Spiking Neural Networks Hardware Implementations and Challenges: A Survey

MAXENCE BOUVIER, ALEXANDRE VALENTIAN, THOMAS MESQUIDA, FRANCOIS RUMMENS, MARINA REYBOZ, ELISA VIANELLO, and EDITH BEIGNE,
CEA-LETI, Minatec Campus

Neuromorphic computing is henceforth a major research field for both academic and industrial actors. As opposed to Von Neumann machines, brain-inspired processors aim at bringing closer the memory and the computational elements to efficiently evaluate machine learning algorithms. Recently, spiking neural networks, a generation of cognitive algorithms employing computational primitives mimicking neuron and synapse operational principles, have become an important part of deep learning. They are expected to improve the computational performance and efficiency of neural networks, but they are best suited for hardware able to support their temporal dynamics. In this survey, we present the state of the art of hardware implementations of spiking neural networks and the current trends in algorithm elaboration from model selection to training mechanisms. The scope of existing solutions is extensive; we thus present the general framework and study on a case-by-case basis the relevant particularities. We describe the strategies employed to leverage the characteristics of these event-driven algorithms at the hardware level and discuss their related advantages and challenges.

CCS Concepts: • **Computing methodologies** → **Bio-inspired approaches**; • **Hardware** → **Emerging architectures**; **Neural systems**; • **Theory of computation** → *Machine learning theory*; • **Computer systems organization** → *Multicore architectures*; *Neural networks*;

Additional Keywords and Phrases: Neuromorphic computing, spiking, event driven, neural network, hardware, machine learning, spiking neural networks, neuromorphic computing, hardware implementation

ACM Reference format:
Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. 2019. Spiking Neural Networks Hardware Implementations and Challenges: A Survey. *J. Emerg. Technol. Comput. Syst.* 15, 2, Article 22 (April 2019), 35 pages.
https://doi.org/10.1145/3304103

1 INTRODUCTION

Machine learning algorithms have been a growing subject of interest for nearly 40 years. The complexity of neural networks has progressively increased, and the field is now referred to as deep learning due to the very large amount of layers composing them. We distinguish two types of neural networks: the formal ones, referred to as artificial neural networks (ANNs), and the event-driven ones, referred to as spiking neural networks (SNNs). ANNs have been studied since

Authors' address: M. Bouvier (corresponding author), A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, CEA-LETI, 17 rue des martyrs, 38054, Grenoble, France; email: maxence.bouvier@cea.fr.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
1550-4832/2019/04-ART22 $15.00
https://doi.org/10.1145/3304103

the beginning of machine learning and SNNs are more recent, with the first reports referring to SNNs being published around 20 years ago. Maass [116] derived that their computational power is at least equal to the one of ANNs.

Because of the increase of the computational density of those algorithms, their evaluation on a traditional computer architecture, known as Von Neumann, now requires an extreme amount of time and power. As a result, hardware designers started working on accelerators dedicated to the evaluation of such algorithms. Neural networks are based on the biological neural computation scheme occurring in the brain, and thus it is no surprise that processors dedicated to their evaluation are also taking inspiration from the brain structure. Such circuits are therefore referred to as neuromorphic [121].

Neuromorphic computing can be applied to both formal and event-driven networks. The scope of applications for machine learning is probably infinite, but their practical realization remains limited due to hardware resource requirements, especially for embedded application. Indeed, battery-constrained devices usually require a connection to the cloud for external evaluation on more powerful machines. Hence, there is a need for low-power hardware and algorithm co-design to enlarge the applicability of deep learning.

There is a growing belief that SNNs may easily enable low-power hardware evaluation. Indeed, they are supposedly efficient for both computation and communication thanks to their event-driven sparsity. However, on this day, it is not yet clear whether hardware implementation of SNNs is effectively efficient. Therefore, in this survey, we deliver an overview of the advancement in this field. We detail the working principles of SNNs: how it can be adapted to hardware and how to leverage their performance gains.

In Section 2, we present the neural networks operation, both formal and spiking, and explain how neuromorphic computing differs from Von Neumann machines. Section 3 discusses the bio-inspiration of SNNs and how bio-plausible models can be implemented using hardware. Section 4 focuses on SNNs for machine learning applications. We review the different strategies allowing to reduce the hardware resources required to evaluate SNNs for machine learning applications and study to what extent SNNs are suited for low-power applications. We also discuss the state of the art regarding crossbar array implementations. Section 5 examines the ways of training an SNN and discusses the interest and feasibility of on-chip learning. Section 6 gives the state of the art of SNNs' hardware implementations, for both large-scale realization and small-scale targeting low power for potential embedded operation. Section 7 introduces potential paths for future improvements of SNNs, on both the software and hardware points of view. We offer our conclusion in Section 8.

2 SNN WORKING PRINCIPLES

2.1 Introduction to Neural Networks

The global interest of researchers, and more recently industry in ANNs, is based on (1) their potential as non-linear universal approximators [65] and (2) the possibility to train them—in other words, iteratively program them to realize their function with a general algorithm, namely back-propagation of the gradient, applicable to any network or target function or both, without precise prior knowledge of how the function should be described [187]. To make it simple, we describe the structure of a multi-layer perceptron—the basic ANN.

ANNs are inspired from the working mechanism of the brain. They are a succession of computational primitives, namely neurons and synapses. Neurons, also called *perceptrons*, are assembled into layers, and each layer is connected via a set of synapses, whose connections depend on the network topology. A representation of a biological neuron and its artificial perceptron couterpart is given in Figure 1.

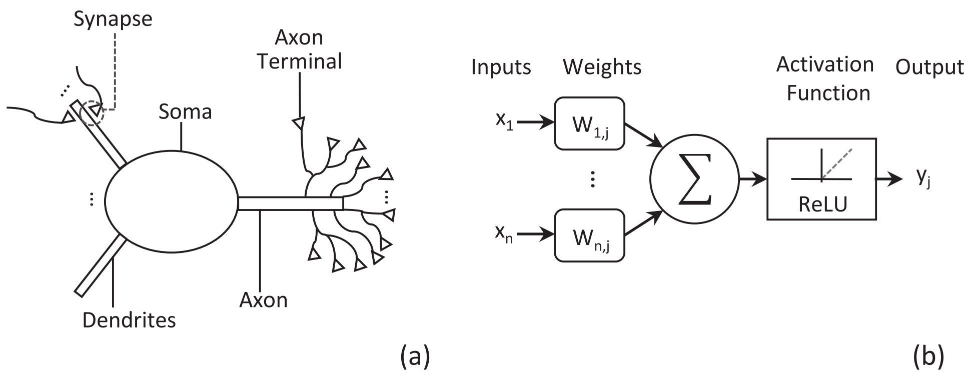

Fig. 1. (a) Schematic representation of a biological neuron. Dendrites receive inputs from upstream neurons via the synapses. The soma membrane voltage integrates those inputs and transmits its output to an axon. Axon terminals are in charge of transmission among many downstream neurons. (b) Schematic diagram of a non-linear perceptron as described by Equation (1) on top of which is added a non-linear activation, such as the represented rectified linear unit (ReLU).

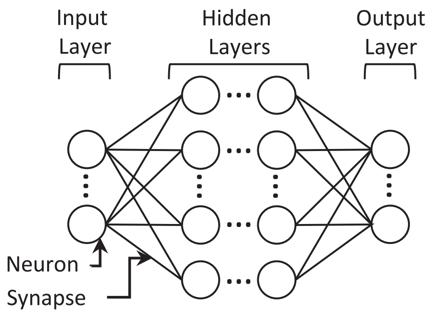

Fig. 2. Schematic representation of an FC neural network.

A synapse connects only one pre- and one post-synaptic neuron. A post-synaptic neuron implements the multiplication and summation of the pre-synaptic incoming activation values x with their corresponding synaptic weights w (also called *synaptic efficacy*) and adds a bias term b to the result. Hence, the post-synaptic neuron evaluates the following:

$$y_j = \sum_i w_{i,j} \times x_i + b_j \tag{1}$$

At that point, perceptrons are linear, whereas their non-linearity plays a crucial role for the universality of neural networks [65]. Hence, on top of that, a non-linear activation function f is applied on y_j. At the end, keeping the previous notation, an incoming activation value x of a neuron in layer l is equal to $x^l = f(y^{l-1})$.

The connections between layers may have various configurations. Fully connected (FC) networks consist of having all the units of a layer connected to every unit of the adjacent layers, such as described in Figure 2. Thus, if two layers of respective capacities n_1 and n_2 are FC, then the total number of synapses is $n_1 \times n_2$. Complementary information on the various topologies, their working principles, and tutorials can be found in several works [56, 86, 134, 161].

An SNN, which is the purpose of this survey, follows more sophisticated design rules. One could summarize the general behavior as follows: when a synapse receives an action potential, also called a *spike*, from its pre-synaptic neuron, it will emit a post-synaptic potential (PSP), which stimulates

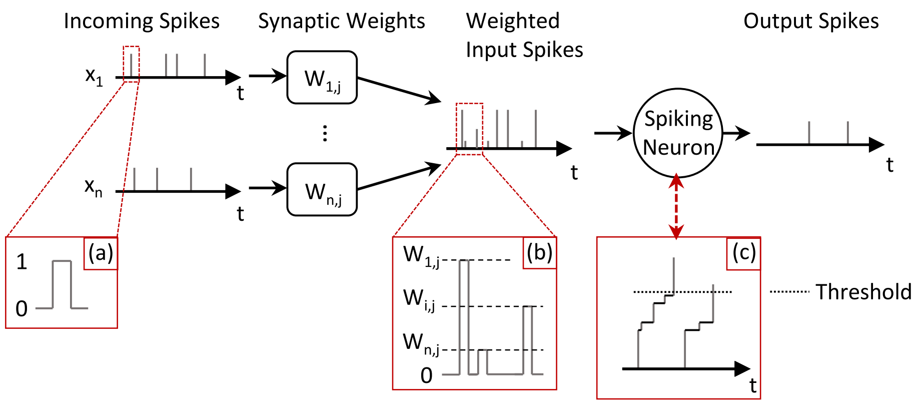

Fig. 3. Schematic representation of a spiking neuron. Incoming binary spikes, received through the synapses, are weighted and integrated. The neuron integrates them and in turn emits binary spikes to downstream units. Inlet (a) represents a binary spike. Inlet (b) illustrates the synaptic weighting of the spikes. Inlet (c) shows the integration process on the membrane potential of a simple IF neuron model.

the membrane of its post-synaptic neuron. The neuron membrane potential is a function of incoming synaptic stimulations and evolves with time. If the potential overcomes a threshold, the post-synaptic neuron fires (i.e., it emits an action potential taking the form of a spike). The operation of a simple integrate and fire (IF) neuron is illustrated in the inlet (c) of Figure 3. We discuss further the different models of neurons in Section 3.1.1. Maass [116] argues that SNNs should allow to compete with and overcome formal (as opposed to spiking) neural networks' computational power for machine learning applications. In this article, Maass compares the number of units required to realize a function and demonstrates that by using spiking neurons, the same amount is required for any function and may be less for specific functions. Moreover, the integration of time in the information propagation and computation of SNNs may enable them to efficiently extract temporal information from time-dependent data [54].

To sum up, spiking neurons communicate via impulsions, which are described using binary signals, and integrate over time the incoming spikes onto their membrane. An advanced description of the neuron and synapse operations in SNNs is discussed in more detail in Section 3.1.

2.2 Information Representation Using Spiking Neurons

In SNNs, a spike takes the form of a single binary bit. The information representation with binary spikes allows these systems to be faithfully implemented using analog or digital hardware. To code network inputs, and readout their outputs, one needs to understand how to represent complex information using spikes. However, information representation in networks of spiking neurons is still being discussed [100, 142, 156]. Different ways to represent information with spikes is depicted in Figure 4. The majority of artificial SNNs use what is commonly called *rate coding*, whereby the information transmitted between neurons is represented as the mean firing rate of emitted spikes over an observation period. Poisson spike trains are usually employed, in which case the precise timing between consecutive spikes is random but the overall frequency of spikes is fixed. However, some neuroscientists suggest that part of the information in the brain is encoded in a temporal manner [180]. Time coding can be implemented under several forms. Some use the time to first spike (TTFS) [154], where intensity of the activation is inversely proportional to the firing delay of the neuron: the one with the highest membrane potential fires first. Others, as presented elsewhere [35, 100, 124], use the inter-spike interval (ISI; i.e., the precise delay between consecutive spikes) to code the intensity of the activation. Nevertheless, Fairhall et al. [43] suggest that the encoding

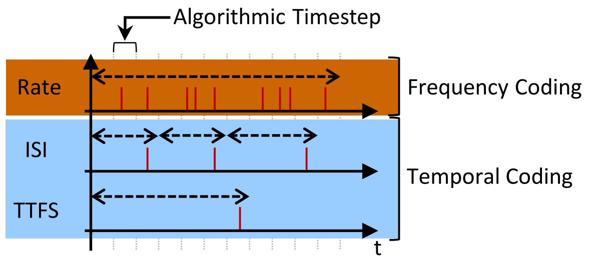

Fig. 4. Temporal diagram of the number of emitted spikes as a function of the type of coding employed.

is realized simultaneously at several time scales, which suggests that real brain communication exploits a combination of the three.

In addition, it is also possible to use information encoding only for input conversion, from frame information to event-driven information. The successive layers of neurons will then simply integrate the spike pattern they receive and fire as soon as their membrane potential overcomes their threshold.

2.3 General Hardware Implementation Strategy

2.3.1 Hardware Platform. Simulation of SNNs is usually performed by discretizing the time and evaluating the state of every neuron at each algorithmic timestep. The shorter the timestep, the more accurate the simulation, but the larger its duration. Even if leveraging the computational sparsity of SNNs in simulation is possible, by updating only the neurons receiving spikes at each step, only a dedicated hardware is able to take full advantage of the spiking behavior. To accelerate the computation of SNNs, neuromorphic ASICs and ASIPs have recently been proposed. The major realizations are BrainScaleS from a European Consortium [160], the recent Loihi from Intel [38], Neurogrid from Stanford University [15], ROLLS from the Institute of Neuroinformatics (INI Zurich) [148], SpiNNaker from the University of Manchester [49], and TrueNorth from IBM [122].

It is largely accepted that the main limitation in performance, in terms of throughput and power consumption, when evaluating ANNs comes from the memory bottleneck [9, 66, 118]. With inspiration taken from the brain computing paradigm, neuromorphic processors' ambition is thus to distribute the memory of the whole architecture within close proximity to the processing elements (PEs). This leads to parallelizing the storage and computation of the different layers of an ANN while reducing the power consumption of the whole operation. It goes the same for SNNs' hardware evaluation. A schematic representation of a neuromorphic hardware architecture parallelized at several scales is depicted in Figure 5. The computational core is separated into several small neurocores, where memory and PEs are specifically arranged. Hence, the layers of the network are distributed/mapped among the neurocores, and each neurocore both stores part of the synapse weights and performs the neuron evaluations of the SNN topology.

In input/output, a core receives/transmits spikes via the address event representation (AER) protocol through a communication scheme like a network on chip (NoC) [19, 111]. This type of architecture is scalable as long as the routers and the control circuitry can manage the AER requests. Still, the definition of the number of neurons and synapses per core, and the number of cores per chip, will limit the implementable topology onto a chip. To overcome this limitation, some implementations even realized scalable multi-chip architecture [48].

2.3.2 Algorithm to Hardware Mapping. The mapping strategy employed to evaluate an SNN onto a neuromorphic hardware may largely affect the performance of the hardware evaluation of the circuit.

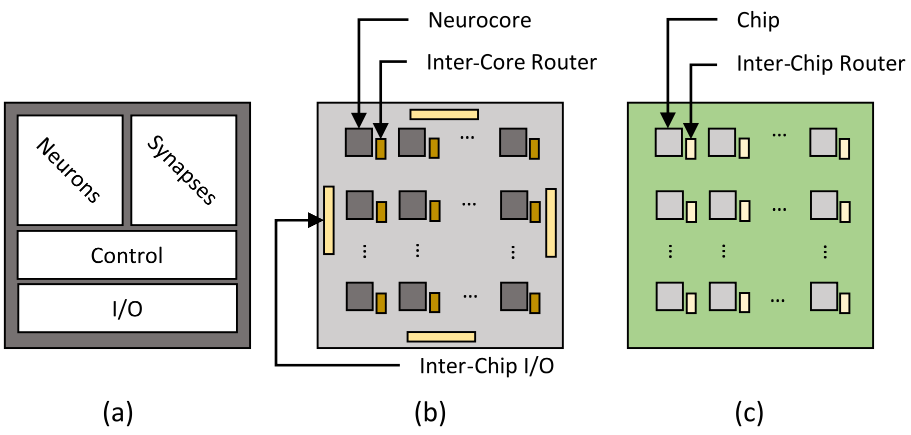

Fig. 5. General neuromorphic hardware description at different scales. (a) Neurocore representation, which integrates a local memory, algorithmic, and control circuitry to evaluate the network mapping. (b) Representation of a chip, which integrates several neurocores and an NoC to communicate AER events between cores. (c) A board representation containing several neuromorphic chips.

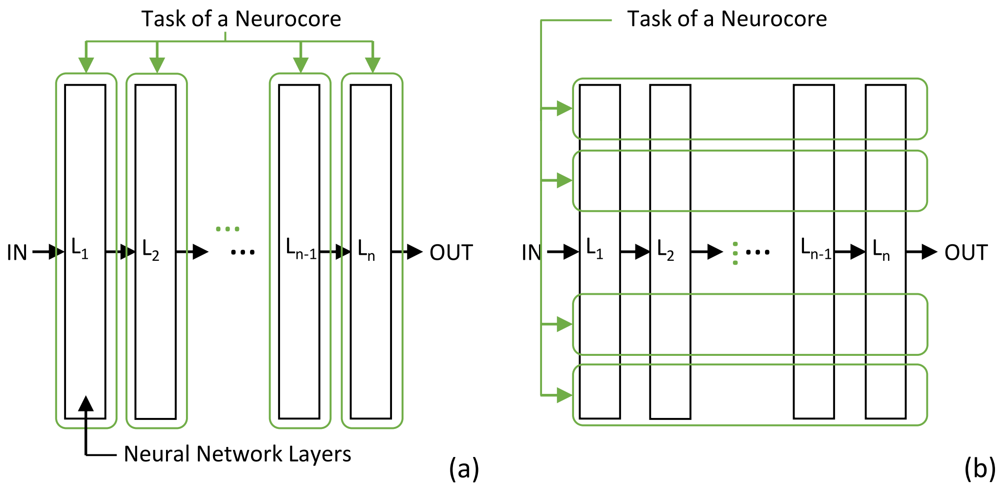

Fig. 6. Illustration of two different algorithms to neuromorphic hardware mapping strategies. (a) Parallelization along the depth of a neural network, in which case each neurocore computes a single layer. (b) Parallelization along the spatial dimensions of a neural network; in this case, each neurocore evaluates a subpart of every layers.

An intuitive mapping strategy is to have one core evaluate one layer, as depicted in Figure 6(a). With this solution, the computational efficiency of the neuromorphic chip is highly dependent on the network topology [153, 182]. For example, let us consider a chip composed of three identical neurocores that evaluates an FC network with a relatively small topology of 100-1,000-10 units. In this case, a single spike in input of the hidden layer requires 1,000 neuron updates. However, a single spike in input of the last layer requires only 10 neuron updates. Hence, if each layer receives an equivalent amount of spikes at each timestep, the computation charge is not distributed among the neurocores, the one dedicated to the hidden layer being the most active. Nevertheless, if the second layer receives 100 times fewer spikes than the third one, the number of updates realized by both neurocores is equivalent. Then, what becomes important is to make sure that the distance traveled by the spikes between neurocores is minimal. Therefore, depending on the layer organization among the available neurocores, their relative activities vary, as well as the number of packets sent through the NoC. In the example, we dealt with a relatively small network, but current deep networks can contain up to a million units, which ultimately increases the number of neurocores required to compute the full network, as well as the neurocore specifications in terms of both memory and computational capabilities. The first approach may thus not be an ideal solution.

A second mapping strategy would be to implement parallelization along the spatial dimension (as opposed to the depth) of the network. In this configuration, each neurocore will compute a subpart of a neural network layer, as described in Figure 6(b). That solution has the advantage of enabling per-layer time multiplexing of the whole neural network computation [2], thereby reducing the required number of neurocores to compute deep networks. Nevertheless, in the case of a convolutional neural network (CNN) [101], every neuron of a single layer shares the same weight values. This thus leads to memory redundancy between the neurocores and ultimately to a loss of efficiency.

This short case study reveals that correctly mapping the algorithm to the hardware is essential to fully exploit neuromorphic hardware capabilities. In real situations, researchers exploit a mix of both strategies to uniformly distribute the neurocores' activity. Some groups are thus working on algorithms to perform optimized mapping of any trained network onto neuromorphic hardware [37, 50]. Others exploit graph theory to maximize parallelization at the computation diagram level [2, 173].

We will further discuss accelerator implementation details and challenges in Section 4.

3 BRAIN-INSPIRED SNNS

In the early days of neuromorphic computing, the goal was to develop a novel kind of circuitry, which closely mimics the brain. The human brain consumes around 25W for 86 billion neurons [63]. Hence, neuromorphic computing paradigm was born.

We differentiate two purposes of brain-inspired computing. One is to leverage neuroscience discoveries to realize low-power machine learning or to improve performances of existing systems such as fault-tolerant hardware. Another is to implement on-chip bio-plausible dynamical elements to either accelerate neuroscience research or interact smoothly with real biological systems. The former is studied in Section 4, whereas in this part, we discuss some mechanisms observed in the brain that inspired SNNs, event-driven circuits, and sensors, excluding brain-inspired learning rules, which we discuss in Section 5.2. We detail brain-inspired neural networks and electronic circuits, and we discuss applications that ensue from it.

3.1 Computation in the Brain

The brain characteristics are different from traditional computers [206]. In short, it is highly parallel with interwoven computation and memory elements, deployed in volume, and operates asynchronously.

3.1.1 Neuron Models. Neuromorphic systems, more precisely networks of spiking neurons, are designed following a bottom-up approach. Building blocks, namely neurons and synapses, are designed and assembled into networks. The neurons are the computing elements of neuromorphic circuits. Several models describe the neuron functions, with a varying degree of complexity [52, 53, 75]. A large amount of them have already been transposed onto hardware [11, 20, 31, 73, 198]. The leaky integrate and fire (LIF) model has encountered a special interest among hardware designers [1, 15, 38, 122], and we thus discuss its working principle in the following.

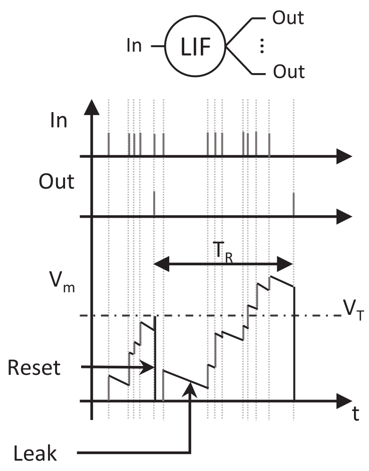

Fig. 7. Membrane potential temporal evolution of a typical LIF neuron. Several spikes are received asynchronously, the reset sets back the potential to zero, and the refractory period forbids the neuron to spike until enough time has passed between two spikes.

The temporal activity of a usual LIF neuron is depicted in Figure 7, and its behavior is described as follows. When a spike inputs the neuron (i.e., $x_i(t-1) = 1$), the synaptic weight w_i associated to this spike will be integrated on the membrane. When the membrane potential V_m overcomes a threshold V_t, the neuron fires (i.e., $y_i = 1$) and resets its membrane potential. Nevertheless, if the refractory time T_R is not reached (i.e., the amount of time since the last output spike is smaller than T_R), the neuron does not fire, even if its membrane potential is above the threshold. In addition, due to leakage, the membrane potential decreases continuously at the leak rate between two input spikes.

The essential parameters of an LIF neuron are the membrane threshold voltage, the reset potential, the refractory period, and the leak rate. At each timestep t, the membrane potential of neuron j of layer l can be described by

$$Vm_j^l(t) = Vm_j^l(t-1) + \sum_i w_{i,j} \times x_i^{l-1}(t-1) - \lambda, \tag{2}$$

where the λ parameter corresponds to the leak and $w_{i,j}$ is the synaptic potentiation. Note that models including complex internal mechanisms may be used to define these parameters instead of constant scalar values.

After integration of all incoming spikes for a given timestep, the potential is compared to the threshold and the output is defined by

$$\begin{cases} x_j^l(t) = 1 \; if \; Vm_j^l(t) > V_t \; and \; t - t_{spike} > T_r|, \\ x_j^l(t) = 0 \; otherwise \end{cases} \tag{3}$$

where t_{spike} is the last timestep at which the neuron j of layer l fired. Reset can be performed following different schemes: reset the potential to a constant value V_r or subtract the reset value from the current membrane potential. Rueckauer et al. [155] discuss the effect of both on final network accuracy. These units are strongly non-linear elements in themselves because their output is a succession in time of binary spikes.

The LIF neuron model is already a simple one with respect to biological dynamics [116], but it is not the simplest. Designers who do not target bio-plausible tasks can use abstracted models to reduce hardware resources, such as the IF neuron, a subset of LIF whose refractory and leakage mechanisms are removed, such as described in Figure 3. The necessary electronic building blocks to implement it are an adder for integration, a comparator for threshold detection, and a memory for membrane potential storage [15].

However, the bio mimicking is sometimes pushed further, with the hardware implementation of ionic channels and other bio-realistic components [15, 70, 148, 191]. These bio-mimicking circuits allow to emulate close-to-real dynamics and are expected to deliver new insights on neuronal function from the neuroscientific point of view. Bio-mimicry hardware can be realized using both digital and analog electronics design. We have seen some groups [31] implementing advanced reconfigurable units based on the work of Izhikevich [75] or bio-realistic ion channels [191] interaction in fully digital designs. SpiNNaker enables evaluation of complex neuron and synapse models but leads to heavy computational tasks [49]. Recently, Partzsch et al. [139] proposed an exponential approximation accelerator to improve the performance of the SpiNNaker neuromorphic processor. Nevertheless, most of the bio-mimicking hardware has been realized using analog electronics, mostly because complex mechanisms can be implemented with transistors biased in the subthreshold regime, where exponential behaviors are commonly observed [121]. Two major analog emulators are Neurogrid [15] and ROLLS [148], enabling real-time (non-discretized) and close to real neural simulation.

3.1.2 Synapse Models. Emitted spikes transit through synapses before reaching downstream neurons. ANNs consider the synapses as scalar values, the weight, which is different from the brain. A biological synapse receives spikes and in turn stimulates the membrane potential of post-synaptic neurons via a current, whose value evolves during time [44]. The synapse's weight, as described in formal networks, actually represents the long-term potentiation (LTP) phenomenon. The intensity of the excitatory current can reach various levels, which is determined by past long-term activity of the synapse. Additionally, another mechanism, referred to as short-term potentiation (STP), modulates the excitatory level according to the recent activity of the synapse [186]. It reduces the synapse's output intensity of each incoming spike arriving one after the other in a short period of time. Again, Neurogrid [15] and ROLLS [148] realize such synaptic behaviors.

3.1.3 Other Computational Elements. In addition to complex computational primitives, their organization and relations in the brain lead to higher-level mechanisms for performing operations. Software and hardware designers have taken inspiration from it to improve performance of deep learning algorithms in a top-down approach.

CNNs are probably the best illustration of this [101]. They are inspired from the hierarchical layering for visual recognition observed in the brain [68, 165] and have largely contributed to enabling image classification algorithms to reach human performance [157].

Winner-take-all (WTA) [117] is another brain-inspired mechanism implemented in inhibitory neural networks. It consists of having the units of a single layer compete with each other, where the one receiving the strongest input spikes and inhibits all the others. A variant is the kWTA, where each layer can have *k* firing neurons before full inhibition of the layer. WTA is a major factor of decision making [4, 159].

Neuron firing synchrony has been observed in the visual cortex and has been associated with dynamic interactions within the network [168]. Hence, not only is information encoded by individual neurons with rate or time coding, it also is communicated on a population level by the simultaneous activity of different units. We argue that artificial SNNs could capitalize on this observation to enhance their computational capability, especially for datasets where input timing is relevant to the targeted application.

Finally, a specific type of spiking networks, called *liquid state machines* (LSMs) or reservoir networks, fully takes its inspiration from the brain. Their topology is unlike traditional FC or convolutional networks, in the sense that there are not organized in layers. They are a combination of neurons randomly interconnected where each unit can receive spikes from both the input and the other neurons. Because of their recurrent nature, they are usually used for time dependent applications such as speech recognition. Given their complex computational graph representation, they may not map efficiently onto general neuromorphic hardware. Hence, some research groups are concentrating their efforts on designing neuromorphic architecture specialized for LSMs on FPGA [80, 162, 185] and other hardware support such as photonics [137] or 3D ICs [98].

To sum up, SNNs implement neuron and synapse models with various degrees of complexity, as well as possible higher-level brain-inspired functionalities. Complex networks require specific hardware to efficiently compute their evolution through time. Analog electronics can faithfully implement advanced neuromorphic operations, especially in the subthreshold regime, but digital realizations of bio-plausible mechanisms have also been proposed. Nevertheless, the use of bio-plausible models is not required for every application relying on SNNs. Machine learning may rely on simplified spiking units, such as LIF or IF models, to enable accurate and efficient hardware implementations.

3.2 Brain-Inspired Technologies

3.2.1 Autonomous and Robust Systems. The scope of applications resulting from bio-inspiration regarding fault-tolerance and adaptation mechanisms is large.

For what concerns adaptation, a famous resulting field is machine learning. Closer to the hardware, software training can also be used to adapt the program onto defective hardware. For example, high variability between novel nanodevices can be autonomously compensated for [108].

Several strategies for developing resilient electronics have emerged, based on the impressive resilience against noise, fault, or damage of biological systems. For instance, the principles of evolution and natural selection gave rise to evolutionary hardware for implementation of reconfigurable fault-tolerant systems [10, 58, 151]. Closer to the brain, we have seen researchers exploit neuroplasticity mechanisms to realize efficient fault-tolerant and reconfigurable systems [83, 109, 110, 166]. Neuroplasticity refers to the ability of the brain to adapt and reconfigure during lifetime operation. It involves connectional structure modification, synapse adaptation, and others [143]. Its implementation onto hardware is thus relatively complex but may allow to remove the need for redundant hardware in critical electronic systems. Some even take inspiration from higher-level brain mechanisms, such as auto-encoder networks, to realize noise-resilient circuits [90].

Hence, bio-inspired computing is not limited to machine learning and already benefits to the implementation of fault-tolerant systems.

3.2.2 Event-Driven Sensors. On top of circuits whose organization and computation schemes are bio-inspired, sensors mimicking the biological senses have also been implemented. Among them are dynamic vision sensors (DVSs) [107, 145] and audio-recording systems such as the cochlea circuit [32]. They output the data in the form of spikes, using the AER protocol, and their sensing principles are based on bio-plausible mechanisms. Their main advantages over frame-based sensors are their fast response time and important dynamic range [107]. Detailed information on their implementations and working principles can be found in other works [105, 111].

4 NEUROMORPHIC MACHINE LEARNING ACCELERATORS

Neuromorphic systems for machine learning applications have recently become a major subject in computer design, for both ANN and SNN acceleration. ANNs evaluation being computationally

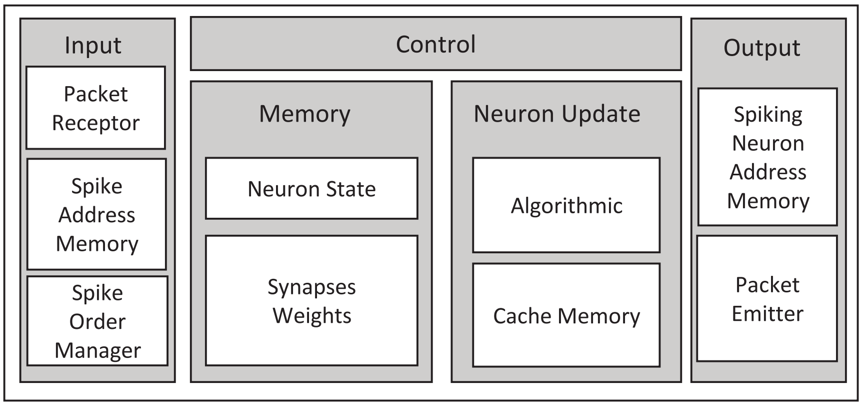

Fig. 8. Schematic representation of the neurocore components.

intensive, hardware for fast and low-power evaluation of ANNs, such as the tensor processing unit [85], are currently being developed. In parallel, SNN accelerators are becoming essential for the evaluation of SNNs. Indeed, their simulation requires sequential computation of several timesteps, and simulation time thus scales with the deepness of the network and the time resolution of the simulation. As discussed in Section 3, hardware implementation of SNNs enables the emulation of more or less bio-plausible neurons and synapse models, which may help in acquiring more knowledge of the operation of the brain. Machine learning application is also done with SNNs, but their simulation takes a large amount of time on traditional computers because of the time discretization. To such use, SNN hardware implementation may leverage properties of SNNs to reduce power consumption and increase evaluation speed, especially when using hardware-friendly models of neurons and synapses, such as the simple IF neuron and constant weights.

We discuss here the neurocore organization in detail, the communication scheme between cores with binary spikes, the sparsity of SNNs, how the fault tolerance of neural networks can be taken advantage of, and finally the implementation of SNNs with crossbar arrays.

4.1 Core Organization for Low-Power SNN Evaluation

As discussed in Section 2.3, a neuromorphic processor is a chip composed of one or several neurocores. A single core integrates PEs to evaluate the neuron membrane potential, memory to store synaptic values and neuron states, input and output interfaces to receive and emit spikes, and control circuitry. The number of neurons computed per neurocore relies on the size in memory of the parameters required per unit. Designs targeting very large scale applications usually implement a large number of neurocores to maximize parallelization at chip level [15, 38, 49, 122, 148, 160]. A large amount of publications report various neurocore organizations with designs optimized in accordance with the characteristics of the network topology to be mapped to the hardware [28, 91, 130, 164, 193] or with a targeted application [34, 38, 103, 148, 153, 192, 194, 201].

Different network topologies require different hardware designs to fully take advantage of neuromorphic computing. For example, a shallow FC network does not have the same algorithmic footprint as a deep CNN, nor the same memory requirements [181]. However, given the large spectra of existing network topologies and their constant evolution, we classify neurocore designs following a hardware-wise, non–network-specific distinction: analog or digital design. To implement the PEs and memory using analog or digital electronics is a designer choice based on the targeted applications and performance. The goal of neuromorphic implementations may be to emulate precise bio-inspired neuronal computation, to compete for the lowest power or highest accuracy on a specific machine learning benchmark, or to enable very large scale neural simulation at low cost, both in time and power.

In Section 3, we discussed analog implementation and concluded that it is well suited for the realization of complex bio-plausible tasks [148, 191]. Nevertheless, machine learning using such circuit designs is still possible, with their main advantage being the ideal co-localization of memory and computation [15, 72]. Qiao and Indiveri [147] recently reported hardware for both real-world interfacing or SNN acceleration, using mixed signal digital-analog circuits whose speed can be adapted to application requirements.

Reported digital designs usually employ a single PE that updates every neuron mapped to the core [28, 34, 38, 122, 130, 153, 167, 193, 194]. For each algorithmic timestep, the neurocore runs through the following procedure. First, every upstream spike received during the previous timestep, which is stored in an event queue such as the spike scheduler [153, 193] or FIFO register [28, 115], is processed sequentially. Each spike is an address that allows to identify the downstream neurons according to the connection map of the part of the SNN processed by the neurocore. Thus, every synaptic weight of downstream neurons, as well as neuron state parameters, is loaded from the core memory. The update of each neuron is then done sequentially in a time-multiplexed manner. A neuron update consists of adding the value of the weight associated to an input spike to the neuron's membrane potential, after which the membrane potential is compared to the threshold of the neuron, as depicted in Figure 3. As soon as a neuron's membrane potential crosses its threshold, an output spike is emitted onto an asynchronous network, its membrane is reset, and the neurocore operation proceeds until every spike from the previous timesteps is processed. One interest of SNNs is thus that an operation is reduced to an addition and a comparison [193]. Hence, digital implementation only requires adders, which significantly reduces computation cost when compared to MAC operators of ANNs. However, because time is a primitive in SNN computation [116], both timing information (for on-chip learning, leakage and refractory period mechanisms) and neuron state variables (membrane potential, threshold) have to be stored, which requires supplementary memory [38, 74, 201]. Depending on the network topology, weight access quickly becomes a bottleneck given the possible large number of units and connections; there are around 10,000 synapses per neuron in deep networks. To reduce the memory impact, several strategies can be followed: its distribution inside neurocores, the use of hierarchical memory to pre-load data, and the employment of a new kind of non-volatile memory (NVM) device [25].

A problem of time multiplexing the neuron updates is that a single algorithmic timestep takes several clock cycles to complete. The more the neuron model is complex, with leakage, refractory period, and so forth, the more the number of clock cycles will be important. Hence, some groups propose to smartly reduce computation cost by updating only what is necessary. For example, Roy et al. [153] implemented LIF neurons but realized "leak-upon-load" so that leakage operation was not realized every timestep for all neurons. However, it requires to store the last time the neuron was loaded. Yin et al. [193] compressed their network, resulting in a small number of neurons per layers. They thus parallelize unit updates and remove time-multiplexing requirements, enabling them to perform voltage and frequency scaling to reduce power consumption while keeping their targeted throughput. Note that voltage and frequency scaling can be employed independently of network compression at the expense of inference speed [64, 96]. Ma et al. [115] physically implement a few neurons and still perform time multiplexing; as such, they create a tradeoff between hardware requirements and throughput reduction.

Many other design choices can help improve performance. The synchronous operation but asynchronous communication between neurocores requires the presence of event queues to receive the incoming spikes asynchronously during an algorithmic timestep. As such, spike schedulers [153, 193] are powerful tools. They allow organizing the sequence of upstream spikes to be delivered to downstream neurocores to reduce potential redundant memory accesses. Roy et al. [153] also interestingly share them between several neurocores. This may maximize the efficiency of topology mapping, such as by allowing a single spike scheduler per SNN layer, even for networks with large layers.

At a higher scale, neuromorphic chips, taking inspiration from the parallel, highly distributed brain architecture, are commonly composed of several neurocores, communicating together via a NoC [48, 74]. Very large scale designs even implemented multi-chip boards for evaluation of deep networks [48]. This parallelization at every scale is possible thanks to the scalability of the AER protocol. It allows distributing the computing task in several smaller computations and by doing so accelerates the overall network evaluation. Moreover, one could program the chip so that each SNN layer is computed within one neurocore or cluster of neurocores spatially close together on the network, to pipeline the overall network evaluation and fully take advantage of the bio-inspired hardware design.

To sum up, neuromorphic design allows to implement parallel SNN evaluation. Given the amount of properties that contribute to power consumption, throughput, and chip size, we argue that accelerators must respond to their target application. A processor designated to training or evaluation or both of very large scale SNNs should concentrate on maximizing the flexibility and speed, without necessarily focusing on power consumption and chip size. However, an accelerator designed to target embedded applications should focus on power consumption and area requirements and should be designed in close consideration of the SNN algorithm to be evaluated. The neurocore architecture is an essential component, but other tools, such as the AER protocol and fault-tolerance exploitation, improve neuromorphic chip performance.

4.2 Reduced Communication Cost

Multicore parallelism of SNN accelerators relies on a specific NoC communication protocol to transmit events between a large number of neurocores at minimal power and delay. The AER is always proposed in neuromorphic systems. It enables to naturally encode the event time and organize the connectivity at low communication cost [19, 111]. It consists of sending a unique packet containing the address of the spiking neuron on a digital bus with asynchronous logic. As soon as a neuron fires, its address is sent onto the NoC, and the firing time is encoded in real time on the asynchronous bus.

When compared to the frame-driven approach of ANNs, the benefits of the AER protocol for large-scale SNN computing are numerous [111]. First, Boahen [19] has shown that it allows to reduce the network bus size while conserving a large connectivity capacity. AER NoC area requirements are thus minimum and enable large-scale design. In addition to the transmitted packets being reduced to a single address, it also guarantees small delay and power (due to switching activity) overheads. Moreover, the neuron activity sparsity of SNN trims the NoC activity, reducing the number of packets sent on the network.

This communication protocol is easily scalable. As long as the routers can manage the requests, any number of units can be connected together. It is thus suitable for multi-chip implementation as well. However, the power consumption of such a system grows with the number of units to connect, the mean firing frequency of each unit, and the overall chip size [37, 138]. Some works target to reduce the power and delay overheads of AER by exploiting the locality and clustering of neural network algorithms [19, 113]. Another possibility is to employ hierarchized router levels, which helps in reducing the power impact of large-scale systems [72, 113, 138]. Rate or time coding have significantly different impacts on such overheads. Using rate coding, a larger amount of spikes, with respect to time coding, is sent through the network and the overall system power consumption drastically increases, as shown in Mostafa et al. [130]. However, rate coding usually employs Poisson spike trains, and an individual spike timing error is thus of small consequence for the precision of the network. It is different for time coding, where timing matters. Timing error or information loss may occur when the algorithmic timestep duration does not allow the neurocore to complete its operation. In this situation, spikes may be missing or even dropped from the network. Nonetheless, it can be used as a tradeoff between accuracy and throughput or power [34].

Finally, the AER protocol is compatible with various NoC routing architectures and broadcast schemes, such as 2D mesh [38, 49, 122], multicast tree [15], or systolic ring [91, 96, 153]. It is thus possible for the hardware designer to adapt the routing scheme employed to best fit the requirements of the accelerator.

Therefore, the AER protocol allows low power and footprint NoC designs. With the packets being reduced to a single address, the communication is very efficient. AER can be adapted on several types of NoC, with various broadcast schemes. In addition, thanks to the intrinsically important sparsity of SNNs, only the necessary information is transmitted.

4.3 Leveraging High Sparsity of Dataflow

Many argue that the main advantage of SNNs is that for a given input data, due to the event-driven nature and the thresholding of every unit, the evaluation will usually not require every neuron of every layer to fire [130]. As such, computation is sparse and the evaluation of the network on the input data requires fewer operations compared to the ANNs' frame-driven approach, where evaluation of every neurons of the network is required for every input data. Here, we describe the sparse operation of SNNs and discuss whether it has an advantage over ANNs.

4.3.1 On the Sparsity of SNNs. ANN accelerators employ guarding strategies to exploit the sparsity of the network activations [127, 189], which requires supplementary hardware to check the nullity of the data. However, when evaluating an SNN on event-driven hardware, the sparsity inherently reduces the switching activity. Indeed, due to the thresholding operation, a neuron with a small activation does not fire.

Moreover, if an event-driven sensor generates the input, the number of computations is adapted to the reality requirement [22, 107, 145]. Event-driven sensors are fully dynamic sensors that emit spikes to encode the detection of a sensed phenomenon. They are opposed to frame-driven sensors, which deliver a fixed amount of data at a specific sampling frequency. Hence, for a DVS, only the portion of the frame where some movement occurs communicates data, which is less power consuming than a traditional frame-driven sensor [107].

However, SNNs do not require an event-driven sensor input to perform. If we want to benchmark an SNN against a conventional formal dataset (e.g., MNIST [101]), the dataset first requires a conversion to spiking information. For visual applications, the presented solutions are (1) conversion of intensity to a Poisson spike train following an arbitrary conversion scale [29, 39]; (2) applying a continuous analog value (voltage or current) on the input neurons' membrane potential, which will be continuously integrated [155]; (3) emitting a single spike per pixel whose latency is inversely proportional to the intensity [88, 154]; and (4) observing the dataset with a dynamic sensor to convert it into spikes [135]. Depending on the chosen conversion model, energy consumption and accuracy statistics will vary.

At the output, we also need to understand the information delivered by the spiking units, and this it is necessary to set an output evaluation scheme [154]. Depending on the information coding scheme, several methods to decide which output neuron represents the output are available. For example, to implement an image classifier using TTFS coding, the class can be represented by the first unit to spike, and, using rate coding, by the neuron with the highest firing rate, or the one with the highest membrane potential. Some works even add a classification layer at the output to make a decision [202].

4.3.2 SNN and ANN Comparison. An important ongoing question with SNN processors is their relative performance to ANN accelerators. A major issue when comparing the ANN and SNN comes from the input data conversion [198]. Indeed, an SNN requires a number of algorithmic timesteps to evaluate the output, during which the input should be sent continuously. Thus, the feedforward operation of the spiking network must be realized several times, whereas the formal one is evaluated only once. Under these conditions, Han et al. [59] have shown that the more complex the dataset, the more the energy per classification of the SNN increases with respect to its formal counterpart. However, one may consider that over the same period of time, an ANN would have computed the output a certain number of times depending on the sampling frequency of the input sensor. In this case, the ANN would be disadvantaged with respect to the SNN because the total number of MAC operations realized by the ANN is proportional to the number of times the ANN evaluates its output. We know that iso-accuracy between an SNN and an ANN on MNIST is possible for a similar network topology [155], but no information is given concerning the relative number of operations. In addition, to the best of our knowledge, employing SNNs on more complex formal datasets such as CIFAR-10 or ImageNet delivers worse accuracies than a formal evaluation [155]. Hence, so far, SNN classification on formal visual datasets delivers lower accuracy results than their formal counterparts.

However, if one wants to evaluate a formal network on an event-driven dataset, it is necessary to convert continuous spiking information to a frame or a succession of frames. Lee et al. [104] performed the conversion of the event-driven N-MNIST dataset [135] by combining the spikes during one tenth of the full event stream. Hence, for each input stream of 300ms, they had 10 frames representing the events over 30ms. They trained an SNN and an ANN with the same topology on N-MNIST and converted N-MNIST, respectively. In spiking, they achieved 98.66% accuracy on digit recognition, whereas their formal network reached 97.8%. Moreover, their SNN allows this 0.8% difference in accuracy for four times fewer operations. In this case, it appears that SNN performs better than ANN.

Thus, spiking hardware is intrinsically sparse in its activity, which makes it an ideal candidate for low-power machine learning applications. However, to compare the performance of SNNs to the ones of usual ANNs is not trivial because it requires the conversion of datasets, which biases the results. In short, SNNs achieve better performance both in terms of accuracy and computation cost on event-driven datasets, whereas ANNs achieve better performance on formal datasets.

4.4 Exploiting Fault Tolerance of Neural Networks

Approximate computing has been used for a broad range of applications to reduce the energy cost of computation in hardware [60]. Two main approximation strategies are used with neural network applications, namely network compression and classical approximate computing.

Due to the large amount of parameters representing the neural networks, researchers targeting embedded applications started to reduce the weight and activation map precision to decrease the memory footprint of ANNs, a method referred to as network compression or quantization. Thanks to the fault tolerance of neural networks, as well as their ability to compensate for approximation while training, reduced bit precision entails only a small loss in accuracy [36, 61, 67, 149]. Once transposed to hardware, weight quantization (WQ) has shown 1.5 to 2 times gains in energy with an accuracy loss of less than 1% [127, 189]. An aggressive quantization to binary neural networks (BNNs) allows to use XNORs instead of the traditional costly MACs [149]. An interesting implementation leveraging both the parallel design of the crossbar array and the XNOR-net implementation is realized in Sun et al. [174]. They report 98.43% accuracy on MNIST with a simple BNN. Quantization is thus a powerful tool to improve energy efficiency and memory requirements of ANN accelerators, with limited accuracy loss.

These methods can be used for SNNs as well [79, 150]. Indeed, Rathi et al. [150] report from 2.2 to 3.1 times gain in energy, and even more in area, for an accuracy loss of around 3%. What is interesting with WQ is that the designer can realize a tradeoff between the accuracy of the SNN application against energy and area requirements of the neural networks. Approximate computing can also be achieved at the neuron level, where insignificant units are deactivated to reduce the computation cost of evaluating SNNs [163].

Moreover, such computation skipping can be implemented at the synapse level in a random manner. Indeed, training ANNs with stochastic synapses enables a better generalization, which results in a better accuracy on the test sets [172, 183]. Again, this method is compatible with SNNs and has been tested both during training [133, 171] and operation [23], and even to define the connectivity between layers [14, 34]. FPGA [167] and hardware [77] implementations of spiking neuromorphic systems with synaptic stochasticity shows that it allows to increase the final accuracy of the networks while reducing memory requirements. On top of that, nanoelectronic devices with intrinsic cycle-to-cycle variability, such as memristors [97] or VO_2 [77], allow to reduce area and power consumption overhead of random number generation. Chen et al. [34] also leverage probabilistic rewiring to increase their throughput. Fewer synapses imply fewer pulses to integrate, and the algorithmic timestep can thus be increased. Doing so, they report an acceleration of 8 times with an energy gain of 7.3 times for an accuracy loss of only 0.25% on MNIST digit recognition, from 98.15% to 97.9%. Stochastic and quantized synapses can thus drastically reduce the memory requirement and power consumption of SNN accelerators, and even more with pruning insignificant weights.

The other approach consists of designing PEs that approximate their computation by employing modified algorithmic logic units [60]. Kim et al. [94] have shown that using carry skip adders enables speed and energy gains of 2.4 times and 43%, respectively, when evaluating SNNs onto neuromorphic hardware for character recognition, with an accuracy loss of only 0.97%.

Thus, approximate computing methods, both at the software and hardware levels, can enable important gains in power consumption and speed. However, as the complexity of the dataset increases, along with the depth of the network topology, such as using the ResNet [62] on ImageNet [157], the accuracy loss becomes more important and may not be negligible anymore [149], especially for critical applications such as autonomous vehicles. It is thus unsure whether network compression techniques and approximate computing are scalable and applicable to any task.

4.5 Crossbar Array With Memristive Devices

Neuron connections inside the brain are organized in three dimensions, allowing very dense and highly parallel networks at the smallest scale. Integration of neural networks on silicon cannot reach similar density, being mostly 2D. Nevertheless, many groups are working on the most parallel implementation: the crossbar array [7, 24, 25, 71, 205]. Such design aims at combining the memory and neuron update parts of a neurocore (Figure 8) inside a single unit, thus enabling speed and energy gains [5, 7] and truly implementing non–Von Neumann computing. It consists of having two metal lines crossing one another in an orthogonal fashion (Figure 9), with a nanoelectronic device mimicking the synaptic behavior set at each cross-point intersection. One direction represents the output of the pre-synaptic neurons, and the other direction represents the connected postsynaptic neurons. As such, the operation of the analog crossbar array consists of applying voltages over the input lines and reading the current of the corresponding output line. With each device's conductance (the inverse of its resistance) symbolizing the synaptic weight of a connection, the resulting currents add up together following the Kirchhoff laws, and the dot product is realized [25].

To enable high-density crossbar implementation, without accuracy loss, the devices have to (1) be small, (2) require low power for read and write operations, and (3) be stable [25]. PCM [92,

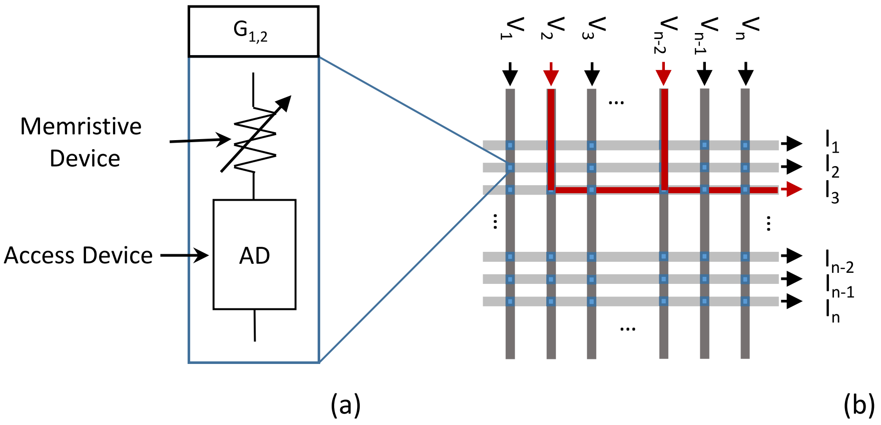

Fig. 9. Representation of a crossbar array implementation. (a) Description of a memristive device implementation. (b) Top view of a typical crossbar design, with input along the vertical lines and output along the horizontal lines.

175] and metal oxide resistive devices [51, 119, 146] are good candidates for (1) and (2) because their power consumption decreases along with their size. However, at a small scale, they do not meet the third requirement yet, which drastically limits their use as neural network algorithm accelerators. Their imperfections, namely weight update non-linearity, different G_{min} and G_{max} values, different conductance variations for identical programming input, resistance drift, and others, limit the accuracy of a neural network mapped onto it [26, 76, 93]. Hence, several groups focus on making better memristive elements [175] to fulfill the requirements, or on adapting the circuits architectures to compensate for the imperfections of the devices [17, 26, 176]. For more details regarding the working principle of the memristive elements and other types of novel devices for synaptic application, we refer the reader to Burr et al. [25] and Basu et al. [11]. Some groups foresee emulating synaptic behaviors with magnetic micro- and nanodevices, such as spintronic memristors [136, 184] or magnetic tunnel junction synapses [171]. Up to now, however, most studies are presenting results obtained from simulations with theoretical models of the devices.

The crossbar array implementation is still at the state of research because of memristive element issues, but recent works already demonstrate impressive results. Ankit et al. [7] realized an SNN simulation at the layout level of a full neuromorphic architecture. They announce large gains, both in terms of energy and speed, when employing the crossbar array with respect to the same neuromorphic architecture designed with classical neurocores. They reach gains of several hundreds when simulating FC networks and several tenths when evaluating CNNs, which confirms that the crossbar gain is highly dependent on the network topology. More recently, Ambrogio et al. [5] claim reaching equivalent accuracy on ANN evaluation, with respect to an ideal simulation, using a PCM crossbar array externally controlled by a computer via a prober. They announce a potential power efficiency increase of one to two orders of magnitude with respect to standard CPUs or GPUs.

So far, what we said applies to both formal and spiking models. Nevertheless, an advantage of SNNs over ANNs is the fact that the activations are binary. Indeed, the voltage applied at each input of the crossbar is thus the same, which simplifies the surrounding circuitry. The last step, neuron update, can be realized in analog electronics or with digital circuitry. An analog implementation can reach very high throughput and very dense implementation, ideally with a capacitor at the output of each line [92], especially if the memristive elements are directly placed between two access lines. However, because of lack of control with cross-point designs [106], crossbars are usually implemented with access devices, which drastically reduces the design density. A digital implementation of read-out circuitry will require at least an A/D converter and memory to store the neuron states [95, 123]. Neuron updates can once again be time multiplexed to reduce the hardware requirement, in which case the crossbar array may still be advantageous in that the computation happens in memory, but the designer must then be careful with regard to the costs of transfers to and from neuron state memory and the A/D conversion. However, ANN operation with full-precision activations on such designs requires either simplification of the network topology [177] or advanced circuitry to deliver accurate voltage values to each input lines [3] and to apply non-linearity after the MAC operation.

To conclude, crossbar arrays with NVM devices are promising in terms of performance and scalability, especially in the case of a fully analog implementation, where neuron update is realized in a parallel fashion. It already shows good results but still requires improvements to guarantee reliable, fully on-chip, and lasting operation life.

5 THE DIFFICULTY OF TRAINING AN SNN

For a long time, retro-propagation of the gradient in SNNs was not realizable due to the non-continuity of the equations of spiking neurons. In addition, training an SNN requires further parameter optimization with respect to an ANN because thresholds, leak rates, and so forth are sensitive values. To train SNNs, several solutions were thus proposed: converting a trained ANN into an SNN, performing bio-inspired unsupervised learning, or developing some other supervised algorithms.

5.1 Study of Conversion ANN to SNN

At first, ANN to SNN conversion was not trivial because spiking neurons behave quite differently than formal perceptrons with non-linear activation. Several works proposed conversion frameworks as explained elsewhere [29, 39, 40, 141, 155]. For a long time, the conversion from ANN to SNN required modifying the topology of the formal network [29, 39, 141]. However, it is now possible to design SNNs that benefit from techniques such as batch normalization and inception layers [155]. Rueckauer et al. [155] achieve the best results in terms of accuracy on usual visual benchmarks using spiking networks but are still behind the state of the art with respect to formal networks.

These transduction techniques usually implement rate coding to transmit activation map values. Such coding reduces the power efficiency once the network is implemented onto hardware. Moreover, the efficiency loss scales with the dynamic range, which is rather large due to floating-point representation used in conventional ANNs. To deal with this issue, Rueckauer and Liu [154] recently implemented a conversion using time coding neurons. However, this lower-power implementation is at the expense of final accuracy and applicability of the conversion method to various ANN topologies.

As discussed in Section 4.3, the difference in accuracy between ANNs and SNNs may come from the incompatibility between the datasets and the spiking operation. Nevertheless, one cannot rely only on formal to event-driven conversion to enable SNN training, especially for event-driven datasets, onto which a formal network could not be trained. Hence, a large amount of effort has been put into direct SNN training.

5.2 Unsupervised Learning: Autonomous Extraction of Pattern in Datasets

The bio-inspiration and the supposed/anticipated consequent effectiveness of unsupervised learning led consideration of the spike timing dependent plasticity (STDP) algorithm as a promising mean of training an SNN [30, 120, 170]. STDP is a Hebbian learning rule where the synapses' weight update depends only on the relative timings of pre- and post-synaptic neuron spikes, as

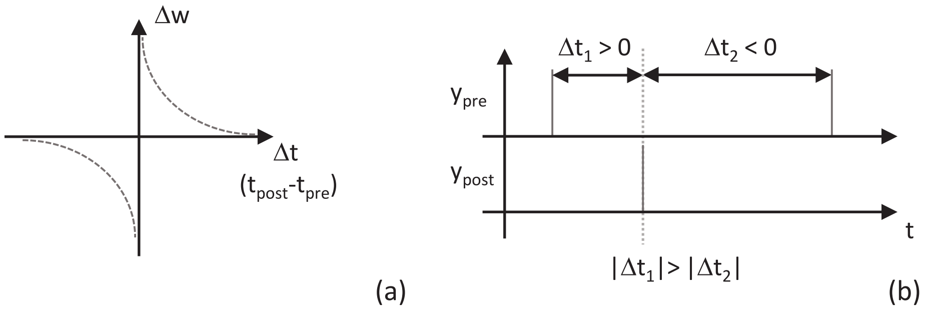

Fig. 10. STDP rule illustration. (a) Diagram of the relation between weight update Δw and relative timings of post- and pre-synaptic spikes Δt. (b) Chronograms illustrating the spike timings and associated weight updates.

depicted in Figure 10. If the post-synaptic neuron fires shortly after the pre-synaptic one, then a causality is expected between the two, and hence their synaptic connection is strengthened. Inversely, if the first to fire is the post-synaptic one, then the relation is weakened. However, unsupervised learning via STDP does not allow multi-layered networks to reach high accuracy [131]. That is mostly due to the close locality of Hebbian learning, where each layer adapts to the output of the preceding layer, without coordination between them to specify the transformation input to output of the full network. Thus, different strategies have been used to improve the accuracy of STDP-trained networks: (1) adding supervision to the training, such as reinforcement learning [45, 131, 132]; (2) pre-processing the input data [87, 88]; and (3) modifying the learning rule or the layer communication scheme [78, 179]. These strategies allow increasing the performance on classification tasks, but they do not reach the state of the art with respect to both SNNs trained with supervision and usual ANNs trained with backpropagation.

Moreover, researchers are discussing the fact that a trained neural network is not able to extract semantics from the learned data but only statistically frequent features [82]. As a confirmation of this, Mozafari et al. [132] have shown that STDP allows to extract features occurring frequently, but not the ones specific to the targeted application. Therefore, even if Kheradpisheh et al. [88] could train a convolutional SNN to an accuracy of 98.4% on MNIST, using pre-processing layers and a simplified version of STDP on the convolutional layers, it is unclear whether unsupervised learning in itself is suited for classification tasks.

We argue that such learning rules should be considered as a research tool or pre-processing method to autonomously extract relevant patterns in complex inputs or to get a better understanding on what to focus on. For example, Lee et al. [102] employed a two-part SNN training algorithm. They use STDP to initialize the weights of the networks, and then employ classical backpropagation. They show that doing so allows to improve the network accuracy with respect to a random initialisation. The activation values of each neuron used for the gradient descent algorithm are obtained by integrating output spikes over time. Such methods could thus facilitate exploitation of complex data where spatio-temporal or spectro-temporal features are essential, such as spike sorting [16, 57, 188] or artificial finger haptic [152, 203].

5.3 Supervised Learning: On the Road to Event-Driven Machine Learning

Although unsupervised learning is not efficient enough for high-accuracy SNN training, the task of finding an efficient supervised learning rule remains. Several SNN-specific learning algorithms or techniques [21, 45, 46, 126, 144, 178, 197] have been proposed, among which are the well-known ReSuMe [144], SPAN [126], and Chronotron [46]. These works follow different paths: (1) some are based on STDP and enable supervised training [45, 144], (2) one analytically computes the weights of the network [178], and (3) others approximate the behavior of spiking neurons to enable gradient computation from an error function [126, 197].

Bohte et al. [21] studied the possibility to backpropagate the gradient in SNNs, and an increasing amount of recent publications refer to its implementation [42, 69, 81, 104, 129, 190, 193, 199]. What limits backpropagation with spiking units is the non-derivable mathematical expression of neurons and synapses. Indeed, spikes are discontinuities and are thus non-derivable. To overcome this difficulty, several research groups explore various approaches. Lee et al. [104] simply consider that non-continuity is negligible and show that ignoring it has only a small impact on final network accuracy. Esser et al. [42] introduce side functions derivable to enable backpropagation without modifying the non-continuous units. This strategy is similarly used by Zenke and Ganguli [199]. Others [129, 193] exploit the continuity of models of neurons and synapses relying on time coding to enable derivation and computation of the gradient. Finally, some groups, such as Huh and Sejnowski [69], design differentiable models of networks to enable gradient evaluation. These works show good accuracy on the MNIST and N-MNIST datasets, which are the more common datasets for SNN evaluation but are still behind the conventional ANNs for similar network topologies. However, Kulkarni and Rajendran [99] recently showed training an SNN with better accuracy than its ANN counterpart, but only the final layer was trained with a spiking model. Jin et al. [81] also showed impressive statistics on these datasets by implementing backpropagation throughout a full SNN. They evaluated local gradients, at the "micro level," from the train of spikes of each neuron and regard these spike traces as some sort of activations, which allows them to realize layer-to-layer gradient descent at the "macro level," in a fashion similar to the classical formal approach. It allows them to achieve an accuracy of 99.49% on MNIST with a convolutional spiking network, which is, to the best of our knowledge, the best result demonstrated on this dataset with an SNN.

Interestingly, Wu et al. [190] introduced a framework for backpropagating the gradient both along the depth and time dimensions of a spiking network, which allows the training to take into account the evolution of the data. This algorithm may enable an SNN to develop a better understanding of event-driven datasets by including the time dimension of the data as part of the searched patterns.

Some argue that using less bio-plausible models may allow deriving efficient learning rules while leveraging the sparse activity of spiking neurons [112]. Such philosophy is also present in the work of Yin et al. [193]. Indeed, they propose two different models of neurons, each adapted to tasks requiring computational elements with different degrees of bio-plausibility, and use them for different tasks. They propose two models of neurons allowing them to implement backpropagation of the gradient, a simple one that does not use time as a computational primitive and another model that does.

To sum up, supervised learning of SNNs, more specifically backpropagation of the gradient, is starting to enable spiking networks to reach accuracies equivalent to the ones of formal neural networks. Some models are hardware friendly and already enable low-power and high-accuracy inference [193]. In addition, some researchers [190] are working on solutions to include time evolution in the evaluation of the gradients for backpropagation, which allow to train the networks on event-driven datasets without any conversion required. Hence, it may be a matter of time before the SNNs finally deliver their full potential.

5.4 On-Chip or Off-Chip Training?

Whether to implement training on-chip or off-chip should be related to the considered application. If the objective is to design a general accelerator for machine learning, obviously the chip should allow on-chip training [24]. Now, if the purpose is to perform a unique machine learning task on embedded low-power hardware, an off-chip learning, which potentially is power consuming, can be realized only once, after which the resulting network is programmed on the chip. At that point, one could argue that, in some cases, the system should need to adapt to its sensing environment while operating, which is referred to as on-line learning. One solution is to enable off-chip training in between operation times and update/fine tune the system during inactive or loading time. However, this last statement still brings a lot of drawbacks—for example, it requires to add a memory that would save the input data acquired during operation. In addition, on-line learning is still being hugely researched because machine learning currently has the major drawback of catastrophic forgetting—that is, a trained network cannot learn a new task without losing accuracy on its previously learned task [8, 47, 55].

The locality of STDP is an argument in favor of on-chip learning, because little or no information is transmitted between neurocores to enable weight update based on spike timing. The neurocore can store, with a counter, the time at which the pulse arrives and compares it to the firing time of the post-synaptic neuron. However, on-chip STDP requires specific circuitry [92], and/or addition of memory to the neurocore, which may reduce power efficiency and increase the chip area. Nevertheless, some groups work on modified STDP rules to reduce its computational cost [78], and we have seen some demonstrations of on-chip STDP implementation with small hardware overhead [114, 201], which may still contribute as an argument in favor of embedding SNNs.

Finally, if we want to apply supervised learning, these algorithms require either complex neurons and synaptic models [126, 144, 197] or floating-point values communication of gradient between layers, and thus between neurocores [104, 193], which makes their hardware implementation impractical. Moreover, if weight update is performed on-line (i.e., during inference), feedforward operation must be paused for learning, which adds an operational delay to the system [201].

6 HARDWARE IMPLEMENTATIONS: COMPARISONS AND DISCUSSIONS

The question of how to compare neural network accelerators remains unanswered. We believe that the comparison should be hierarchized in the following way. First, the type of neural network to be accelerated (i.e., ANN or SNN) should be differentiated to avoid biasing one versus the other, as discussed in Section 4.3. Then, one must consider the application. An embedded accelerator, with a restricted power budget, cannot be compared to a general deep learning processor. They do not have the same constraints in terms of area, power, flexibility, and so forth. Finally, one should decide which criteria are relevant for the targeted application.

For example, a general machine learning accelerator focuses on speed of operation, flexibility of design to adapt to various topologies, and power to enable very large scale algorithm evaluation. However, an embedded processor should focus on power consumption, accuracy of the mapped network, and die area. Both devices perform the same operation, but their main constraints are different. In the next sections, we give an overview of both general SNN accelerators and application-specific small-scale SNN processors.

6.1 General Neural Network Accelerators

Evaluation of large-scale neural networks of various topologies requires the accelerator to be highly configurable. The well-known large-scale architectures TrueNorth [122], Neurogrid [15], BrainScaleS [160], Loihi [38], and SpiNNaker [49] adopt different characteristics to emulate networks of spiking neurons. We refer the reader to Furber [48] for an advanced analysis of these designs.

SpiNNaker [49] uses a network of CPUs tightly connected to local memory on a single chip. It is the most reconfigurable of the four, but it is not as energy efficient, or as fast, as the others, especially if complex models of neurons and synapses are simulated. BrainScaleS [160] is

Table 1. Big Ones Characteristics

Processor	BrainScaleS [160]	Neurogrid [15]	TrueNorth [122]	SpiNNaker [49]	Loihi [38]
Implementation	Analog	Analog	Digital	Digital	Digital
Time	Discretized	Real time	Discretized	Discretized	Discretized
Neuron Update		Real time	Time MUX	Time MUX	Time MUX
Synapse Resolution	4b	13b shared	1b	Variable	1 to 64 b
Bio-Mimicry	Not configurable	Not configurable	Limited to LIF	Configurable	Configurable
On-Chip Learning	STDP only	No	No	Yes	Yes
NoC	Hierarchical	Tree multicast	2D mesh unicast	2D mesh multicast	2D mesh unicast
Neurons per Core	8 to 512	65e3	256	~1e3	max. 1,024
Synapses per Core	~130k	100e6	65k × 1b	~1e6	16,000 × 64 b
Cores per Chip	352 (wafer scale)	1	4,096	16	128
Chip Area (mm^2)	50 (single core)	168	430	102	60
Technology (nm)	180	180	28	130	14 (FinFET)
Energy/SOP (pJ)	174[a]	941[a]	27[b]	27e3[c]	min. 105.3[d]

[a]Calculated as *total power*/(*spike per second* × *number of synapses*) (data from Benjamin et al. [15]).
[b]Calculated as *total power*/(*number of neurons* × *firing frequency* × *number of synapses*) (data from Merolla et al. [122]).
[c]Calculated as *total power/number of connections per second* (data from Furber et al. [49]).
[d]Calculated as *active neural update* + *synaptc operation* + *within tile transmission* (data from Davies et al. [38]).

implemented as several wafers interconnected together, with each wafer consisting of several HiCANN neurocores. It targets the emulation of brain-size neural networks with precise biological neural behavior at accelerated time. Neurogrid [15] is an SNN evaluator designed for subthreshold analog electronics operation. It operates in real time and emulates a few bio-realistic mechanisms. Finally, TrueNorth [122] is a neuromorphic chip implemented in digital electronics. It targets very low power large-scale networks evaluation and implements crossbar array with limited weight values representation and time-multiplexed neuron updates. These four chips are considered a big step in the advance of spiking neuromorphic processors, mostly with the aim of mimicking biology, and TrueNorth targeting low-power machine learning with spiking operators.

Recently, Intel put a step in the balance with the Loihi processor [38]. It is a digital processor targeting flexibility of operation for large-scale SNN evaluation. In terms of functionality, the Loihi chip is at the frontier between bio-mimicry and machine learning with SNNs. It integrates on-chip learning with various learning rules implementable, complex neuron models, and several information coding protocols, among others. It thus enables the emulation of many different algorithms. The authors target acceleration of the research on SNN performance. Table 1, inspired from Furber [48], describes relevant characteristics for large-scale system comparisons.

6.2 Low-Power Spiking Machine Learning

With regard to application-specific accelerators, we focus here on the recently proposed image recognition processors. Visual classification is currently a major field of machine learning because of the potential applications in autonomous cars, robots, and drones, among others. Therefore, many SNN accelerator designers focus on realizing image classifier accelerators. Spiking hardware proofs of concept are often benchmarked on the MNIST digit recognition dataset, even though it is a formal dataset. We believe it is so because it allows to train shallow SNNs with various learning rules and still obtain relatively high accuracies. Hence, we base our comparison on results obtained on this dataset. Nevertheless, we still argue that there are two major problems related to this dataset. First, it requires conversion from frame- to event-driven information, whose protocol may vary between benchmarked works. Second, it does not contain temporal information in itself, so it may not be adapted to exploit the full computational capacity of SNNs. Moreover, around 0.2% error on MNIST was already achieved in 2003 using a formal neural network [27], whereas the currently demonstrated best results using spiking networks are above 0.5% error [81, 155]. Hence, we argue that researchers in the field should select a dataset onto which SNN accelerators could be compared fairly, where timing information is relevant, and no input conversion is required. Several event-driven datasets obtained with bio-inspired image sensors have already been proposed [18, 125, 169, 204].

Recently, a few groups presented near ideal accuracies on visual datasets thanks to the development of powerful learning rules or conversion methods [81, 88, 104, 155] but did not evaluate their algorithms on dedicated hardware. To perform operations on specialized processors, the algorithms are usually modified to better fit the circuit capabilities, even if the hardware and software are co-designed. We thus focus on the works that resulted in neuromorphic circuit evaluation of SNNs on MNIST [34, 115, 130, 193, 201]. We exclusively consider hardware-related data to give a comparison overview of the latest SNN accelerators targeting low-power and potential embedding (Table 2). Moreover, for the sake of fairness, we add similar data measured on a very efficient formal ANN processor [189] in the last column of this table.

Yin et al. [193], in the SNN hardware category, reach the highest accuracy at the smallest energy consumption for equivalent area. They train their network directly with spikes, off-chip, using a modified version of LIF neurons that enable backpropagation via straight through estimator (STE) gradients. Through simulation of a spiking CNN using the same computational units, they reach 99.40% accuracy, which puts them slightly behind the state of the art [81, 155]. However, Whatmough et al. [189] propose an ANN accelerator that realizes near identical accuracy (0.34% less) for two times less energy per classification, but with more than three times increase in area. Nevertheless, Yin et al. [193] report that by reducing their accuracy by less than 0.2%, they can realize three times gain in energy at iso-accuracy with respect to Whatmough et al. [189]. It illustrates the ability of SNNs to achieve impressive energy accuracy tradeoffs. Interestingly, they can reach such efficiency because their hardware implementation is designed specifically for the SNN topology it evaluates. It thus lacks flexibility but trades it for superior energy efficiency.

In addition, each of the presented processors use different strategies to improve its efficiency, such as WQ. However, it remains important to consider both hardware and software performance when dealing with energy efficiency improvement. For example, Zheng and Mazumder [201] reduced the size and complexity of their topology to enable SNN evaluation at low circuit cost, but it induced an important accuracy loss. Indeed, before simplification, their network reached an accuracy of 97.8%, and after down sampling the input data, evaluation with approximate computing, and other methods, they reduced their accuracy to around 90% on the same dataset.

We must specify that the data given for the chip of Chen et al. [34] are actually obtained through evaluation of a BNN onto the neuromorphic hardware. They present a way of evaluating formal binary networks on neuromorphic hardware with LIF neurons. Their threshold is set at 0, with an infinite leak rate, and the weights are quantized to either 0 or 1. Nevertheless, as proof of concept, they also test their hardware for SNN evaluation using on-chip STDP learning for image reconstitution, such as in Knag et al. [96].

Therefore, spiking neuromorphic hardware allows to perform the MNIST digit recognition task for hundreds of nanojoules at greater than 96% accuracy. Such accelerators allow energy/accuracy

Table 2. Small-Scale Low-Power Accelerator Comparison

Processor	Yin et al. [193]	Mostafa et al. [130]	Zheng and Mazumder [201]	Chen et al. [34]	Whatmough et al. [189]
Year	2017	2017	2018	2018	2017
Network Type	SNN				ANN
Technology	Simulated 28nm	FPGA	Simulated 65nm	10nm FinFET	28nm
Implementation	Digital	Digital	Digital	Digital	Digital
Training Type	Backpropagation Off-chip	Backpropagation Off-chip	Modulated STDP On-chip	Formal BNN Off-chip	Formal Backpropagation
Coding Scheme	Rate	ItL[a]	Rate	Rate	NR
Accuracy on MNIST	98.7	96.08	~90	97.9[b]	98.36
Spikes per Classification	NC	135	NC	~130k	NR
Network Topology	FC 3 layers	FC 3 layers	FC 3 layers	FC 4 layers	FC 5 layers
Number of Neurons	1,306	1,394	316[c]	2,330	1,562
Input Conversion Mode	Bernoulli spike probability	Binarized ItL	Continuous values [200]	/	NR
Energy per Classification	773nJ	NC	1.12uJ	1.7uJ	360nJ
Chip Size	1.65mm^2	NC	1.1mm^2	1.72mm^2	5.76mm^2
Energy Efficiency	WQ[d]	WQ + Single spike per neuron + Approx. multiplication (decay)	WQ + Approx. division (training) + Random re-fractoriness	WQ + Sparse connect. + Stochastic dropping	WQ + Approximate computing + Zero guarding + Voltage scaling

[a]Intensity to latency conversion.
[b]Evaluation realized with a BNN on neuromorphic spiking hardware.
[c]MNIST data down sampled from 28 × 28 to 16 × 16.
[d]Weight quantization.
NC, not communicated; NR, not relevant.

trade-off by reducing the number of time steps at inference. The inference can even be realized in a single time step without accuracy loss if the network is trained under this constraint (Park et al. [207]). SNNs are thus low power and can reach equivalent results on MNIST at lower power than the current state-of-the-art ASIC for energy-efficient ANN evaluation. Nevertheless, we argue that this dataset is not suited for SNN evaluation given that it does not allow these networks to exploit their full computational capability.

7 WHAT COMES NEXT?

Throughout this survey, it became apparent that algorithm and hardware performance are interwoven. We would like to give an overview of the ways SNNs performance could be improved from both software and hardware perspectives.

7.1 Algorithm Improvements

From a software point of view, SNNs still have an important improvement margin. Backpropagation of the gradient for supervised learning has recently delivered state-of-the-art results [81, 104]. It is, however, not compatible with every kind of network. It requires modification of neuron models and thus still limits its scope of applications among SNN topologies. Nevertheless, it allows reaching good results, and we may soon begin to see machine learners working with deep SNNs.

Moreover, information coding in SNNs can allow reducing the total number of operations for a dedicated task on top of being intrinsically sparse. It is thus no surprise that an increasing number of works report implementing time coding SNNs [129, 132, 154, 196]. Nevertheless, precise timing operations are still limited because of the lack of efficient training methods. Mozafari et al. [131] and Mozafari et al. [132] employ reward-modulated STDP to train the network, but the learning rule does not enable multi-layer training. Rueckauer and Liu [154] convert an ANN to an SNN using a temporal encoding unit. They can thus train deep networks, but the conversion induces a loss in accuracy, with an error on MNIST of 1.04% and 1.43% before and after conversion, respectively. Mostafa [129] uses backpropagation directly with spikes, but the reached accuracy on MNIST of 97.55% does not compete with a trained ANN. Therefore, even if temporal coding is appealing thanks to the low number of spikes required to operate, algorithms based on it do not yet allow to reach accuracies comparable to state-of-the-art algorithms, both formal- and event driven. Another way of reducing the number of spikes when using rate coding is to use adapting neuron models, which permits to maintain a low spike frequency while enlarging the dynamic range of the units. Zambrano and Bohte [198] demonstrate that using such units allows to reach higher accuracies than traditional rate coding SNNs while reducing the total number of operations per classification.

To sum up, SNN algorithms are very interesting because they allow both complex modeling of bio-plausible dynamics and machine learning tasks. The designers can adapt the complexity of their model to the targeted application. However, today, the scope of available solutions is large, and not one of these stands out in the crowd for machine learning tasks. Once again, this may be because the benchmarks employed are not suited for fully exploiting SNN capabilities, given that researchers usually compare themselves to common ANNs.

7.2 Hardware Implementations

In terms of hardware, two fields are starting to stand out: crossbar arrays with NVM devices and 3D integration. As we already discussed in Section 4.5, the crossbar arrays are quite promising for fast and low-power neuromorphic architecture design. They are still at the state of research, but promising results have already been demonstrated [5].

3D integration technology brings the advantages of high bandwidth, shorter interconnection designs, and potential high parallelism [140]. It enables interconnecting circuits on more than a single plan, with vertical wiring of the plans. Many works have already considered leveraging 3D technology to improve neuromorphic computing efficiency by implementing one layer by 3D tier (Figure 11(b)) [13, 33], or by separating the memory and logic part on different tiers (Figure 11(c)) [33, 89], or by maximizing parallel processing of analog/mixed signal designs with the use of crossbar arrays on the second plan [41]. Chang et al. [33] show that in monolithic 3D, implementation of digital neuromorphic chip for formal processing is more performant if memory and logic are both distributed on several chips, which allows saving around 20% power with respect to the same 2D implementation. However, in terms of speed, Kim et al. [89] show that stacking memory on top of the logic plan can highly increase throughput with respect to the traditional memory-to-the-side implementation. Based on this, Amir et al. [6] are already envisioning sensors with integrated DNN computations at a low footprint. In addition, some groups [41, 84] have proposed to exploit

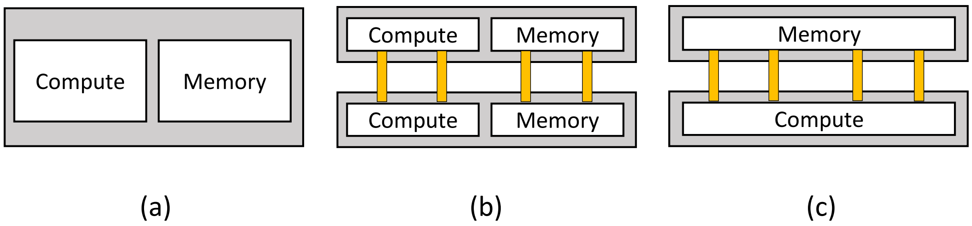

Fig. 11. Comparison of a traditional 2D implementation of a neurocore and novel 3D designs. (a) Schematic of a 2D design with a computing part and the memory aside of it. (b) 3D design with both algorithmic and memory circuitry in each tier. (c) Another 3D implementation with a layer dedicated to memory and another to the computing elements.

the through-silicon via technology process constraints on the density of interconnects to design analog neuron models with a reduced capacitance footprint. Therefore, we see works dealing with formal ANN processors whose throughput and power consumption are increased and decreased, respectively, or works studying the benefits of 3D for a parallel event-driven architecture. However, 3D circuits is not a mature technology yet [12, 158]. It brings several design constraints and comes at an important process cost. Moreover, because the AER protocol already allows low-power communication between neurocores, further work is required to see how far 3D technology can improve the performance of SNN accelerators.

Crossbar arrays and 3D circuits are not the only trail of hardware improvement. For example, Morro et al. [128] have reported replacing a part of their ASIC, initially designed with traditional digital gates, with neuromorphic hardware. This sort of mixed neuromorphic integration is probably relevant for other applications. Yousefzadeh et al. [195] developed a chip for evaluating formal ANNs that uses event-driven communication between layers, a topology they refer to as the hybrid neural network. It requires circuits to convert asynchronous event-driven information into frames, and inversely. They employ the AER protocol between layers, where information is encoded with 4 bits, thus guaranteeing minimal AER bus width overhead. For each non-zero activation, a single word packet is sent through the asynchronous NoC, which is sufficient to transmit the full information from neuron to neuron. They reach a relatively low accuracy of 97.09% on MNIST, for 7uJ/frame with an implementation on FPGA, which is below the state of the art (see Section 6.2). However, further optimization and dedicated silicon implementation may allow better performance. It is an interesting approach in the sense that this technology could combine the advantages of both the SNN and ANN while mitigating their drawbacks.

8 CONCLUSION

Neuromorphic computing is at an early stage, yet it is already a major field of research. In this survey, we focused on the hardware implementation of SNNs and presented the state of the art. Neuromorphic processors for SNN acceleration can be designed for both deep network evaluation, where large scale circuits are implemented, and embedded applications, employing smaller-scale accelerators. In Section 4, we discussed the fact that hardware requirements are minimized thanks to intrinsic properties of SNNs. Moreover, algorithmic modifications, such as WQ or stochastic computing, help to reduce hardware resource requirements, in terms of memory and computation, which is beneficial for embedded operation. However, it may lead to a significant loss in accuracy of the network if those methods are not applied carefully, and formal networks can also leverage most of these techniques. We observed that designers targeting machine learning applications are increasingly using digital implementations, which shows a cutoff point with the initial philosophy around hardware implementation of SNNs that focused on bio-emulation in analog electronics. We suggest that it may be because, up to now, bio-plausibility is not associated with better algorithmic performance, and many works are thus implementing simplified versions of neuron and synapse models, which are suited for digital evaluation.

This work also discussed the current trend in the community regarding benchmarking with the MNIST dataset, and we argued that it might not allow to fully exploit the SNN capabilities. Designers should prefer benchmarks dedicated to SNN evaluation, where temporal information is inherent to the dataset. Nevertheless, based on this dataset, current state-of-the-art low-power SNN accelerators are slightly better than ANN ones in terms of energy and area. It is important to note that the ANN reference of Table 2 is a programmable accelerator able to manage several small topologies, whereas the accelerator of Yin et al. [193] has been specifically designed to implement a single network topology. We concede that this design strategy may not be suited for any applications; however, it is very efficient.

Finally, we briefly presented the ongoing research on the evaluation of neural networks on crossbar array and 3D integration technology. Those may enable efficiency improvement of neuromorphic hardware; however, crossbar arrays with memristive devices still do not meet the requirements, and 3D technology is at a very early stage of research.

REFERENCES

[1] S. A. Aamir, Y. Stradmann, P. Müller, C. Pehle, A. Hartel, A. Grübl, J. Schemmel et al. 2018. An accelerated LIF neuronal network array for a large scale mixed-signal neuromorphic architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers* 65, 12 (2018), 4299–4312.

[2] K. Abdelouahab, M. Pelcat, J. Sérot, C. Bourrasset, and F. Berry. 2017. Tactics to directly map CNN graphs on embedded FPGAs. *IEEE Embedded Systems Letters* 9, 4 (2017), 113–116.

[3] S. Agarwal, T.-T. Quach, O. Parekh, A. H. Hsia, E. P. DeBenedictis, C. D. James, M. J. Marinella et al. 2016. Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding. *Frontiers in Neuroscience* 9 (2016), 484.

[4] S.-I. Amari and M. A. Arbib. 1977. Competition and cooperation in neural nets. In *Systems Neurosciences*, J. Metzler (Ed.). Academic Press, Cambridge, MA, 119–165.

[5] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler et al. 2018. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* 558, 7708 (2018), 60–67.

[6] M. F. Amir, J. H. Ko, T. Na, D. Kim, and S. Mukhopadhyay. 2018. 3-D stacked image sensor with deep neural network computation. *IEEE Sensors Journal* 18, 10 (2018), 4187–4199.

[7] A. Ankit, A. Sengupta, P. Panda, and K. Roy. 2017. RESPARC: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks. In *Proceedings of the 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC'17)*. 1–6.

[8] B. Ans and S. Rousset. 1997. Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Académie des Sciences—Series III—Sciences de la Vie* 320, 12 (1997), 989–997.

[9] J. Backus. 2007. Can programming be liberated from the von Neumann style?: A functional style and its algebra of programs. In *ACM Turing Award Lectures*. ACM, New York, NY.

[10] W. Barker, D. M. Halliday, Y. Thoma, E. Sanchez, G. Tempesti, and A. M. Tyrrell. 2007. Fault tolerance using dynamic reconfiguration on the POEtic tissue. *IEEE Transactions on Evolutionary Computation* 11, 5 (2007), 666–684.

[11] A. Basu, J. Acharya, T. Karnik, H. Liu, H. Li, J.-S. Seo, and C. Song. 2018. Low-power, adaptive neuromorphic systems: Recent progress and future directions. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 8, 1 (2018), 6–27.

[12] P. Batude, C. Fenouillet-Beranger, L. Pasini, V. Lu, F. Deprat, L. Brunet, B. Sklenard et al. 2015. 3DVLSI with CoolCube process: An alternative path to scaling. In *Proceedings of the 2015 Symposium on VLSI Technology (VLSI Technology'15)*. T48–49.

[13] B. Belhadj, A. Valentian, P. Vivet, M. Duranton, L. He, and O. Temam. 2014. The improbable but highly appropriate marriage of 3D stacking and neuromorphic accelerators. In *Proceedings of the 2014 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'14)*. 1–9.

[14] G. Bellec, D. Kappel, W. Maass, and R. Legenstein. 2017. Deep rewiring: Training very sparse deep networks. arXiv:1711.05136.

[15] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza et al. 2014. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE* 102, 5 (2014), 699–716.

[16] M. Bernert and B. Yvert. 2017. Fully unsupervised online spike sorting based on an artificial spiking neural network. *bioRxiv* (2017), 23622436224.

[17] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. DeSalvo, and C. Gamrat. 2012. Visual pattern extraction using energy-efficient "2-PCM synapse" neuromorphic architecture. *IEEE Transactions on Electron Devices* 59, 8 (2012), 2206–2214.

[18] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck. 2017. DDD17: End-to-end DAVIS driving dataset. arXiv:1711.01458.

[19] K. A. Boahen. 2000. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47, 5 (2000), 416–434.

[20] S. M. Bohte. 2012. Efficient spike-coding with multiplicative adaptation in a spike response model. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*. 1835–1843.

[21] S. M. Bohte, J. N. Kok, and H. L. Poutrã. 2002. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, (2002), 17–37.

[22] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. 2014. A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits* 49, 10 (2014), 2333–2341.

[23] L. Buesing, J. Bill, B. Nessler, and W. Maass. 2011. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLOS Computational Biology* 7, 11 (2011), e1002211.

[24] G. W. Burr, P. Narayanan, R. M. Shelby, S. Sidler, I. Boybat, C. di Nolfo, and Y. Leblebici. 2015. Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power). In *Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM'15)*. 4.4.1–4.4.4.

[25] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani et al. 2017. Neuromorphic computing using non-volatile memory. *Advances in Physics: X* 2, 1 (2017), 89–124.

[26] G. W. Burr, R. M. Shelby, S. Sidler, C. di Nolfo, J. Jang, I. Boybat, R. S. Shenoy et al. 2015. Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices* 62, 11 (2015), 3498–3507.

[27] A. Calderón, A. Calderón, S. Roa, and J. Victorino. 2003. Handwritten digit recognition using convolutional neural networks and Gabor filters. In *Proceedings of the International Congress on Computational Intelligence*.

[28] L. A. Camuñas-Mesa, Y. L. Domínguez-Cordero, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco. 2018. A configurable event-driven convolutional node with rate saturation mechanism for modular ConvNet systems implementation. *Frontiers in Neuroscience* 12 (2018), 63.

[29] Y. Cao, Y. Chen, and D. Khosla. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision* 113, 1 (2015), 54–66.

[30] N. Caporale and Y. Dan. 2008. Spike timing–dependent plasticity: A Hebbian learning rule. *Annual Review of Neuroscience* 31, 1 (2008), 25–46.

[31] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta et al. 2013. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN'13)*. 1–10.

[32] V. Chan, S.-C. Liu, and A. van Schaik. 2007. AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems I: Regular Papers* 54, 1 (2007), 48–59.

[33] K. Chang, D. Kadetotad, Y. Cao, J. Sun Seo, and S. K. Lim. 2017. Monolithic 3D IC designs for low-power deep neural networks targeting speech recognition. In *Proceedings of the 2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED'17)*. 1–6.

[34] G. K. Chen, R. Kumar, H. E. Sumbul, P. Knag, and R. K. Krishnmurthy. 2018. A 4096-neuron 1M-synapse 3.8pJ/SO P spiking neural network with on-chip STDP learning and sparse weights in 10nm FinFET CMOS. In *Proceedings of the IEEE Symposium on VLSI Circuits*. C22–24.

[35] C.-H. Chien, S.-C. Liu, and A. Steimer. 2018. A neuromorphic VLSI circuit for spike-based random sampling. *IEEE Transactions on Emerging Topics in Computing* 6, 1 (2018), 135–144.

[36] M. Courbariaux, Y. Bengio, and J.-P. David. 2015. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, Vol. 2. 3123–3131.

[37] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, and S. Schaafsma. 2018. Mapping of local and global synapses on spiking neuromorphic hardware. In *Proceedings of the 2018 Design, Automation, and Test in Europe Conference and Exhibition (DATE'18)*. 1217–1222.

[38] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.

[39] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN'15)*. 1–8.

[40] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci. 2016. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. arXiv:1601.04187.

[41] M. A. Ehsan, Z. Zhou, and Y. Yi. 2017. Neuromorphic 3D integrated circuit. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI'17)*. 221–226.

[42] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha. 2015. Backpropagation for energy-efficient neuromorphic computing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS'15)*, Vol. 1. 1117–1125.

[43] A. L. Fairhall, G. D. Lewen, W. Bialek, and R. R. de Ruyter van Steveninck. 2001. Efficiency and ambiguity in an adaptive neural code. *Nature* 412, 6849 (2001), 787–792.

[44] D. E. Feldman. 2009. Synaptic mechanisms for plasticity in neocortex. *Annual Review of Neuroscience* 32, 1 (2009), 33–55.

[45] R. V. Florian. 2007. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation* 19, 6 (2007), 1468–1502.

[46] R. V. Florian. 2012. The chronotron: A neuron that learns to fire temporally precise spike patterns. *PLOS ONE* 7, 8 (2012), e40233.

[47] R. M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences* 3, 4 (1999), 128–135.

[48] S. Furber. 2016. Large-scale neuromorphic computing systems. *Journal of Neural Engineering* 13, 5 (2016), 051001.

[49] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. 2014. The SpiNNaker project. *Proceedings of the IEEE* 102, 5 (2014), 652–665.

[50] F. Galluppi, S. Davies, A. Rast, T. Sharp, L. A. Plana, and S. Furber. 2012. A hierarchical configuration system for a massively parallel neural hardware platform. In *Proceedings of the 9th Conference on Computing Frontiers (CF'12)*. 183.

[51] D. Garbin, E. Vianello, O. Bichler, Q. Rafhay, C. Gamrat, G. Ghibaudo, B. DeSalvo, and L. Perniola. 2015. HfO2-based OxRAM devices as synapses for convolutional neural networks. *IEEE Transactions on Electron Devices* 62, 8 (2015), 2494–2501.

[52] W. Gerstner and W. M. Kistler. 2002. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.

[53] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. 2014. *Neuronal Dynamics—From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, Cambridge, UK.

[54] S. Ghosh-Dastidar and H. Adeli. 2009. Spiking neural networks. *International Journal of Neural Systems* 19, 4 (2009), 295–308.

[55] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv:1312.6211.

[56] I. Goodfellow, B. Yoshua, and C. Aaron. 2016. *Deep Learning*. MIT Press, Cambridge, MA.

[57] F. Grassia, T. Levi, E. Doukkali, and T. Kohno. 2018. Spike pattern recognition using artificial neuron and spike-timing-dependent plasticity implemented on a multi-core embedded platform. *Artificial Life and Robotics* 23, 2 (2018), 200–204.

[58] P. C. Haddow and A. M. Tyrrell. 2011. Challenges of evolvable hardware: Past, present and the path to a promising future. *Genetic Programming and Evolvable Machines* 12, 3 (2011), 183–215.

[59] B. Han, A. Ankit, A. Sengupta, and K. Roy. 2018. Cross-layer design exploration for energy-quality tradeoffs in spiking and non-spiking deep artificial neural networks. *IEEE Transactions on Multi-Scale Computing Systems* 4, 4 (2018), 613–623.

[60] J. Han and M. Orshansky. 2013. Approximate computing: An emerging paradigm for energy-efficient design. In *Proceedings of the 2013 18th IEEE European Test Symposium (ETS'13)*. 1–6.

[61] S. Han, H. Mao, and W. J. Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv:1510.00149.

[62] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*.

[63] S. Herculano-Houzel. 2011. Scaling of brain metabolism with a fixed energy budget per neuron: Implications for neuronal activity, plasticity and evolution. *PLOS ONE* 6, 3 (2011), e17514.

[64] S. Hoppner, Y. Yan, B. Vogginger, A. Dixius, J. Partzsch, F. Neumarker, S. Hartmann et al. 2017. Dynamic voltage and frequency scaling for neuromorphic many-core systems. In *Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS'17)*. 1–4.

[65] K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.

[66] M. Horowitz. 2014. 1.1 Computing's energy problem (and what we can do about it). In *Proceedings of the 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC'14)*. 10–14.

[67] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. 2016. Quantized neural networks: Training neural networks with low precision weights and activations. arXiv:1609.07061.

[68] D. H. Hubel and T. N. Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology* 160, 1 (1962), 106–154. DOI: https://doi.org/10.1113/jphysiol.1962.sp006837

[69] D. Huh and T. J. Sejnowski. 2017. Gradient descent for spiking neural networks. arXiv:1706.04698.

[70] G. Indiveri. 2003. A low-power adaptive integrate-and-fire neuron circuit. In *Proceedings of the 2003 International Symposium on Circuits and Systems (ISCAS'03)*. IV-820–823. DOI: https://doi.org/10.1109/ISCAS.2003.1206342

[71] G. Indiveri, E. Chicca, and R. Douglas. 2006. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Transactions on Neural Networks* 17, 1 (2006), 211–221.

[72] G. Indiveri, F. Corradi, and N. Qiao. 2015. Neuromorphic architectures for spiking deep neural networks. In *Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM'15)*. 4.2.1–4.2.4.

[73] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu et al. 2011. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience* 5 (2011), 73.

[74] G. Indiveri and S.-C. Liu. 2015. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE* 103, 8 (2015), 1379–1397.

[75] E. M. Izhikevich. 2004. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks* 15, 5 (2004), 1063–1070.

[76] R. B. Jacobs-Gedrim, S. Agarwal, K. E. Knisely, J. E. Stevens, M. S. van Heukelom, D. R. Hughart, J. Niroula et al. 2017. Impact of linearity and write noise of analog resistive memory devices in a neural algorithm accelerator. In *Proceedings of the 2017 IEEE International Conference on Rebooting Computing (ICRC'17)*. 1–10.

[77] M. Jerry, A. Parihar, B. Grisafe, A. Raychowdhury, and S. Datta. 2017. Ultra-low power probabilistic IMT neurons for stochastic sampling machines. In *Proceedings of the 2017 Symposium on VLSI Technology*. T186–187.

[78] Y. Jin and P. Li. 2016. AP-STDP: A novel self-organizing mechanism for efficient reservoir computing. In *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN'16)*. 1158–1165.

[79] Y. Jin and P. Li. 2017. Performance and robustness of bio-inspired digital liquid state machines. *Neurocomputing* 226, C (2017), 145–160.

[80] Y. Jin, Y. Liu, and P. Li. 2016. SSO-LSM: A sparse and self-organizing architecture for liquid state machine based neural processors. In *Proceedings of the 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'16)*. 55–60.

[81] Y. Jin, W. Zhang, and P. Li. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. arXiv:1805.07866.

[82] J. Jo and Y. Bengio. 2017. Measuring the tendency of CNNs to learn surface statistical regularities. arXiv:1711.11561.

[83] A. P. Johnson, J. Liu, A. G. Millard, S. Karim, A. M. Tyrrell, J. Harkin, J. Timmis et al. 2018. Homeostatic fault tolerance in spiking neural networks: A dynamic hardware perspective. *IEEE Transactions on Circuits and Systems I: Regular Papers* 65, 2 (2018), 687–699.

[84] A. Joubert, M. Duranton, B. Belhadj, O. Temam, and R. Héliot. 2012. Capacitance of TSVs in 3-D stacked chips a problem? Not for neuromorphic systems! In *Proceedings of the 2012 Design Automation Conference (DAC'12)*. 1260–1261.

[85] N. P. Jouppi, A. Borchers, R. Boyle, P. Luc Cantin, C. Chao, C. Clark, J. Coriell et al. 2017. In-datacenter performance analysis of a tensor processing unit. arXiv:1704.04760.

[86] A. Karpathy. 2018. CS231n Convolutional Neural Networks for Visual Recognition. Retrieved March 14, 2019 from http://cs231n.github.io/.

[87] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier. 2016. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing* 205 (2016), 382–392.

[88] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier. 2018. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks* 99 (2018), 56–67.

[89] D. Kim, J. Kung, S. Chai, S. Yalamanchili, S. Mukhopadhyay, D. Kim, J. Kung et al. 2016. Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory. In *Proceedings of the 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA'16)*. 380–392.

[90] H. Kim, S. Hwang, J. Park, S. Yun, J.-H. Lee, and B.-G. Park. 2018. Spiking neural network using synaptic transistors and neuron circuits for pattern recognition with noisy images. *IEEE Electron Device Letters* 39, 4 (2018), 630–633.

[91] J. K. Kim, P. Knag, T. Chen, and Z. Zhang. 2015. A 640M pixel/s 3.65mW sparse event-driven neuromorphic object recognition processor with on-chip learning. In *Proceedings of the 2015 Symposium on VLSI Circuits (VLSI Circuits'15)*. C50–51.

[92] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan et al. 2015. NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. In *Proceedings of the 2015 IEEE International Electron Devices Meeting (IEDM'15)*. 17.1.1–17.1.4.

[93] S. Kim, M. Lim, Y. Kim, H.-D. Kim, and S.-J. Choi. 2018. Impact of synaptic device variations on pattern recognition accuracy in a hardware neural network. *Scientific Reports* 8, 1 (2018), 2638.

[94] Y. Kim, Y. Zhang, and P. Li. 2013. An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems. In *Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'13)*. 130–137.

[95] Y. Kim, Y. Zhang, and P. Li. 2015. A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing. *ACM Journal on Emerging Technologies in Computing Systems* 11, 4 (2015), 1–25.

[96] P. Knag, J. K. Kim, T. Chen, and Z. Zhang. 2015. A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding. *IEEE Journal of Solid-State Circuits* 50, 4 (2015), 1070–1079.

[97] P. Knag, W. Lu, and Z. Zhang. 2014. A native stochastic computing architecture enabled by memristors. *IEEE Transactions on Nanotechnology* 13, 2 (2014), 283–293.

[98] B. W. Ku, Y. Liu, Y. Jin, S. Samal, P. Li, and S. K. Lim. 2018. Design and architectural co-optimization of monolithic 3D liquid state machine-based neuromorphic processor. In *Proceedings of the 55th Annual Design Automation Conference (DAC'18)*. 165:1–165:6.

[99] S. R. Kulkarni and B. Rajendran. 2018. Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization. *Neural Networks* 103 (2018), 118–127.

[100] A. Kumar, S. Rotter, and A. Aertsen. 2010. Spiking activity propagation in neuronal networks: Reconciling different perspectives on neural coding. *Nature Reviews Neuroscience* 11, 9 (2010), 615–627.

[101] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.

[102] C. Lee, P. Panda, G. Srinivasan, and K. Roy. 2018. Training deep spiking convolutional neural networks with STDP-based unsupervised pre-training followed by supervised fine-tuning. *Frontiers in Neuroscience* 12 (2018), 435.

[103] C.-E. Lee, T. Chen, and Z. Zhang. 2017. A 127mW 1.63TOPS sparse spatio-temporal cognitive SoC for action classification and motion tracking in videos. In *Proceedings of the 2017 Symposium on VLSI Circuits*. C226–227.

[104] J. H. Lee, T. Delbruck, and M. Pfeiffer. 2016. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience* 10 (2016), 508.

[105] J. A. Leñero-Bardallo, R. Carmona-Galán, and A. Rodríguez-Vázquez. Applications of event-based image sensors—Review and analysis. *International Journal of Circuit Theory and Applications* 46, 9, 1620–1630.

[106] J. Liang and H.-S. P. Wong. 2010. Cross-point memory array without cell selectors—Device characteristics and data storage pattern dependencies. *IEEE Transactions on Electron Devices* 57, 10 (2010), 2531–2538.

[107] P. Lichtsteiner, C. Posch, and T. Delbruck. 2008. A 128 x 128 120 dB 15 us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits* 43, 2 (2008), 566–576.

[108] K. Likharev, A. Mayr, I. Muckra, and Ö. Türel. 2003. CrossNets: High-performance neuromorphic architectures for CMOL circuits. *Annals of the New York Academy of Sciences* 1006, 1 (2003), 146–163.

[109] J. Liu, J. Harkin, L. P. Maguire, L. J. McDaid, and J. J. Wade. 2018. SPANNER: A self-repairing spiking neural network hardware architecture. *IEEE Transactions on Neural Networks and Learning Systems* 29, 4 (2018), 1287–1300.

[110] L. Liu, Y. Jin, Y. Liu, N. Ma, Z. Zou, and L. Zheng. 2017. Designing bio-inspired autonomous error-tolerant massively parallel computing architectures. In *Proceedings of the 2017 30th IEEE International System-on-Chip Conference (SOCC'17)*. 274–279.

[111] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas. 2015. *Event-Based Neuromorphic Systems*. John Wiley & Sons.

[112] T. Liu, Z. Liu, F. Lin, Y. Jin, G. Quan, and W. Wen. 2018. MT-Spike: A multilayer time-based spiking neuromorphic architecture with temporal error backpropagation. arXiv:1803.05117.

[113] X. Liu, W. Wen, X. Qian, H. Li, and Y. Chen. 2018. Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems. In *Proceedings of the 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC'18)*. 141–146.

[114] Y. Liu, Y. Jin, and P. Li. 2018. Online adaptation and energy minimization for hardware recurrent spiking neural networks. *Journal on Emerging Technologies in Computing Systems* 14, 1 (2018), 11:1–11:21.

[115] D. Ma, J. Shen, Z. Gu, M. Zhang, X. Zhu, X. Xu, Q. Xu et al. 2017. Darwin: A neuromorphic hardware co-processor based on spiking neural networks. *Journal of Systems Architecture* 77 (2017), 43–51.

[116] W. Maass. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks* 10, 9 (1997), 1659–1671.

[117] W. Maass. 2000. On the computational power of winner-take-all. *Neural Computation* 12, 11 (2000), 2519–2535.

[118] N. R. Mahapatra and B. Venkatrao. 1999. The processor-memory bottleneck. *Crossroads* 5, 3es (1999), Article 2.

[119] T. Marukame, K. Nomura, M. Matusmoto, S. Takaya, and Y. Nishi. 2018. Proposal, analysis and demonstration of analog/digital-mixed neural networks based on memristive device arrays. In *Proceedings of the 2018 International Symposium on Circuits and Systems (ISCAS'18)*. 1–5.

[120] T. Masquelier and S. J. Thorpe. 2007. Unsupervised learning of visual features through spike timing dependent plasticity. *PLOS Computational Biology* 3, 2 (2007), e31.

[121] C. Mead. 1990. Neuromorphic electronic systems. *Proceedings of the IEEE* 78, 10 (1990), 1629–1636.

[122] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668.

[123] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha. 2011. A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *Proceedings of the 2011 IEEE Custom Integrated Circuits Conference (CICC'11)*. 1–4.

[124] T. Mesquida, A. Valentian, D. Bol, and E. Beigne. 2016. Impact of the AER-induced timing distortion on spiking neural networks implementing DSP. In *Proceedings of the 2016 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME'16)*. 1–4.

[125] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr et al. 2016. Steering a predator robot using a mixed frame/event-driven convolutional neural network. arXiv:1606.09433.

[126] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov. 2012. SPAN: Spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems* 22, 4 (2012), 1250012.

[127] B. Moons and M. Verhelst. 2016. A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets. arXiv:1606.05094.

[128] A. Morro, V. Canals, A. Oliver, M. L. Alomar, F. Galan-Prado, P. J. Ballester, and J. L. Rossello. 2018. A stochastic spiking neural network for virtual screening. *IEEE Transactions on Neural Networks and Learning Systems* 29, 4 (2018), 1371–1375.

[129] H. Mostafa. 2017. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 29, 7 (2017), 3227–3235.

[130] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs. 2017. Fast classification using sparsely active spiking networks. In *Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS'17)*. 1–4.

[131] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier. 2018. Combining STDP and reward-modulated STDP in deep convolutional spiking neural networks for digit recognition. arXiv:1804.00227.

[132] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh. 2017. First-spike based visual categorization using reward-modulated STDP. arXiv:1705.09132.

[133] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs. 2016. Stochastic synapses enable efficient brain-inspired learning machines. *Frontiers in Neuroscience* 10 (2016), 241.

[134] M. A. Nielsen. 2015. Neural Networks and Deep Learning. Retrieved March 14, 2019 from http://neuralnetworksanddeeplearning.com.

[135] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. 2015. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience* 9 (2015), 437.

[136] Z. Pajouhi. 2018. Energy efficient neuromorphic processing using spintronic memristive device with dedicated synaptic and neuron terminology. In *Proceedings of the 2018 19th International Symposium on Quality Electronic Design (ISQED'18)*. 61–68.

[137] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar. 2012. Optoelectronic reservoir computing. *Scientific Reports* 2, 1 (2012), Article 287.

[138] J. Park, T. Yu, S. Joshi, C. Maier, and G. Cauwenberghs. 2017. Hierarchical address event routing for reconfigurable large-scale neuromorphic systems. *IEEE Transactions on Neural Networks and Learning Systems* 28, 10 (2017), 2408–2422.

[139] J. Partzsch, S. Hoppner, M. Eberlein, R. Schuffny, C. Mayr, D. R. Lester, and S. Furber. 2017. A fixed point exponential function accelerator for a neuromorphic many-core system. In *Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS'17)*. 1–4.

[140] V. F. Pavlidis, I. Savidis, and E. G. Frian. 2008. *Three-Dimensional Integrated Circuit Design.* Morgan Kaufmann.

[141] J. A. Perez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. Linares-Barranco. 2013. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (2013), 2706–2719.

[142] D. H. Perkel and T. H. Bullock. 1968. Neural coding. *Eurosciences Research Program Bulletin* 6, 3 (1968), 221–348.

[143] C. Pittenger and R. S. Duman. 2008. Stress, depression and neuroplasticity: A convergence of mechanisms. *Neuropsychopharmacology* 33, 1 (2008), 88–109.

[144] F. Ponulak. 2005. *ReSuMe—New Supervised Learning Method for Spiking Neural Networks.* Technical Report. Institute of Control and Information Engineering, Poznan University of Technology, Poland.

[145] C. Posch, D. Matolin, and R. Wohlgenannt. 2011. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits* 46, 1 (2011), 259–275.

[146] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov. 2015. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521, 7550 (2015), 61–64.

[147] N. Qiao and G. Indiveri. 2017. Analog circuits for mixed-signal neuromorphic computing architectures in 28 nm FD-SOI technology. In *Proceedings of the 2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S'17).* 1–4.

[148] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri. 2015. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers in Neuroscience* 9 (2015), 141.

[149] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. 2016. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *Computer Vision—ECCV 2016.* Lecture Notes in Computer Science, Vol. 9908. Springer, 525–542.

[150] N. Rathi, P. Panda, and K. Roy. 2018. STDP based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. arXiv:1710.04734.

[151] D. Roggen, S. Hofmann, Y. Thoma, and D. Floreano. Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot. In *Proceedings of the NASA/DoD Conference on Evolvable Hardware.* 189–198.

[152] U. B. Rongala, A. Mazzoni, and C. M. Oddo. 2017. Neuromorphic artificial touch for categorization of naturalistic textures. *IEEE Transactions on Neural Networks and Learning Systems* 28, 4 (2017), 819–829.

[153] A. Roy, S. Venkataramani, N. Gala, S. Sen, K. Veezhinathan, and A. Raghunathan. 2017. A programmable event-driven architecture for evaluating spiking neural networks. In *Proceedings of the 2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED'17).* 1–6.

[154] B. Rueckauer and S.-C. Liu. 2018. Conversion of analog to spiking neural networks using sparse temporal coding. In *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS'18).* 1–5.

[155] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience* 11 (2017), 682.

[156] R. V. Rullen and S. J. Thorpe. 2001. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation* 13, 6 (2001), 1255–1283.

[157] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang et al. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[158] K. Sakuma. 2018. *3D Integration in VLSI Circuits.* CRC Press, Boca Raton, FL.

[159] C. D. Salzman and W. T. Newsome. 1994. Neural mechanisms for forming a perceptual decision. *Science (New York, N.Y.)* 264, 5156 (1994), 231–237.

[160] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner. 2010. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems.* 1947–1950.

[161] J. Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

[162] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. V. Campenhout. 2008. Compact hardware liquid state machines on FPGA for real-time speech recognition. *Neural Networks* 21, 2 (2008), 511–523.

[163] S. Sen, S. Venkataramani, and A. Raghunathan. 2017. Approximate computing for spiking neural networks. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'17).* 193–198.

[164] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa et al. 2009. CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking. *IEEE Transactions on Neural Networks* 20, 9 (2009), 1417–1438.

[165] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. 2007. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 3 (2007), 411–426.

[166] M. A. J. Sethi, F. A. Hussin, and N. H. Hamid. 2017. Bio-inspired fault tolerant network on chip. *Integration: The VLSI Journal* 58 (2017), 155–166.

[167] S. Sheik, S. Paul, C. Augustine, C. Kothapalli, M. M. Khellah, G. Cauwenberghs, and E. Neftci. 2016. Synaptic sampling in hardware spiking neural networks. In *Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS'16)*. 2090–2093.

[168] W. Singer. 1999. Neuronal synchrony: A versatile code for the definition of relations? *Neuron* 24, 1 (1999), 49–65, 111–125.

[169] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. 2018. HATS: Histograms of averaged time surfaces for robust event-based object classification. arXiv:1803.07913.

[170] S. Song, K. D. Miller, and L. F. Abbott. 2000. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience* 3, 9 (2000), 919–926.

[171] G. Srinivasan, A. Sengupta, and K. Roy. 2016. Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip STDP learning. *Scientific Reports* 6 (2016), 29545.

[172] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15 (2014), 1929–1958.

[173] I. Sugiarto, P. Campos, N. Dahir, G. Tempesti, and S. Furber. 2017. Optimized task graph mapping on a many-core neuromorphic supercomputer. Retrieved March 14, 2019 from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041212528&doi=10.1109%2fHPEC.2017.8091066&partnerID=40&md5=4e039afff2e11e5dac9a1510996ac487.

[174] X. Sun, S. Yin, X. Peng, R. Liu, J. Sun Seo, and S. Yu. 2018. XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks. In *Proceedings of the 2018 Design, Automation, and Test in Europe Conference and Exhibition (DATE'18)*. 1423–1428.

[175] M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume et al. 2011. Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction. In *Proceedings of the 2011 International Electron Devices Meeting*. 4.4.1–4.4.4.

[176] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat et al. 2013. Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Transactions on Electron Devices* 60, 7 (2013), 2402–2409.

[177] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang. 2017. Binary convolutional neural network on RRAM. In *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. 782–787.

[178] J. C. Tapson, G. K. Cohen, S. Afshar, K. M. Stiefel, Y. Buskila, R. M. Wang, T. J. Hamilton et al. 2013. Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Frontiers in Neuroscience* 7 (2013), 153.

[179] J. C. Thiele, O. Bichler, and A. Dupret. 2018. Event-based, timescale invariant unsupervised online deep learning with STDP. *Frontiers in Computational Neuroscience* 12 (2018), 46.

[180] S. Thorpe, D. Fize, and C. Marlot. 1996. Speed of processing in the human visual system. *Nature* 381, 6582 (1996), 520–522.

[181] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr. 2018. Recent progress in analog memory-based accelerators for deep learning. *Journal of Physics D: Applied Physics* 51, 28 (2018), 283001. DOI: https://doi.org/10.1088/1361-6463/aac8a5

[182] G. Urgese, F. Barchi, E. Macii, and A. Acquaviva. 2016. Optimizing network traffic for spiking neural network simulations on densely interconnected many-core neuromorphic platforms. *IEEE Transactions on Emerging Topics in Computing* 6, 3 (2016), 317–329.

[183] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. 2013. Regularization of neural networks using DropConnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*. Vol. 28. 1058–1066.

[184] K. L. Wang, J. G. Alzate, and P. K. Amiri. 2013. Low-power non-volatile spintronic memory: STT-RAM and beyond. *Journal of Physics D: Applied Physics* 46, 7 (2013), 074003.

[185] Q. Wang, Y. Jin, and P. Li. 2015. General-purpose LSM learning processor architecture and theoretically guided design space exploration. In *Proceedings of the 2015 Biomedical Circuits and Systems Conference (BioCAS'15)*. 1–4.

[186] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu et al. 2016. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nature Materials* 16 (2016), 101.

[187] P. J. Werbos. 1990. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE* 78, 10 (1990), 1550–1560.

[188] T. Werner, E. Vianello, O. Bichler, D. Garbin, D. Cattaert, B. Yvert, B. D. Salvo et al. 2016. Spiking neural networks based on OxRAM synapses for real-time unsupervised spike sorting. *Frontiers in Neuroscience* 10 (2016), 474.

[189] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G.-Y. Wei. 2017. 14.3 A 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications. In *Proceedings of the 2017 IEEE International Solid-State Circuits Conference (ISSCC'17)*. 242–243.

[190] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi. 2017. Spatio-temporal backpropagation for training high-performance spiking neural networks. arXiv:1706.02609.

[191] S. Yang, J. Wang, B. Deng, C. Liu, H. Li, C. Fietkiewicz, and K. A. Loparo. 2018. Real-time neuromorphic system for large-scale conductance-based spiking neural networks. *IEEE Transactions on Cybernetics*. E-pub ahead of print.

[192] S. Yin, D. Kadetotad, B. Yan, C. Song, Y. Chen, C. Chakrabarti, and J. Sun Seo. 2017. Low-power neuromorphic speech recognition engine with coarse-grain sparsity. In *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC'17)*. 111–114.

[193] S. Yin, S. K. Venkataramanaiah, G. K. Chen, R. Krishnamurthy, Y. Cao, C. Chakrabarti, and J. Sun Seo. 2017. Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations. arXiv:1709-06206.

[194] A. Yousefzadeh, T. Masquelier, T. Serrano-Gotarredona, and B. Linares-Barranco. 2017. Hardware implementation of convolutional STDP for on-line visual feature learning. In *Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS'17)*. 1–4.

[195] A. Yousefzadeh, G. Orchard, E. Stromatias, T. Serrano-Gotarredona, and B. Linares-Barranco. 2018. Hybrid neural network, an efficient low-power digital hardware implementation of event-based artificial neural network. In *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS'18)*. 1–5.

[196] Q. Yu, H. Li, and K. C. Tan. 2018. Spike timing or rate? Neurons learn to make decisions for both through threshold-driven plasticity. *IEEE Transactions on Cybernetics*. E-pub ahead of print.

[197] Q. Yu, H. Tang, K. C. Tan, and H. Li. 2013. Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns. *PLOS ONE* 8, 11 (2013), e78318.

[198] D. Zambrano and S. M. Bohte. 2016. Fast and efficient asynchronous neural computation with adapting spiking neural networks. arXiv:1609-02053.

[199] F. Zenke and S. Ganguli. 2017. SuperSpike: Supervised learning in multi-layer spiking neural networks. arXiv:1705.11146.

[200] N. Zheng and P. Mazumder. 2018. Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity. *IEEE Transactions on Neural Networks and Learning Systems* 29, 9 (2018), 4287–4302.

[201] N. Zheng and P. Mazumder. 2018. A low-power hardware architecture for on-line supervised learning in multi-layer spiking neural networks. In *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS'18)*. 1–5.

[202] Y. Zheng, S. Li, R. Yan, H. Tang, and K. C. Tan. 2018. Sparse temporal encoding of visual features for robust object recognition by spiking neurons. *IEEE Transactions on Neural Networks and Learning Systems* 29, 12 (2018), 5823–5933.

[203] Y. Zhengkun and Z. Yilei. 2017. Recognizing tactile surface roughness with a biomimetic fingertip: A soft neuromorphic approach. *Neurocomputing* 244 (2017), 102–111.

[204] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. 2018. The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robotics and Automation Letters* 3, 3 (2018), 2032–2039.

[205] A. M. Zyarah, N. Soures, L. Hays, R. B. Jacobs-Gedrim, S. Agarwal, M. Marinella, and D. Kudithipudi. 2017. Ziksa: On-chip learning accelerator with memristor crossbars for multilevel neural networks. In *Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS'17)*. 1–4.

[206] Semiconductor Industry Association. 2015. Rebooting the IT Revolution: A Call to Action. Retrieved March 14, 2019 from https://www.semiconductors.org/?s=rebooting+the+IT+revolution%3A+a+call+to+action.

[207] J. Park, J. Lee, and D. Jeon. 2019. 7.6 A 65nm 236.5nJ/classification neuromorphic processor with 7.5% energy overhead on-chip learning using direct spike-only feedback. In *Proceedings of the 2019 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC'19)*. 140–141.

Received June 2018; revised November 2018; accepted December 2018