss of control over the photonic elements.

E. Faulty Behavior Models

Developing and utilizing fault models for both photonic and electronic components within PNNs is essential for mitigating and understanding the impacts of faulty behaviors.

In the literature, several studies model faults in PNNs as uncertainties in the phase angles and splitting ratios of the photonic circuit. These uncertainties, that are sampled from Gaussian distributions, have also been explored in terms of their spatial correlation, sometimes representing maps of FPVs in integrated platforms [55]–[58]. Faults during detection have also been modeled as Gaussian distribution perturbations on the outputs [59]. Fault models that start from the transfer matrix abstraction of devices and scale up to system-level are also used to provide a more high-level description for specific FPVs [60], or to model crosstalk noise and losses [61], [62]. The use of low precision encoding in DACs has also been explored in low-power applications [55], [58].

Usually, the methodology for assessing the impacts of faults in PNNs is straight-forward. A PNN is trained on a benchmarking dataset, and its inference accuracy is assessed using a test dataset on a faulty version of the circuit (i.e., through fault injection). This process then is repeated to gather statistical data with respect to the NN accuracy. When focusing on the operation layers, rather than the full network, we see the use of distance metrics, such as the relative variation distance or the fidelity [6], [55], to quantify the deviation from the intended operation.

PNNs have degraded accuracy as uncertainties in phase shifts and splitting ratios increase. Accuracy tends to be more impacted if these uncertainties are uncorrelated or if they act on components that correspond to the initial layers of the network. Passive photonic circuits were shown to be more robust to faults than active ones, pointing to how optimization strategies that push for the use of less active devices [63] or reduced power consumption [64] might lead to more robust systems. Similarly, the thermal imbalance in these circuits can be mitigated during training [50], leading to new device designs which can be conceived to avoid configurations that might be hard to obtain due to FPVs [56].

V. Conclusion and future directions

The very common use of Artificial Neural Networks in almost all type of applications, including safety ones, mandates for more robust and mature reliability assessment processes. However, the size and complexity of modern ANN generates very challenging aspects when proposing reliability assessment strategies for such applications. In this paper, an overview of the state-of-the-art proposals for the reliability of CNN, SNN and PNN has been presented.

Regarding CNNs, some of the most relevant reliability assessment were highlighted, with particular attention to fault detection techniques, including new strategies for fault propagation analysis that may help to reduce computational fault injection time, while increasing accuracy.

The field of Spiking Neural Networks (SNNs) remains a vast and largely unexplored area with numerous opportunities for further research. Scaling up SNN architectures to handle larger and more complex datasets without compromising performance or efficiency is an urgent challenge that requires investigation. Additionally, the effective utilization of distributed computing and parallelization techniques for scaling SNNs is an open research question. The development of learning and plasticity mechanisms for training SNNs, particularly in scenarios with limited labeled data or in continual learning settings, is another critical area of research. Furthermore, energy efficiency, throughput, and real-time processing constraints must be addressed in SNN implementations on hardware platforms, such as neuromorphic chips or specialized accelerators. As new developments in SNNs continue to emerge, it is crucial to prioritize the resilience of SNNs as a primary feature rather than an afterthought, as it is often the case today.

The resiliency of PNNs is tied to the resiliency of the electronic components that control and interact with photonic circuits. However, this has not been thoroughly addressed by the literature, since faults in the electronic circuitry are mostly modeled as uncertainties in applied phase shifts or in the detection process. Addressing these gaps might involve running fault injection campaigns on the electronic devices, observing the impact on the PNN at a system level. Moreover, we notice a focus on interferometric mesh architectures, leaving room for the characterization of the reliability of other implementations and architectures.

Acknowledgment

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them. This work has also been supported by the Agnece National de la Recherche - France within the EMINENT Project ANR-19-CE24-0001

[1] S. Davidson and S. B. Furber, "Comparison of artificial and spiking neural networks on digital hardware," *Frontiers in Neuroscience*, vol. 15, 2021.

[2] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, "A comprehensive survey on model quantization for deep neural networks in image classification," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 6, nov 2023. [Online]. Available: https://doi.org/10.1145/3623402

[3] V. Rajagopal, C. K. Ramasamy, A. Vishnoi, R. N. Gadde, N. R. Miniskar, and S. K. Pasupuleti, "Accurate and efficient fixed point inference for deep neural networks," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 1847–1851.

[4] B. D. L. N. Yamazaki K, Vo-Ho VK, "Spiking neural networks and their applications: A review," *Brain Sci.*, vol. 12, no. 7, 2022.

[5] M. M. Waldrop, "The chips are down for moore's law," *Nature News*, vol. 530, no. 7589, p. 144, 2016.

[6] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature photonics*, vol. 11, no. 7, pp. 441–446, 2017.

[7] F. Ashtiani, A. J. Geers, and F. Aflatouni, "An on-chip photonic deep neural network for image classification," *Nature*, vol. 606, no. 7914, pp. 501–506, 2022.

[8] X. Xu, M. Tan, B. Corcoran, J. Wu, A. Boes, T. G. Nguyen, S. T. Chu, B. E. Little, D. G. Hicks, R. Morandotti *et al.*, "11 tops photonic convolutional accelerator for optical neural networks," *Nature*, vol. 589, no. 7840, pp. 44–51, 2021.

[9] A. Ruospo, G. Gavarini, C. de Sio, J. Guerrero, L. Sterpone, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, "Assessing convolutional neural networks reliability through statistical fault injections," in *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2023, pp. 1–6.

[10] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: a framework for quantifying the resilience of deep neural networks," in *Proceedings of the 55th Annual Design Automation Conference*, ser. DAC '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3195970.3195997

[11] A. Ruospo, E. Sanchez, M. Traiola, I. O'Connor, and A. Bosio, "Investigating data representation for efficient and reliable convolutional neural networks," *Microprocessors and Microsystems*, vol. 86, p. 104318, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933121004786

[12] Y.-J. Jung, S.-H. Han, and H.-J. Choi, "Explaining cnn and rnn using selective layer-wise relevance propagation," *IEEE Access*, vol. 9, pp. 18 670–18 681, 2021.

[13] C. Bolchini, L. Cassano, A. Miele, and A. Nazzari, "Selective hardening of cnns based on layer vulnerability estimation," in *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2022, pp. 1–6.

[14] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[15] A. Mahmoud, S. K. Sastry Hari, C. W. Fletcher, S. V. Adve, C. Sakr, N. Shanbhag, P. Molchanov, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Optimizing selective protection for CNN resilience," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 127–138.

[16] Y. Ibrahim, H. Wang, M. Bai, Z. Liu, J. Wang, Z. Yang, and Z. Chen, "Soft error resilience of deep residual networks for object recognition," *IEEE Access*, vol. 8, pp. 19 490–19 503, 2020.

[17] A. Ruospo, G. Gavarini, I. Bragaglia, M. Traiola, A. Bosio, and E. Sanchez, "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.

[18] C. Schorn, A. Guntoro, and G. Ascheid, "Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 979–984.

[19] L. Liu and J. Deng, "Dynamic deep neural networks: optimizing accuracy-efficiency trade-offs by selective execution," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[20] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *SC17: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[21] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[22] A. Ruospo, D. Piumatti, A. Floridia, and E. Sanchez, "A suitability analysis of software based testing strategies for the on-line testing of artificial neural networks applications in embedded devices," in *2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2021, pp. 1–6.

[23] W. Li, Y. Wang, H. Li, and X. Li, "Rramedy: Protecting reram-based neural network from permanent and soft faults during its lifetime," in *2019 IEEE 37th International Conference on Computer Design (ICCD)*, 2019, pp. 91–99.

[24] A. Ruospo, G. Gavarini, A. Porsia, M. S. Reorda, E. Sanchez, R. Mariani, J. Aribido, and J. Athavale, "Image test libraries for the on-line self-test of functional units in gpus running cnns," in *2023 IEEE European Test Symposium (ETS)*, 2023, pp. 1–6.

[25] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SAÏGHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, 2011. [Online]. Available: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2011.00073

[26] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," pp. 4.4.1–4.4.4, 2011.

[27] A. Basu, J. Acharya, T. Karnik, H. Liu, H. Li, J.-S. Seo, and C. Song, "Low-power, adaptive neuromorphic systems: Recent progress and future directions," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 6–27, 2018.

[28] E. I. Vatajelu and L. Anghel, "Fully-connected single-layer stt-mtj-based spiking neural network under process variability," in *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, 2017, pp. 21–26.

[29] E.-I. Vatajelu, G. Di Natale, and L. Anghel, "Special session: Reliability of hardware-implemented spiking neural networks (snn)," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019, pp. 1–8.

[30] G. K. C. N. T. Garrick Orchard, Ajinkya Jayawant, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Front. Neurosci, Sec. Neuromorphic Engineering*, vol. 9, 2015.

[31] M. Baharloo, M. Abdollahi, and A. Baniasadi, "System-level reliability assessment of optical network on chip," *Microprocessors and Microsystems*, vol. 99, p. 104843, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933123000893

[32] H. Zhou, J. Dong, J. Cheng, W. Dong, C. Huang, Y. Shen, Q. Zhang, M. Gu, C. Qian, H. Chen *et al.*, "Photonic matrix multiplication lights up photonic accelerator and beyond," *Light: Science & Applications*, vol. 11, no. 1, p. 30, 2022.

[33] W. Bogaerts, D. Pérez, J. Capmany, D. A. Miller, J. Poon, D. Englund, F. Morichetti, and A. Melloni, "Programmable photonic circuits," *Nature*, vol. 586, no. 7828, pp. 207–216, 2020.

[34] N. C. Harris, Y. Ma, J. Mower, T. Baehr-Jones, D. Englund, M. Hochberg, and C. Galland, "Efficient, compact and low loss thermo-optic phase shifter in silicon," *Optics express*, vol. 22, no. 9, pp. 10 487–10 493, 2014.

[35] B. J. Shastri, A. N. Tait, T. Ferreira de Lima, W. H. Pernice, H. Bhaskaran, C. D. Wright, and P. R. Prucnal, "Photonics for artificial intelligence and neuromorphic computing," *Nature Photonics*, vol. 15, no. 2, pp. 102–114, 2021.

[36] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Physical review letters*, vol. 73, no. 1, p. 58, 1994.

[37] Q. Cheng, J. Kwon, M. Glick, M. Bahadori, L. P. Carloni, and K. Bergman, "Silicon photonics codesign for deep learning," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1261–1282, 2020.

[38] L. De Marinis, M. Cococcioni, P. Castoldi, and N. Andriolli, "Photonic neural networks: A survey," *IEEE Access*, vol. 7, pp. 175 827–175 841, 2019.

[39] Y. Bai, X. Xu, M. Tan, Y. Sun, Y. Li, J. Wu, R. Morandotti, A. Mitchell, K. Xu, and D. J. Moss, "Photonic multiplexing techniques for neuromorphic computing," *Nanophotonics*, vol. 12, no. 5, pp. 795–817, 2023. [Online]. Available: https://doi.org/10.1515/nanoph-2022-0485

[40] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walsmley, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, p. 1460, 12 2016.

[41] I. A. D. Williamson, T. W. Hughes, M. Minkov, B. Bartlett, S. Pai, and S. Fan, "Reprogrammable electro-optic nonlinear activation functions for optical neural networks," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 1, pp. 1–12, 2020.

[42] A. Jha, C. Huang, and P. R. Prucnal, "Reconfigurable all-optical nonlinear activation functions for neuromorphic photonics," *Opt. Lett.*, vol. 45, no. 17, pp. 4819–4822, Sep 2020. [Online]. Available: https://opg.optica.org/ol/abstract.cfm?URI=ol-45-17-4819

[43] R. Cardoso, C. Zrounba, M. Abdalla, P. Jimenez, M. Gomes, B. Charbonnier, F. Pavanello, I. O'Connor, and S. Le Beux, "Photonic convolution engine based on phase-change materials and stochastic computing,"

in *2023 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2023, pp. 1–6.

[44] A. Jha, C. Huang, H.-T. Peng, B. Shastri, and P. R. Prucnal, "Photonic spiking neural networks and graphene-on-silicon spiking neurons," *Journal of Lightwave Technology*, vol. 40, no. 9, pp. 2901–2914, 2022.

[45] M. Abdalla, C. Zrounba, R. Cardoso, P. Jimenez, G. Ren, A. Boes, A. Mitchell, A. Bosio, I. O'Connor, and F. Pavanello, "Minimum complexity integrated photonic architecture for delay-based reservoir computing," *Opt. Express*, vol. 31, no. 7, pp. 11 610–11 623, Mar 2023. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?URI=oe-31-7-11610

[46] X. Chen, M. Mohamed, Z. Li, L. Shang, and A. R. Mickelson, "Process variation in silicon photonic devices," *Appl. Opt.*, vol. 52, no. 31, pp. 7638–7647, Nov 2013. [Online]. Available: https://opg.optica.org/ao/abstract.cfm?URI=ao-52-31-7638

[47] Z. Lu, J. Jhoja, J. Klein, X. Wang, A. Liu, J. Flueckiger, J. Pond, and L. Chrostowski, "Performance prediction for silicon photonics integrated circuits with layout-dependent correlated manufacturing variability," *Optics express*, vol. 25, no. 9, pp. 9712–9733, 2017.

[48] M. Nikdast, G. Nicolescu, J. Trajkovic, and O. Liboiron-Ladouceur, "Chip-scale silicon photonic interconnects: A formal study on fabrication non-uniformity," *Journal of Lightwave Technology*, vol. 34, no. 16, pp. 3682–3695, 2016.

[49] F. Flamini, N. Spagnolo, N. Viggianiello, A. Crespi, R. Osellame, and F. Sciarrino, "Benchmarking integrated linear-optical architectures for quantum information processing," *Scientific Reports*, vol. 7, no. 1, p. 15133, 2017.

[50] Y. Zhu, G. L. Zhang, B. Li, X. Yin, C. Zhuo, H. Gu, T.-Y. Ho, and U. Schlichtmann, "Countering variations and thermal effects for accurate optical neural networks," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–7.

[51] S. V. R. Chittamuru, I. G. Thakkar, and S. Pasricha, "Analyzing voltage bias and temperature induced aging effects in photonic interconnects for manycore computing." 2017.

[52] P. Dumon, R. Kappeler, D. Barros, I. McKenzie, D. Doyle, and R. Baets, "Measured radiation sensitivity of silica-on-silicon and silicon-on-insulator micro-photonic devices for potential space application," in *Photonics for Space Environments X*, vol. 5897. SPIE, 2005, pp. 119–128.

[53] Y. Xie, M. Nikdast, J. Xu, W. Zhang, Q. Li, X. Wu, Y. Ye, X. Wang, and W. Liu, "Crosstalk noise and bit error rate analysis for optical network-on-chip," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 657–660.

[54] D. Gizopolous, G. Di Natale, S. Di Carlo, A. Bosio, and R. Canal, *Cross-layer Reliability of Computing Systems*, ser. Materials, circuits and devices series. Institution of Engineering and Technology, 2020. [Online]. Available: https://books.google.fr/books?id=Ef45zgEACAAJ

[55] S. Banerjee, M. Nikdast, and K. Chakrabarty, "Characterizing coherent integrated photonic neural networks under imperfections," *Journal of Lightwave Technology*, vol. 41, no. 5, pp. 1464–1479, 2022.

[56] R. Hamerly, S. Bandyopadhyay, and D. Englund, "Asymptotically fault-tolerant programmable photonics," *Nature Communications*, vol. 13, no. 1, p. 6831, 2022.

[57] K. H. R. Mojaver, B. Zhao, E. Leung, S. M. R. Safaee, and O. Liboiron-Ladouceur, "Addressing the programming challenges of practical interferometric mesh based optical processors," *Optics Express*, vol. 31, no. 15, pp. 23 851–23 866, 2023.

[58] M. Y.-S. Fang, S. Manipatruni, C. Wierzynski, A. Khosrowshahi, and M. R. DeWeese, "Design of optical neural networks with component imprecisions," *Optics express*, vol. 27, no. 10, pp. 14 009–14 029, 2019.

[59] R. Cardoso, C. Zrounba, M. Abdalla, P. Jimenez, M. G. de Queiroz, B. Charbonnier, F. Pavanello, I. O'Connor, and S. Le Beux, "Towards a robust multiply-accumulate cell in photonics using phase-change materials," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–2.

[60] P. Agnihotri, P. Kalla, and S. Blair, "Transfer-matrix abstractions to analyze the effect of manufacturing variations in silicon photonic circuits," in *2022 IEEE International Test Conference India (ITC India)*. IEEE, 2022, pp. 1–8.

[61] A. Shafiee, S. Banerjee, K. Chakrabarty, S. Pasricha, and M. Nikdast, "Analysis of optical loss and crosstalk noise in mzi-based coherent photonic neural networks," *Journal of Lightwave Technology*, 2024.

[62] Y. Liu, J. Zhang, J. Feng, S. Chen, and J. Xu, "A reliability concern on photonic neural networks," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1059–1064.

[63] S. Yu and N. Park, "Heavy tails and pruning in programmable photonic circuits for universal unitaries," *Nature Communications*, vol. 14, no. 1, p. 1853, 2023.

[64] M. G. de Queiroz, R. Cardoso, P. Jimenez, M. Abdalla, I. O'Connor, A. Bosio, and F. Pavanello, "Power reduction in photonic meshes by mzi optimization," in *Frontiers in Optics*. Optica Publishing Group, 2023, pp. JW4A–7.