POLITECNICO DI TORINO
Repository ISTITUZIONALE

Special Session: Neuromorphic hardware design and reliability from traditional CMOS to emerging technologies

Original
Special Session: Neuromorphic hardware design and reliability from traditional CMOS to emerging technologies / Pavanello, F., Ioana Vatajelu, E., Bosio, A., Van Vaerenbergh, T., Bienstman, P., Charbonnier, B., Carpegna, A., DI CARLO, S., Savino, A.. - ELETTRONICO. - (2023), pp. 1-10. (41st IEEE VLSI Test Symposium San Diego CA (USA) 24-26 April 2023) [10.1109/vts56346.2023.10139932].

Availability:
This version is available at: 11583/2981394 since: 2023-08-30T10:48:35Z

Publisher:
IEEE

Published
DOI:10.1109/vts56346.2023.10139932

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

©2023 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

Special Session: Neuromorphic hardware design and reliability from traditional CMOS to emerging technologies

Fabio Pavanello[1], Elena Ioana Vatajelu[2], Alberto Bosio[1], Thomas Van Vaerenbergh[3], Peter Bienstman[4], Benoit Charbonnier[5], Alessio Carpegna[6], Stefano Di Carlo[6], Alessandro Savino[6]
[1]Univ. Lyon, Ecole Centrale de Lyon, INSA Lyon, Université Claude Bernard Lyon 1, CPE Lyon, CNRS, INL
[2]Univ. Grenoble Alpes, CNRS, Grenoble INP, TIMA, 38000 Grenoble, France
[3]Hewlett Packard Labs, HPE Belgium, B-1831 Diegem, Belgium, [4]Ghent University - imec, Gent, Belgium
[5]Univ. Grenoble Alpes, CEA, LETI, Grenoble, France
[6]Politecnico di Torino, Control and Computer Eng. Department, Torino, Italy

***Abstract*—The field of neuromorphic computing has been rapidly evolving in recent years, with an increasing focus on hardware design and reliability. This special session paper provides an overview of the recent developments in neuromorphic computing, focusing on hardware design and reliability. We first review the traditional CMOS-based approaches to neuromorphic hardware design and identify the challenges related to scalability, latency, and power consumption. We then investigate alternative approaches based on emerging technologies, specifically integrated photonics approaches within the NEUROPULS project. Finally, we examine the impact of device variability and aging on the reliability of neuromorphic hardware and present techniques for mitigating these effects. This review is intended to serve as a valuable resource for researchers and practitioners in neuromorphic computing.**

***Index Terms*—augmented silicon photonics, neuromorphic hardware, artificial neural networks, spiking neural networks, reliability, phase change materials**

I. Introduction

Artificial Neural Networks (ANNs) have enabled complex computations but require significant computational resources for both training and inference [1]. The main bottleneck in these networks is the transfer of large amounts of data to support different tasks. However, the trend in computing is moving towards edge devices, such as the Internet of Things (IoT), to improve security and reduce power consumption and latency [2]. Furthermore, there is a growing demand for powerful yet energy-efficient accelerators in various fields, including fault detection in microprocessors [3] and intrusion detection systems [4].

The design space for ANNs is vast, and it involves choices in four fundamental aspects: the neuron model, the architecture structure, the information encoding, and the training method [5]. These choices can significantly impact the hardware design and optimization process. Convolutional Neural Networks (CNNs), Deep Neural Networks (DNNs), and Spiking Neural Networks (SNNs) are three popular types of ANNs, each with unique characteristics and applications.

CNNs are commonly used in image recognition and processing tasks, and they rely on convolutional layers to extract spatial features from input images [6]. DNNs, on the other hand, are used in a wide range of applications, from speech recognition to natural language processing and game playing [7]. They are characterized by multiple layers of neurons that learn increasingly abstract features of the input data.

The limitations of traditional computing architectures, particularly the communication bottleneck between memory and processor and the latency of information propagation and manipulation, have highlighted the need for alternative approaches to ANNs. While software approaches for ANNs offer advantages when implemented on specialized hardware such as Graphical Processing Units (GPUs), these limitations persist [8].

SNNs have emerged as the next generation of ANNs that exchange information in spikes, inspired by the behavior of biological brains [9]. This allows for more efficient computation and reduced power consumption and is particularly interesting when working with time sequences such as audio, video, and electrical signals [10], [11]. The model complexity and internal parameters determine the model's suitability to the input data, with shorter time constants detecting shorter temporal correlations and higher values catching more prolonged time effects.

Eventually, the resilience of the ANN is crucial when designing the entire system and cannot be ignored [12], [13]. Retaking inspiration from biology, the human brain is intrinsically resilient to malfunctioning and faults. It loses approximately 50000 neurons daily but can still perform complex tasks and learn new ones, creating connections between the remaining neurons. ANNs have inherited this characteristic at

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101070238. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. It was also partially supported by the ANR within the EMINENT Project ANR-19-CE24-0001

979-8-3503-4630-5/23/$31.00 ©2023 IEEE

a certain level, but they still need to improve significantly, particularly in mission-critical and safety-critical applications. Therefore, a deeper study of ANNs from this perspective is required.

Neuromorphic computing, which merges memory and processing units within neurons and synapses and maps computing architectures more closely to Neural Networks (NNs) models, offers a promising solution to address the limitations of traditional computing architectures [14]. Various technologies, including CMOS, memristors, and optoelectronics/all-optical approaches, have been explored to develop neuromorphic hardware. This paper provides an overview of these approaches, highlighting their advantages and limitations. Background on ANNs is given in section II, while section III discusses the design and performance of the digital version of ANNs, with a focus on the SNNs. section IV discusses the potential of silicon photonics for ANNs, and section V covers reliability challenges in the usage of ANNs.

II. Background

The choice of the neuron model, architecture structure, information encoding, and training method can significantly impact the hardware design and optimization process for each of these ANNs. Therefore, it is essential to consider these aspects carefully when designing and implementing hardware for ANNs.

CNNs and DNNs use different neuron models. In CNNs, the neuron model is typically based on the Rectified Linear Unit (ReLU) function, a non-linear activation function commonly used in deep learning. The ReLU function is simple and computationally efficient, making it a popular choice for CNNs. The ReLU function is $f(x) = \max(0, x)$, where x is the input to the neuron. The output of the ReLU function is zero if the input is negative and equal to the input if the input is positive. The ReLU function effectively reduces overfitting in DNNs and has been used in various computer vision tasks, such as object detection and recognition [15].

In DNNs, the neuron model is typically based on a non-linear activation function. The sigmoid function, defined as $f(x) = \frac{1}{1+e^{-x}}$, where x is the input to the neuron, is one of the simplest. The output of the sigmoid function is between 0 and 1, which makes it useful for tasks such as classification. Another popular activation function in DNNs is the hyperbolic tangent ($\tanh$) function, similar to the sigmoid function but outputs values between -1 and 1. The choice of activation function depends on the task at hand and the NNs's architecture. For example, the sigmoid and $\tanh$ functions were commonly used in early DNN architectures, such as the Multilayer Perceptron (MLP) [16] but have since been largely replaced by the ReLU function in more recent architectures. However, the sigmoid and $\tanh$ functions are still used in certain NNs, such as Recurrent Neural Networks (RNNs) and autoencoders [17].

State-of-the-art implementations of CNNs and DNNs often use 32-bit floating-point numbers in software or model-based approaches. However, implementing such algorithms in hardware is challenging due to their extensive data requirements, high energy consumption, and large memory bandwidth. To address these challenges, quantization, and regularization techniques have been explored, using fixed-point computations with 16 bits, 8 bits, or lower precision. Although these methods reduce the precision of synaptic weights and inter-layer signals, IBM's TrueNorth [18] chip has achieved acceptable precision using only five synaptic states, albeit at high design and energy costs [19], [20].

In the case of SNNs, the neuron models and architectures are the most complex to target due to the nature of the information they carry and how they treat it. Many different mathematical models describe and mimic the behavior of biological neurons, such as the Hodgkin-Huxley model [21], the Izhikevich model [22], the Leaky Integrate and Fire (LIF) model [23], and the Integrate and Fire (IF) model [24]. These models range from very complex and detailed to much simpler and more suitable for machine learning and hardware applications, with varying degrees of biological plausibility and computational efficiency.

Neurons in SNNs are generally treated as leaky integrators where input spikes are integrated over time after being weighted by corresponding synapses, affecting the neuron's state, usually the electrical potential across its membrane. The neuron membrane depolarizes due to internal charge leakage without spikes, except in the IF model, where it is kept at a constant value. A new spike is generated when the membrane potential exceeds a specific threshold value, causing the neuron to fire, and the potential drops suddenly into a reset state.

The architecture of the SNN can also be very flexible. The literature reports Fully-Connected (FC) Feed-Forward (FF) SNN [25], regularly recurrent structures [10] or randomly recurrent architectures [26], used for example, in Reservoir Computing (RC). There can be only excitatory connections, with positive weights, or adding inhibitory connections, with negative weights [25], or in more detailed models, separated excitatory and inhibitory neurons, as observed in some regions of the human brain.

RC is an efficient technique where a randomly initialized RNN trains only a linear combination of the signals at each node. It has shown promising results in various applications, including photonics [27]. Different RC architectures have been investigated using photonics, such as spatial-multiplexing and time-multiplexing approaches. Spatial-multiplexing involves physically separating the nodes, while time multiplexing requires a faster sampling speed and more complex processing of the read-out layer.

The choice of input encoding can significantly affect the performance of the CNN or DNN. It often requires careful consideration and experimentation to determine the optimal encoding for the given task. Factors to consider when selecting an input encoding include the nature and complexity of the input data, the available computational resources, and the application's performance requirements.

The most common input encoding for CNNs is raw pixel

values, where each pixel in the image is represented as a numerical value. In contrast, for DNNs that process non-image data, input encoding may involve feature engineering techniques such as transforming the raw input data into a set of meaningful features more amenable to learning by the network [15]. For example, in natural language processing, input encoding may involve converting text into a numerical representation, such as bag-of-words or word embeddings.

Due to their nature, SNNs require more advanced methods for encoding and interpreting information. The main approaches are rate coding [28], temporal coding [29], and population rank coding [30]. Rate coding uses the average spike frequency to encode information and is suitable for static input data. It is less efficient regarding spike activity but more robust to noise. Temporal coding encodes information in the precise arrival time of spikes or their relative distance, requiring fewer spikes to process information, and is suitable for encoding time-varying signals. However, it is more sensitive to noise. Population rank coding uses the joint activity of a group of neurons to process information.

Training approaches involve optimizing the model parameters to minimize a given loss function for CNNs and DNNs. In supervised learning, this involves iteratively adjusting the weights and biases of the network to reduce the difference between the predicted output and the ground truth labels. The optimization is typically performed using gradient descent methods, which involve calculating the gradient of the loss function concerning the model parameters and updating them in the direction of the negative gradient. Commonly used gradient descent methods include Stochastic Gradient Descent (SGD) [31], AdaGrad [32], etc.

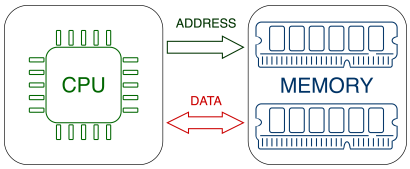
Regularization techniques such as Dropout [33], etc., are often used to prevent overfitting and improve generalization. Additionally, data augmentation methods such as flipping, rotating, and cropping the input images are employed to increase the size of the training dataset and improve model robustness [34].

Unsupervised learning approaches, such as Autoencoders [17] and Restricted Boltzmann Machines [35], can also be used for pretraining the model parameters. Transfer learning approaches can also be employed, where a pre-trained network is fine-tuned on a new dataset with similar or related features [36]. Finally, reinforcement learning can also be used to train CNNs and DNNs, where the model learns to take actions based on a reward signal, such as in game-playing agents [37].

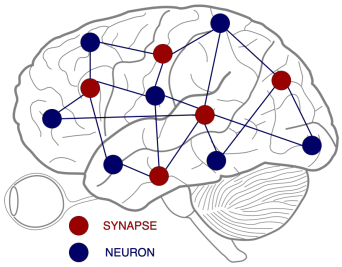
Training SNNs is challenging due to the non-differentiability of the thresholding function of neurons. Classical back-propagation methods cannot be directly applied, but several approaches have been developed, including supervised and unsupervised training. In supervised training, the most common approach is to convert an ANN into an SNN, where the ANN's differentiable non-linear function is trained using back-propagation, and the weights are used directly in the SNN. Alternatively, a Back-Propagation Through Time (BPTT) can be applied directly to the SNN, where a surrogate gradient replaces the neuron's thresholding function with a differentiable function during the backward pass. In contrast, inspired by biology, most unsupervised approaches update weights locally based on the relative spike timing between the inputs and the output without depending on a global error signal propagating across the network, resulting in a lighter memory footprint and computational overhead, with Spike Timing Dependent Plasticity (STDP) being the most common method [38].

III. Digital Accelerators

As seen before, the architecture of an ANN, and in the same way of an SNN, is generally composed of many independent neurons and, as such, is intrinsically strongly parallelizable. This poorly fits the common CPU-based computing approach, in which the parallelism is limited to a few tens of very powerful cores. For this reason, one active research branch in the field of ANN and SNN is directed towards accelerating such algorithms, using computing platforms to execute them more efficiently. The goal is to broaden their application to many contexts, such as performance-constrained, power-constrained, or real-time tasks.



(a) Classical Von Neuman computing approach



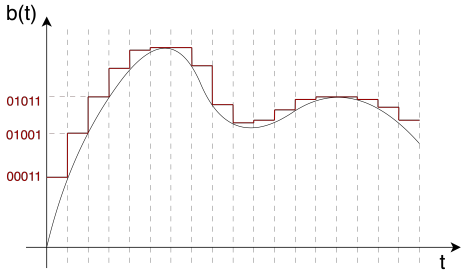
(b) Neural network's computation and memory co-location

When discussing acceleration in the digital domain, several solutions have different degrees of optimization, efficiency, and cost. One first approach, already mainstream for many ANN models, is to exploit the computational parallelism offered by general-purpose hardware accelerators like GPUs. Regarding SNNs, several software frameworks natively support the deployment of the code on GPUs, for example, the ones based on *pyTorch*, like *snnTorch* [39] and *spikeTorch*, or CUDA accelerated C++ frameworks, like *SLAYER* [40] and CARLsim 4 [41].

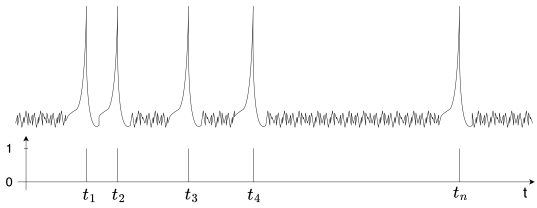
However, SNNs have many features unsuitable with a GPU execution. For example, spikes can be represented in the digital domain as single-bit events (high in the presence of a spike and

low otherwise). The numerical representation used in GPU, based on words with 8, 16, 32, 64, or similar bit widths, is inefficient. Moreover, the computation can be based on events: a neuron can react only in the presence of an active input spike, remaining in a quiescent state otherwise. Such an update policy would allow exploiting the sparsity of spikes typical of SNNs, with considerable savings in switching power, but again is not supported by GPUs.

This incompatibility has pushed for developing specialized hardware accelerators explicitly designed to support SNN features. Interestingly SNNs are intrinsically more suitable for developing these accelerators than other ANN models. The main reason is again how SNN encodes the information, which in the digital domain corresponds to single-bit signals. This drastically reduces the interconnection and memory requirements.



(a) Multi-bit digital coding



(b) Spikes information encoding

When designing hardware accelerators, the roads that can be followed are two: fixed hardware or reconfigurable hardware. In the first case, the accelerator becomes an Application Specific Integrated Circuit (ASIC), able to perform in a very efficient way the specific tasks for which it has been designed, but nothing more. On the other hand, specific hardware platforms can be reconfigured multiple times, allowing for flexible architectures. This is the case of Field Programmable Gate Arrays (FPGA). The choice between the two depends a lot on the kind of application and constraints. Generally, designers try to fit as many functionalities as possible when developing an ASIC since the hardware remains fixed.

Companies and universities are investing in developing programmable chips to enable fast simulation of large-scale SNN. Among them, a first attempt is SpiNNaker [42]. The idea behind SpiNNaker is to use classical CMOS architectures, particularly ARM968 RISC processors, to simulate the neurons' dynamics and optimize the routing between them to fit a spike-based communication perfectly. A specific communication protocol, i.e., Address Event Representation (AER), is used to do this. This protocol is designed explicitly for neuromorphic circuits, in which a spike is represented through the ID of the neuron that generated it and the corresponding generation time stamp. The IBM TrueNorth [18] is a fully custom ASIC realized in 28nm CMOS technology. Again the neurons use standard CMOS digital gates, but the communication is asynchronous. The Intel Loihi [43] has an approach that is a hybrid between the previous two. It has 128 custom chips implementing 1024 LIF primitives. The custom connection mesh optimizes the typical sparse communication between spiking neurons. Additionally, Loihi includes specific components to perform learning directly on-chip. It provides a set of configurable parameters to allow different local learning rules, from a basic STDP to more complex alternatives. Finally, Tianjic [44] is a hybrid ANN/SNN accelerator with a custom interconnection between the various cores.

In the field of ASICs, it is worth citing ODIN [45], a small network implemented in 28nm Fully-Depleted Silicon-On-Insulator (FDSOI) CMOS technology and targeting low-power applications. Neurons can be configured to implement a LIF model or one of the 20 possible Izhikevich behaviors. The routing of the spikes between neurons is again performed through an AER protocol.

Table I compares the accelerators in terms of area and energy efficiency, expressed in Giga-Synapse Operations (GSOPS) per Watt. For a more detailed comparison, see [46].

TABLE I: ASIC comparison table [46]

Design	[42]	[18]	[43]	[44]	[45]
# of neurons	18k	1M	128k	39k	256
# of synapses	18M	256M	128M	9.75M	64k
Area (mm^2)	88.4	413	60	14.4	0.086
Process (nm)	130	28	14	28	65
Energy (GSOPS/W)	0.033	400	-	649	78.7

The last approach in designing the SNN accelerators is to target a reconfigurable hardware platform, such as an FPGA. The FPGA can host different accelerators. This is the reason why many embedded platforms are starting to include them. Second, online reconfigurability allows hardware modification while the system is on. This can be used to add an accelerator after the system has been deployed, remove it once it is no longer required, and modify its functionality.

Taking SNN as an example, the architecture of the network can be modified to target a different set of data if necessary. Finally, partial reconfigurability can add and remove functionalities to a specific accelerator. For example, online learning can be activated and deactivated on request, enabling and disabling the corresponding circuitry and physically adding or removing the required piece of hardware, guaranteeing the optimal architecture for the required application. This is the idea behind [25], where the authors started to design a tiny hardware accelerator to fit an FPGA together with other components employing a LIF neuron model.

The idea is then to have a first degree of reconfigurability, making the accelerator programmable in many aspects, such

TABLE II: FPGA comparison table [47]

Design	[48]	[49]	[50]	[47]	[25]
Clock frequency(MHz)	75	120	25	100	100
Data format	16bit Fixed	8bit Fixed	32bit Fixed	16bit Floating	16bit Fixed
Computing scheme	Event-Driven	Clock-Driven	Event-Driven	Adaptive Clock/Event-Driven	Clock-Driven
Neuron model	LIF	LIF	LIF	LIF	LIF
FPGA platform	Spartan 6	Virtex 6	Spartan 6	Virtex 7	Artix 7
Neurons	1794	1591	1794	1094	1384
Synapses	647000	638208	647000	177800	313600
Task	MNIST	MNIST	MNIST	MNIST	MNIST
Computation time	$0.53s$/image	$8.40s$/image	$0.16s$/image	$3.15ms$/image	$215\mu s$/image
Computation time @100MHz	$0.40s$/image	$10.08s$/image	$40.00ms$/image	$3.15ms$/image	$215\mu s$/image
Energy	$0.80J$/image	$1.12J$/image	Not reported	$5.04mJ$/image	$13mJ$/image
Energy/Synapse	$1.2\mu J$/synapse	$1.76\mu J$/synapse	Not reported	$0.028\mu J$/synapse	$0.041\mu J$/synapse

as weights and thresholds. Then, to add a layer of flexibility by allowing easy modification of the hardwired network hyper-parameters, such as the membrane time constant, the network architecture, the internal bit-widths, etc. Several other works are targeting a more standard but still configurable implementation, such as [48], [49], [50], [47]. Table II compares different accelerators. For more details, see [47].

In general, digital accelerators can help a lot in increasing the execution efficiency of SNNs. CMOS technology is decades old and nowadays widespread, low-cost, and highly optimized. However, the intrinsic behavior of digital devices is very far from that observed in biological components, and the response time and power consumption are still a burden. Augmented silicon photonics platforms can cover most design aspects with more efficient solutions.

IV. Augmented silicon photonics platforms

Integrated photonics is one of the key technologies that has been investigated to build neuromorphic hardware [51]–[54]. In particular, Photonic Neural Networks (PNNs) based on silicon photonics have been extensively investigated for developing lightweight, low-latency, high-speed computing hardware with ultra-low power consumption [54]–[57].

Such properties arise from the intrinsic nature of light manipulation and propagation and the capabilities currently integrated photonics platforms can offer. For example, different frequencies of light can be used to encode different data streams separately onto each frequency and then be processed in parallel, thus increasing computing density [53]. Such wavelength multiplexing approaches are beneficial for increasing the parallelization degree of architectures. This key feature is used in broadcast and weight protocol where each frequency channel has a specific weight assigned (for each layer) before being summed up together, e.g., by a photodetector [58]. In this approach, ring resonators are key devices enabling the multiplexing (filtering) and weighting of the signals at different frequencies [54].

Indeed, photonic approaches allow light manipulation (e.g., weights application) while preserving signals propagation at the speed of light throughout the photonic network, thus resulting in ultra-low latencies, limited only by the physical size of the network leading to orders of magnitude lower values compared to electronics implementations [55].

Another essential feature of photonic neuromorphic systems is the possibility of operating with analog complex-valued signals, which is beneficial to leverage non-linearities in neuromorphic hardware such as the electro-optic conversion between complex-valued optical fields into intensities (i.e., photocurrents at the photodetection), but also thanks to connection matrices presenting a more considerable richness in degrees of freedom [52], [55], [59].

Furthermore, PNNs can operate at much higher speeds than digital accelerators, with their main limitation coming from electro-optic conversion stages, e.g., at the read-out of a PNN where photodetectors allow to operate at speeds of hundreds of GHz depending on the technology and responsivity required [55].

Among the various integrated photonic platforms that have been considered, Silicon-on-Insulator (SOI) platforms are those that have attracted the most vital interest thanks to the availability of both active and passive components and their reduced footprint compared to platforms with lower refractive indices contrasts, such as Silicon Nitride-on-Insulator (SiNOI).

Table III shows a comparison under different metrics for digital accelerators, flash technology, and three different types of PNNs based on hybrid lasers, co-integrated silicon photonics, and sub-λ nanophotonics. For the latter, the device footprint shrinks by at least an order of magnitude due to the robust localization of the optical fields, e.g., in photonic crystal cavities. It is worth noting that such constrained photonic approaches can expect a significant gain in energy consumption and latency. More information on the specific implementations can be found here [56].

In particular, SOI platforms can provide high-speed modulators, e.g., in MZI or Ring Resonator (RR) configurations, as well as high-speed broadband photodetectors ($>$ 50 GHz) and low propagation losses ($<$ 3 dB/cm) [65]. More specifically, MZI devices allow to modulate light and change its amplitude (and phase), therefore providing a practical way to implement ANN weights [55]. They can be arranged in meshes in precise ways, e.g., matrix multiplications as shown in Fig. 3(a) [60]. The traditionally used physical mechanisms for modulation are either (i) thermo-optic, (ii) electro-optic, (iii) carrier plasma effect [66] as schematically shown in Fig. 3(b). For the thermo-optic approach, one of the interferometer arms is heated up by

TABLE III: Photonic versus electronic approaches comparison table. Latency is the time for a single matrix multiplication operation to compute at the given vector size. Speed is the time between subsequent matrix multiplies [56].

Technology	Google TPU [61]	Flash (Analog) [62]	Hybrid laser NN [63]	Co-Integrated Si NN [64]	Sub-λ nanophotonics [56]
Energy/MAC [fJ]	430	7	220	2.7	0.03
Comp. density [TMACs/s/mm^2]	0.58	18	4.5	50	5000
Vector size	256	100	56	148	300
Precision [bits]	8	5	5.1+	5.1+	5.1+
Latency/speed [ns]	2000/1.42	15	< 0.1	< 0.1	< 0.05

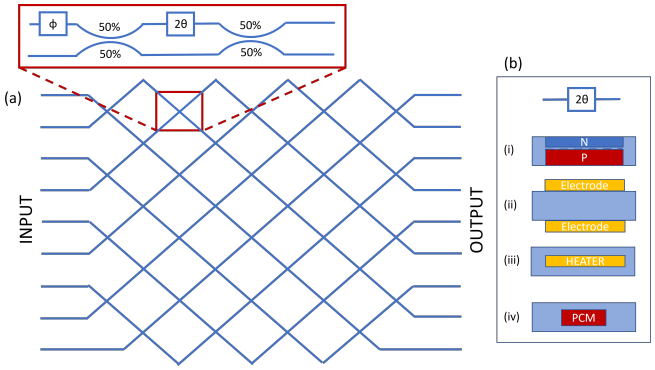


Fig. 3: (a) Photonic implementation of a network for 9x9 matrix multiplication based on Mach-Zenhder Interferometers (MZIs) [60]. Inset describes what each crossing consists of, i.e., a phase shifter (ϕ), then a 50/50 splitter, another phase shifter (2θ), and then a 50/50 combiner. (b) Top-view of phase shifter implementations in waveguides based on (i) thermo-optic effect, (ii) electro-optic effect, (iii) carrier plasma effect, and (iv) Phase Change Material (PCM)-induced shift. Blue lines where no crossing is present are optical waveguides.

a micro-heater, resulting in a change in the refractive index of the arm. In the second case, electrodes can establish an electric field (for electro-optic materials - not present in native Si platforms) that modifies the refractive index. In the third case, a p-n/p-i-n junction is used where carriers concentration in the depletion region is modified by an applied voltage, thus changing the refractive index by the plasma carrier effect [66]. All these approaches are of interest (depending on the platform available) and have enabled vector-matrix multiplication and PNNs for classification tasks [55].

However, in silicon platforms, one of the main limitations during the inference process is the need to dissipate energy to keep the values of the weights. This energy can account for up to 10 mW with a total π phase shift per MZI. Such a shift allows routing full signals from one output to the other of an MZI with two output ports (see inset of Fig. 3)(a) starting from a signal only coming from the former port.

In [55], the authors used a specific arrangement of mesh devices to perform matrix multiplication through singular value decomposition. This mapping is precisely defined between the elements of the matrix and the phase shifters MZIs in the mesh. Unlike conventional electronics like CPUs and GPUs, the energy consumed per calculation (Floating Point Operations per Second (FLOP)) reduces to zero as the size of the mesh grows larger (assuming non-volatile weights). In contrast, conventional electronics require a fixed amount of energy per FLOP, and the overall energy consumption scales quadratically with the size of the problem, i.e., N^2 instead of just N for PNNs with N representing the mesh size. The authors in [55] also used this MZIs as weights to build a Feed-Forward Neural Network (FFNN) that could analyze vowels, achieving a simulated correctness of 90%, comparable to that of a digital computer which would reach 91.7%.

One solution to avoid constant power dissipation is setting the weights using waveguides integrating non-volatile materials such as PCMs, e.g., above the waveguide as in (iv) approach in Fig. 3)(b). Their response (change in the degree of crystalline to amorphous ratio) can be set using electrical or optical pulses [67]. Such materials are exciting also to implement STDP thanks to their very rapid response time (sub-ns) to stimuli, thus allowing to build of optical plastic synapses and spiking architectures based on ring resonators [54].

The recently started Horizon Europe NEUROPULS project investigates a series of approaches that leverage silicon photonics platforms with the addition of PCMs and III-V materials for building more efficient neuromorphic hardware.

An approach to improving CNN performance is to use an accelerator designed for matrix-vector products, such as a mesh of modulators programmed to perform a specific matrix multiplication. Singular value decomposition can factor the matrix, resulting in more efficient computation on the accelerator. This approach can also handle other matrix operations, including convolutions, and optimize hardware architecture for specific tasks, leading to significant training and inference times speed-ups [55].

Although optical components can benefit matrix operations, their size can limit the size of matrices that can be implemented. To overcome this limitation, alternative approaches can be explored. One possible solution is to employ pruning techniques that remove unnecessary connections and weights from the network, thereby reducing the overall size of the matrix. This makes it possible to implement larger systems using the available optical components.

Another option is to use block matrix decompositions, which involve dividing the matrix into smaller blocks that can be processed separately. This approach can enable the implementation of larger matrices using a smaller number of optical components. The smaller blocks can be computed

independently and combined for the final result. This technique can also be combined with other optimization strategies, such as quantization and compression, further to reduce the size and complexity of the system.

The tensor-train approach proposed in [68] will be explored. This approach represents the matrix as a product of low-rank tensors, which can be processed more efficiently using optical components. The goal is to develop a scalable optical architecture capable of handling larger matrices and more complex neural networks using the abovementioned approaches. This will facilitate faster and more efficient training of neural networks using photonics.

The research will also explore RNN applications, including fully trainable RNNs as well as RCs. We will investigate the potential applications of RC in photonics for various applications, including nonlinear dispersion compensation of telecom signals, as demonstrated in [69]. The research aims to develop new techniques and approaches for utilizing photonics in Machine Learning (ML) and other fields, which could lead to significant advancements in the performance and efficiency of these systems.

To implement non-volatile optical weights, the proposed architecture will incorporate PCMs. Previous studies, such as [54] and [70], have explored these materials and shown a strong potential for neuromorphic systems. Incorporating non-volatile weights can significantly reduce power consumption compared to volatile weights, which require continuous driving or periodic refreshing. Including these materials in the proposed architecture will be crucial in developing low-power and high-performance systems for machine learning and other applications.

However, in addition to their non-volatility, PCMs have another advantage: their nonlinear dynamics. E.g., by exciting the material with pulses rather than continuous-wave excitation, the nonlinear behavior of the material enables other computing paradigms, such as SNNs. In such networks, the neurons communicate using brief pulses or spikes rather than continuously varying signals. This opens up the implementation of energy-efficient and highly parallel neural networks. To generate the spikes injected into the system, Advanced high-extinction ratio (ER $>$ 8 dB) Q-switched spiking lasers can be used, which will be monolithically integrated into III-V materials on the same platform [71]. These hybrid III-V-on-Si spiking lasers are a scalable and cost-effective alternative to previous Q-switched lasers made purely from III-V materials.

III-V-on-Si spiking lasers will generate highly precise and controlled optical spikes, essential for many photonics and ML applications. These lasers offer several advantages, including high extinction ratios, low power consumption, and compatibility with standard silicon processing techniques.

V. Reliability studies and concerns

ANNs have an intrinsic error tolerance from an algorithmic point of view thanks to their redundant nature. However, hardware designs to deploy such algorithms must be analyzed to assess the impact of hardware restrictions or faulty manifestation on the network functional's behavior.

Due to manufacturing issues, hardware faults can occur randomly, provoked by neuron and synapse defects and imprecisions. Still, they can also be malicious, introduced by different kinds of attacks (i.e., laser beams fault injection or row hammer attack) [72], [73]. The authors of [73] have analyzed the misclassification rate of MLP based deep neural networks face to models derived from physical phenomena. Faulty-neuron behaviors have been injected randomly or deterministically, with injection scenarios considering single and multiple faulty neurons per layer. This is usually done during the function activation timeframe, which can be hundreds to thousands of cycles. Results indicate that in some cases, even a relatively small number of faulty neurons ($\approx 10\%$) can lead to a high risk of misclassification ($\approx 62\%$). As seen in section II, different activation functions have been studied for DNNs. They show that for a higher miss-classification rate ($>$50%), at least half of the neurons in a given hidden layer should be faulty, which is the case for sigmoid and $\tanh$ activation functions. In the case of ReLU, at least 3/4 of the neurons should be faulty to achieve the same miss-classification rate.

Extensive work has been dedicated in the last years to studying and evaluating AI hardware accelerators' errors and fault tolerance. An overview of fault tolerance techniques for feedforward neural networks is presented in [74]. In this paper, the authors review fault types, models, and measures used to evaluate performance and provide a taxonomy of the main techniques to enhance the intrinsic properties of some neural models based on the principles and mechanisms they exploit to achieve fault tolerance passively.

In [75], the authors present a study of the fault characterization and mitigation of Register-Transfer Level (RTL) model of NN accelerators by characterizing the vulnerability of NNs to application-level specifications, network topology, and activation functions, as well as architectural level specifications. In [76], authors present an experimental evaluation of the resilience of DNN systems (i.e., DNN software running on specialized accelerators) under Soft errors caused by high-energy particles.

An empirical study of DNNs resilience can be found in [77], where a fault injection framework named Ares, which can deal with fully connected and CNN-based DNN accelerators, is presented. It uses hardware fault models related to technology and environment variability, single event faults transients in memory elements, and algorithmic level faults models such as faults occurring in weights, activation, and hidden states. Fault injection is performed static, offline, before the inference process, and dynamically during inference execution.

The analysis of SNNs fault tolerance and reliability is a relatively newer field of research since their hardware implementations are much more recent than classical DNNs. Consensus has yet to be reached on the main applications of these networks. Moreover, the variety of signal-to-spike coding and training algorithms brings a specific heterogeneity in their characteristic which should be accounted for when

their fault tolerance and reliability is discussed. One of the first works in this field, [78], estimates the accuracy of a FC SNN, capable of STDP learning designed with spintronic devices under the effect of process variability. Both the neuron and synapse behavior are strongly affected by process variability, and the accuracy drops by approximately 10% when assuming moderate variability compared to the ideal case.

Another interesting study is presented in [79], where a taxonomy of faults was defined for spiking neural networks. The accuracy of a hardware-implemented spiking neural network designed to perform STDP online training was analyzed under the assumption that (i) both learning and inference were performed on faulty hardware, (ii) only inference was performed on faulty hardware. This paper shows that performing learning directly on faulty hardware reduces the impact of faults on the network accuracy by an average of 15% and, in extreme cases, can reach up to 30%. Moreover, in [80], faults affecting the signal-to-spike conversion layer have the strongest impact on the network accuracy, with the synaptic stuck-at faults coming to a very close second. These faults strongly affect the learning process, which only exacerbates during inference.

On the other hand, faults, like delayed synapse activation or stuck lateral inhibition, have a marginal effect. The study presented in [81] takes a different approach. The fault tolerance study is performed on an SNN inference accelerator, a multi-layer SNN with supervised off-line learning, designed in VHDL and implemented on an FPGA. The fault injection experiments identify the parts of the design that need to be protected against faults and the inherently fault-tolerant parts. They have shown that the behavior under faults of the chosen type of SNN is similar to the behavior under faults of an ANN in that faults injected in the most significant bits of the synaptic values affect have a more substantial effect on network accuracy that when the faults are injected in the less critical bits, and also faults affecting the last layer (where the classification is performed) are more relevant than faults affecting the first layer (where there is higher computing redundancy). In addition, the authors have shown that their proposed SNN implementation is much more sensitive to faults injected in the routing of signals than in the synaptic weight or neuron computation.

Several studies have compared SNNs and their ANNs counterparts, mainly focusing on performance or power consumption rather than their relative fault tolerance. Therefore, a study was conducted to assess the fault resilience of both network types, assuming different quantizations of weights and neural computation and various training scenarios for the SNN. The study compared the fault tolerance of an MLP and an SNN with the same topology and bit precision. A 784 X 100 X 10 network was implemented to solve the MNIST classification problem [6]. The MLP uses a sigmoid activation function for all layers and has a base accuracy of 98% after 20 training epochs. The SNN used rate coding for signal-to-spike conversion and several training algorithms: (i) Shadow Training (ST) (with a base accuracy of 96%), (ii) BPTT (with a base accuracy of 97%), (iii) STDP (with a base accuracy of 87%). The precision of synaptic weights was 16, 8, and 4 bits. Figure 4 illustrates selected results of the analysis, showing that the SNNs are more fault-tolerant than the MLP, and the SNN trained using BPTT is the most resilient of the three. This study also demonstrates how the SNN is trained in its fault resilience, with online training being more resilient than offline training. The last two columns in Figure 5 correspond to the SNN being trained by STDP, and the network accuracy degradation is computed when the presence of faults is assumed only at inference (AT) or during both training and inference (BT). These results show that the fault tolerance of the SNN depends on the training mechanism and that an SNN trained online is more resilient to faults than an SNN trained offline. The results suggest that SNNs have the potential to be more resilient to faults than ANNs, and further comparison between the two is necessary to consolidate these findings.

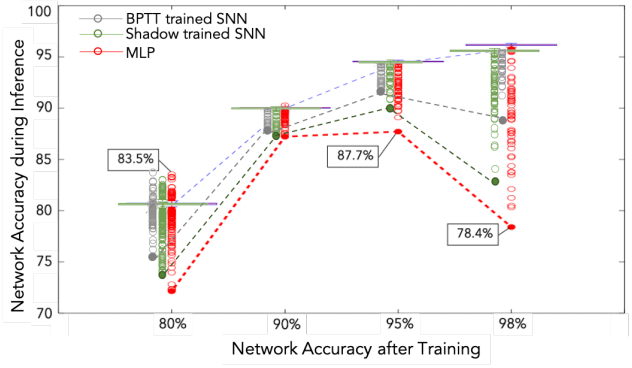


Fig. 4: Network accuracy under synaptic fault injection: a comparison between MLP and SNN. The synaptic weight precision is 8 bits, and 10^{-4} of the synaptic bits are considered faulty.

	Relative accuracy loss				
	ANN	ST SNN	BPTT SNN	STDP SNN	
				BT	AT
@Max accuracy	20,00%	14,58%	9,28%	3,07%	5,36%
95%	7,68%	6,32%	4,21%	3,17%	3,64%
90%	4,44%	4,00%	3,33%	3,21%	6,50%
80%	10,00%	7,50%	5,00%	3,33%	6,32%

Fig. 5: Maximum network accuracy degradation for different training scenarios under synaptic fault injection: a comparison between MLP and SNN. The synaptic weight precision is 8 bits, and 10^{-4} of the synaptic bits are considered faulty. AT = after training, BT = before training

VI. Conclusion

In this paper, the fundamentals of Convolutional Neural Network, Deep Neural Network, and Spiking Neural Network were introduced, and their digital counterparts were described using standard CMOS technologies. The discussion then focused on augmented silicon photonics improvements and the reliability aspects of Artificial Neural Network. The benefits

and drawbacks of each technology were thoroughly analyzed, including its reliability. Our analysis suggests that the choice of technology for Artificial Neural Network design should depend on the specific application requirements and design constraints. While CMOS technology offers established fabrication processes and high reliability, it may have limitations in power consumption and scalability. Silicon photonics, on the other hand, provides high power efficiency and scalability, but their reliability is still under ongoing research. We hope this paper will serve as a valuable reference for researchers and practitioners in the field of Artificial Neural Network design as they explore the potential of these emerging technologies.

REFERENCES

[1] A. Marchisio *et al.*, "Deep learning for edge computing: Current trends, cross-layer optimizations, and open research challenges," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2019, pp. 553–559.

[2] M. Shafique *et al.*, "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the iot era," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 827–832.

[3] S. Dutto *et al.*, "Exploring deep learning for in-field fault detection in microprocessors," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1456–1459.

[4] H. Sedjelmaci *et al.*, "A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[5] C.-W. Zhou *et al.*, "A brief survey on deep neural networks," *arXiv preprint arXiv:1802.08882*, 2018.

[6] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11. IEEE, 1998, pp. 2278–2324.

[7] I. Goodfellow *et al.*, "Deep learning," in *Deep learning*. MIT Press, 2016, pp. 1–800.

[8] C. D. Schuman *et al.*, "Opportunities for neuromorphic computing algorithms and applications," *Nat Comput Sci*, vol. 2, no. 1, pp. 10–19, Jan. 2022, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s43588-021-00184-y

[9] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural networks*, vol. 10, no. 9, 1997.

[10] B. Cramer *et al.*, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2022.

[11] A. Amirshahi and M. Hashemi, "Ecg classification algorithm based on stdp and r-stdp neural networks for real-time monitoring on ultra low-power personal wearable devices," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 6, pp. 1483–1493, 2019.

[12] A. Vallero *et al.*, "Syra: Early system reliability analysis for cross-layer soft errors resilience in memory arrays of microprocessor systems," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 765–783, 2019.

[13] A. Ruospo *et al.*, "A survey on deep learning resilience assessment methodologies," *Computer*, vol. 56, no. 2, pp. 57–66, 2023.

[14] C. D. James *et al.*, "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," *Biologically Inspired Cognitive Architectures*, vol. 19, pp. 49–64, Jan. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212683X16300561

[15] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, may 2017. [Online]. Available: https://doi.org/10.1145/3065386

[16] D. E. Rumelhart *et al.*, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1127647

[18] F. Akopyan *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, 2015.

[19] S. Draghici, "On the capabilities of neural networks using limited precision weights," *Neural Networks*, vol. 15, no. 3, pp. 395–414, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608002000321

[20] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1254642

[21] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[22] E. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[23] C. Teeter *et al.*, "Generalized leaky integrate-and-fire models classify multiple neuron types," *Nature Communications*, vol. 9, no. 1, p. 709, 2018.

[24] A. N. Burkitt, "A review of the integrate-and-fire neuron model: Ii. inhomogeneous synaptic input and network properties," *Biological Cybernetics*, vol. 95, no. 2, pp. 97–112, 2006.

[25] A. Carpegna *et al.*, "Spiker: an fpga-optimized hardware accelerator for spiking neural networks," in *2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2022, pp. 14–19.

[26] F. Corradi and G. Indiveri, "A neuromorphic event-based neural recording system for smart brain-machine-interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 5, pp. 699–709, 2015.

[27] G. V. der Sande *et al.*, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, pp. 561–576, 2017. [Online]. Available: https://doi.org/10.1515/nanoph-2016-0132

[28] D. J. Enoka RM, "Rate coding and the control of muscle force," *Cold Spring Harb Perspect Med*, 2017.

[29] B. Petro *et al.*, "Selection and optimization of temporal spike encoding methods for spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 2, pp. 358–370, 2020.

[30] Z. Pan *et al.*, "Neural population coding for effective temporal classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[31] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[32] J. Duchi *et al.*, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2121–2159, jul 2011.

[33] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, jan 2014.

[34] R. Takahashi *et al.*, "Ricap: Random image cropping and patching data augmentation for deep cnns," in *Proceedings of The 10th Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Zhu and I. Takeuchi, Eds., vol. 95. PMLR, 14–16 Nov 2018, pp. 786–798. [Online]. Available: https://proceedings.mlr.press/v95/takahashi18a.html

[35] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, L. Alvarez *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 14–36.

[36] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010.

[37] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013.

[38] P. M. Bi GQ, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *The Journal of Neuroscience*, vol. 18, no. 24, 1998.

[39] J. K. Eshraghian *et al.*, "Training spiking neural networks using lessons from deep learning," *arXiv preprint arXiv:2109.12894*, 2021.

[40] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio *et al.*, Eds. Curran Associates, Inc., 2018, pp. 1419–1428. [Online]. Available: http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf

[41] T.-S. Chou *et al.*, "Carlsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[42] S. B. Furber *et al.*, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, 2014.

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

[44] J. Pei *et al.*, "Towards artificial general intelligence with hybrid tianjic chip architecture," *Nature*, vol. 572, no. 7767, 2019.

[45] C. Frenkel *et al.*, "A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, 2019.

[46] A. Basu *et al.*, "Spiking neural network integrated circuits: A review of trends and future directions," in *2022 IEEE Custom Integrated Circuits Conference (CICC)*, 2022.

[47] S. Li *et al.*, "A fast and energy-efficient snn processor with adaptive clock/event-driven computation scheme and online learning," *IEEE transactions on circuits and systems. I, Regular papers*, vol. 68, no. 4, pp. 1543–1552, 2021.

[48] D. Neil and S.-C. Liu, "Minitaur, an event-driven fpga-based spiking network accelerator," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 22, no. 12, pp. 2621–2628, 2014.

[49] Q. Wang *et al.*, "Energy efficient parallel neuromorphic architectures with approximate arithmetic on fpga," *Neurocomputing (Amsterdam)*, vol. 221, pp. 146–158, 2017.

[50] D. Ma *et al.*, "Darwin: A neuromorphic hardware co-processor based on spiking neural networks," *Journal of systems architecture*, vol. 77, pp. 43–51, 2017.

[51] Y. Paquot *et al.*, "Optoelectronic Reservoir Computing," *Scientific Reports*, vol. 2, no. 1, p. 287, Feb. 2012. [Online]. Available: https://doi.org/10.1038/srep00287

[52] K. Vandoorne *et al.*, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nat Commun*, vol. 5, no. 1, p. 3541, Mar. 2014, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/ncomms4541

[53] H.-T. Peng *et al.*, "Neuromorphic Photonic Integrated Circuits," *IEEE J. Select. Topics Quantum Electron.*, vol. 24, no. 6, pp. 1–15, Nov. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8364605/

[54] J. Feldmann *et al.*, "All-optical spiking neurosynaptic networks with self-learning capabilities," *Nature*, vol. 569, no. 7755, pp. 208–214, May 2019. [Online]. Available: http://www.nature.com/articles/s41586-019-1157-8

[55] Y. Shen *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photon*, vol. 11, no. 7, pp. 441–446, Jul. 2017. [Online]. Available: http://www.nature.com/articles/nphoton.2017.93

[56] M. A. Nahmias *et al.*, "Photonic Multiply-Accumulate Operations for Neural Networks," *IEEE J. Select. Topics Quantum Electron.*, vol. 26, no. 1, pp. 1–18, Jan. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8844098/

[57] J. Feldmann *et al.*, "Parallel convolutional processing using an integrated photonic tensor core," *Nature*, vol. 589, no. 7840, pp. 52–58, Jan. 2021. [Online]. Available: http://www.nature.com/articles/s41586-020-03070-1

[58] Y. Nahmias and Y. Loewenstein, "Neuromodulation of hebbian plasticity: relevance and mechanisms," *Current Opinion in Neurobiology*, vol. 69, pp. 150–159, 2021.

[59] M. Abdalla *et al.*, "Minimum complexity integrated photonic architecture for delay-based reservoir computing," *Opt. Express*, vol. 31, no. 7, p. 11610, Mar. 2023. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=oe-31-7-11610

[60] W. R. Clements *et al.*, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, p. 1460, Dec. 2016. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=optica-3-12-1460

[61] "In-Datacenter Performance Analysis of a Tensor Processing Unit | Proceedings of the 44th Annual International Symposium on Computer Architecture." [Online]. Available: https://dl.acm.org/doi/10.1145/3079856.3080246

[62] M. R. Mahmoodi and D. Strukov, "An ultra-low energy internally analog, externally digital vector-matrix multiplier based on NOR flash memory technology," in *Proceedings of the 55th Annual Design Automation Conference*. San Francisco California: ACM, Jun. 2018, pp. 1–6. [Online]. Available: https://dl.acm.org/doi/10.1145/3195970.3195989

[63] M. A. Nahmias *et al.*, "A Leaky Integrate-and-Fire Laser Neuron for Ultrafast Cognitive Computing," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, no. 5, pp. 1–12, Sep. 2013, conference Name: IEEE Journal of Selected Topics in Quantum Electronics.

[64] A. N. Tait *et al.*, "Silicon Photonic Modulator Neuron," *Phys. Rev. Appl.*, vol. 11, no. 6, p. 064043, Jun. 2019, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.11.064043

[65] N. Margalit *et al.*, "Perspective on the future of silicon photonics and electronics," *Appl. Phys. Lett.*, vol. 118, no. 22, p. 220501, May 2021. [Online]. Available: https://aip.scitation.org/doi/10.1063/5.0050117

[66] W. Bogaerts *et al.*, "Silicon microring resonators," *Laser & Photon. Rev.*, vol. 6, no. 1, pp. 47–73, Jan. 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/lpor.201100017

[67] C. Rios *et al.*, "Controlled switching of phase-change materials by evanescent-field coupling in integrated photonics [Invited]," *Opt. Mater. Express*, vol. 8, no. 9, p. 2455, Sep. 2018. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=ome-8-9-2455

[68] X. Xiao *et al.*, "Large-scale and energy-efficient tensorized optical neural networks on iii–v-on-silicon moscap platform," *APL Photonics*, vol. 6, no. 12, p. 126107, 2021. [Online]. Available: https://doi.org/10.1063/5.0070913

[69] S. Sackesyn *et al.*, "Experimental realization of integrated photonic reservoir computing for nonlinear fiber distortion compensation," *Opt. Express, OE*, vol. 29, no. 20, pp. 30 991–30 997, Sep. 2021, publisher: Optica Publishing Group. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?uri=oe-29-20-30991

[70] M. Miscuglio *et al.*, "Artificial synapse with mnemonic functionality using gsst-based photonic integrated memory," in *2020 International Applied Computational Electromagnetics Society Symposium (ACES)*, 2020, pp. 1–3.

[71] K. Mekemeza-Ona *et al.*, "All optical q-switched laser based spiking neuron," *Frontiers in Physics*, vol. 10, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2022.1017714

[72] Y. Liu *et al.*, "Fault injection attack on deep neural network," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017, pp. 131–138.

[73] J. Breier *et al.*, "Deeplaser: Practical fault attack on deep neural networks," 2018.

[74] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17 322–17 341, 2017.

[75] B. Salami *et al.*, "On the resilience of rtl nn accelerators: Fault characterization and mitigation," in *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018, pp. 322–329.

[76] G. Li *et al.*, "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.

[77] B. Reagen *et al.*, "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.

[78] 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.

[79] E.-I. Vatajelu *et al.*, "Special session: Reliability of hardware-implemented spiking neural networks (snn)," in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019, pp. 1–8.

[80] A. Bosio *et al.*, "Rebooting computing: The challenges for test and reliability," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2019, pp. 8138–8143.

[81] T. Spyrou *et al.*, "Reliability analysis of a spiking neural network hardware accelerator," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 370–375.