Microprocessors and Microsystems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Spike-based neuromorphic computing: An overview from bio-inspiration to hardware architectures and learning mechanisms☆

Anteneh Gebregiorgis [a,*], Amirreza Yousefzadeh [b], Sherif Eissa [c], Muhammad Ali Siddiqi [d], Charlotte Frenkel [a], Friedemann Zenke [e], Sander Bohte [f,g], Abdulqader Nael Mahmoud [h], Anup Das [i], Said Hamdioui [a], Henk Corporaal [c], Federico Corradi [c,*]

[a] Delft University of Technology, The Netherlands
[b] University of Twente, The Netherlands
[c] Eindhoven University of Technology, The Netherlands
[d] Lahore University of Management Sciences, Pakistan
[e] Friedrich Miescher Institute for Biomedical Research, Switzerland
[f] Centrum Wiskunde & Informatica, The Netherlands
[g] University of Amsterdam, The Netherlands
[h] NXP Semiconductors, The Netherlands
[i] Drexel University, USA

ARTICLE INFO

Keywords:
Neuromorphic computing
Spiking neural networks
Hardware architectures
Learning mechanisms
Computing-in-memory
Emerging technologies
Artificial intelligence

ABSTRACT

The endeavor to emulate the extraordinary efficiency and adaptability inherent in the human brain via spike-based neuromorphic computing presents significant potential across a diverse array of applications. The attainment of this objective necessitates the translation of biological principles into artificial systems, a task that continues to pose a complex challenge requiring a profound comprehension of the mechanisms by which neural systems produce robust computational outcomes. This tutorial paper provides a comprehensive overview of the foundational concepts and emerging design trends in spike-based neuromorphic computing, covering advances from materials and circuits to hardware architectures and learning mechanisms. It begins with an examination of key aspects of brain biology and their influence on neuromorphic design, followed by a brief discussion of biologically plausible neuron and synapse models. The paper then defines the core principles and defining attributes of neuromorphic computing, highlighting the trade-offs and design choices underlying current implementations. Building on these foundations, it explores the critical properties of neuromorphic systems, surveys a variety of learning algorithms, and reviews hardware-level realizations of bioinspired neurons and synapses. Subsequent sections discuss state-of-the-art spiking neural network architectures, mapping and compilation strategies, and representative application domains. By providing this end-to-end perspective, the article aims to guide the development of future neuromorphic systems that more closely emulate brain efficiency, scalability, and resilience.

1. Introduction

The human brain is remarkably efficient at processing information, consuming only about 20 W, roughly the power of a dim light bulb, while enabling perception, reasoning, and learning. In contrast, modern artificial neural networks require orders of magnitude more energy to perform and train on similar tasks. For instance, large-scale models such as GPT-3 or GPT-4 require thousands of high-end GPUs, with each GPU (e.g., NVIDIA A100 or H100) consuming 300–700 W under load. Training such models can consume several gigawatt-hours (GWh) of electricity, comparable to the annual energy usage of hundreds of households. Whereas such models depend on massive centralized computation and energy resources, biological systems achieve can handle complex and noisy data, learn rapidly, and adapt to new situations while operating in a highly parallel and distributed manner. Ongoing research aims to bridge this efficiency gap through advances in information theory,

☆ This article is part of a Special issue entitled: 'EuroMicro' published in Microprocessors and Microsystems.

* Corresponding authors.
E-mail addresses: a.gebregiorgis@tudelft.nl (A. Gebregiorgis), f.corradi@tue.nl (F. Corradi).

https://doi.org/10.1016/j.micpro.2025.105240
Received 28 August 2025; Received in revised form 18 October 2025; Accepted 15 December 2025
Available online 20 December 2025

learning algorithms, emerging materials, and specialized neuromorphic hardware architectures for brain-inspired computing.

Neuromorphic research moves beyond our current notion of artificial intelligence (AI) because it aims at providing computing systems that can scale to the level of intelligence and efficiency of the brain by following equivalent operating processes. Historically, the term "neuromorphic" was coined by Carver Mead, an engineering professor at the California Institute of Technology, who pioneered the field and helped establish its foundations. Mead's work in the 1980s focused on developing analog circuits that could simulate the behavior of neurons and synapses. He demonstrated that these circuits could perform computations more efficiently than traditional digital circuits [1]. This led to the first neuromorphic chips, such as the "analog neural network" chip, developed by Mead and his team in 1989. The main driver behind this research was the Boltzmann statistics of ionic and electronic channel transport, which provide isomorphic physical foundations with electronic hardware devices [2]. Over the years, neuromorphic engineering has significantly broadened to encompass various research areas, including the creation of novel materials that mimic neuronal and synaptic behaviors, the design of digital and mixed-signal neuromorphic circuits, neuromorphic sensors, and specialized computing architectures. With these advances, neuromorphic engineering has the potential to transform computing by enabling efficient and robust systems modeled on the structure and function of the brain.

Neuromorphic systems replicate key brain features such as massive parallelism, spike-based communication, and distributed in-memory computation [3]. They also integrate sensory systems that follow biological encoding principles, enabling low-power, low-latency, and event-driven sensing, as in the silicon retina [4], cochlea [5], touch [6], and vestibular sensors [7]. Prominent neuromorphic computing architectures include BrainScaleS [8], SpiNNaker [9], NeuroGrid [10], TrueNorth [11], Tianji [12], Loihi [13], ODIN [14], PRIME [15], DYNAPs [16], and μBrain [17]. All these architectures, inspired by spike-based computation, demonstrate energy-efficient and low-latency neural processing, illustrating the potential of neuromorphic hardware for scalable and adaptive computing.

Driven by these outstanding research activities, several surveys and tutorials related to neuromorphic computing and engineering have been published: some works have presented a generic overview of neuromorphic computing [18,19], while others have focused on extending the concept of neuromorphic physical computing with memristor-based and in-memory computing strategies [20,21]. In addition, various survey papers are focused on hardware architecture concepts [22,23], and learning algorithm [24], while some others focused on spike coding, stereo vision, and photonics-based neuromorphic computing [25,26]. A full stack review was provided in [27]. It included emerging materials for neuromorphic systems, neuromorphic hardware implementations with their related algorithms, and learning techniques [27]. However, the literature needs a comprehensive tutorial paper on neuromorphic computing, and this tutorial paper aims to close this gap by providing a coherent and complete review of neuromorphic computing systems, starting from the basics of biological processes to emerging devices, circuits, and computing architectures. More specifically, we highlight the commonalities among the various research lines and focus on the most common SNNs algorithms and systems. Therefore, this tutorial paper first introduces the fundamentals of brain biology and its inspiring features in Section 2. Then, different biological plausible neuron and synaptic models of neuromorphic systems are presented in Section 3 followed by the detailed discussion on the fundamental properties of neuromorphic systems (Section 4), learning algorithms (Section 5), hardware implementation of bio-plausible neurons and synapses (Section 6) as well as hardware architecture for spiking neural networks in Section 7. Finally, the mapping and compilation for neuromorphic systems, deployment and applications of spiking neural networks are presented in Sections 8 and 9, respectively. The tutorial is concluded by highlighting the challenges and future directions of brain-inspired neuromorphic computing in Section 10.

2. Fundamentals of brain's biology

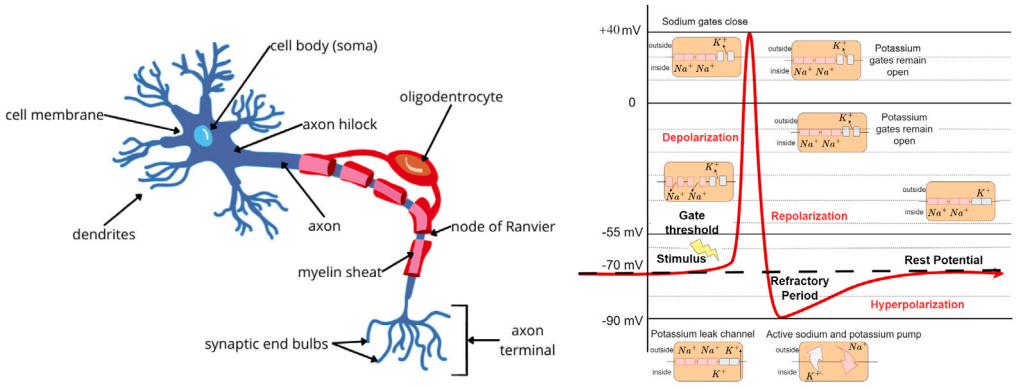
Brains are found across all animal species. They come in various sizes, weighing from a few micrograms (e.g., fruit fly) to almost ten kilograms (e.g., whales), and they contain many neurons: from a few thousands to many billions of neurons interconnected by thousand to trillions of synapses. The brain's primary way of computation and communication is through action potentials, or "spikes". Spiking activity patterns are the basis of information processing in biological neural systems. Much research has been devoted to linking the brain's spiking activity with the representation of the physical world as observed through the senses. For example, the spike patterns of activity relate to muscle activations [28], working memories [29], spatial navigation [30], abstract thinking as the perception of the number of items (i.e., numerosity) [31], decision-making [32], among many others. Unlike artificial intelligence systems, the brain's computation is inseparable from its physical structure. Thus, this section focuses on the biology that inspires neuromorphic systems, mainly spike generation and communication, and describes the fundamental principles of neurons, synapses, and their physical properties. The remainder of this section is organized as follows: Section 2.1 presents the basics of biological neurons followed by a detailed discussion on biological synapses in Section 2.2.

2.1. Biological neuron

The essential elements of any natural neural system are the neuron soma and its synapses. Neurons are electrically excitable cells whose behavior is governed by the membrane potential, i.e., the potential difference across their cell membrane maintained by ionic concentrations inside and outside the cell. This membrane potential provides the basis for signal generation and propagation, enabling neurons to integrate incoming inputs and communicate through action potentials. Neurons are the computational elements and they integrate information coming from the synapses, which, in turn, connect neurons and enable communication using neurotransmitter release or direct electrical stimulation. Neurons are basic functional cells of the nervous system. Their anatomical structure can be simplified as in Fig. 1a. A neuron consists of three main parts:

- **The cell body**, a.k.a soma: it contains the cell nucleus and other cell organelles. The boundary of the cell body is called the membrane;
- **The dendrites**: these are short, branching structures that extend from the soma and they act as receivers for incoming signals (action potentials/spikes) from other neurons;
- **The axon**: a structure that emerges from the soma like a single cable from a region called the axon hillock and branches out at the very end into what are called the axon terminals farthest from the soma [33]. It carries electrical signals (action-potentials/spikes) to other neurons. Axons have a coating known as the myelin sheath, which increases the speed at which information is transmitted along them. In general, axons comprise of multiple long myelinated sections separated by short gaps called nodes of Ranvier [34]. The action potential is propagated by jumping from one node to the next. In addition, Ranvier's nodes are rich in ion channels to mediate sodium and chloride exchange, thus supporting fast propagation of the action potential. At each node, the action potential is regenerated, enabling the action potential to travel along the fiber.

Neurons are electrically excitable cells whose function depends on the membrane potential, created by ionic concentration differences across the cell membrane (Fig. 1b). Key ions include Na^+ and K^+, which set the resting potential, and Ca^{2+}, which mediates synaptic transmission [35]. Since ions cannot freely cross the membrane, specialized ion



(a) Neuron's anatomy (b) Action Potential

Fig. 1. (a) Neuron's anatomical structure illustrating various parts such as cell body, axon and axon terminal with synaptic end bulbs (b) Action potential with illustration on polarization as well as spike generation and transfer.

channels and pumps regulate their flow. At rest, neurons maintain a potential of about –70 mV through the sodium–potassium pump and leak channels. When the potential becomes more negative, the neuron is *hyperpolarized*; when it becomes more positive, it is *depolarized* and ready to fire in response to incoming signals. Potassium K+ and Sodium Na+ channels open and close depending on the voltage across the membrane as detailed in Fig. 1b.

An action potential is a rapid rise and fall of the membrane potential following a characteristic pattern (Fig. 1b). When sufficient current depolarizes the membrane beyond a threshold (typically –55 mV), voltage-gated ion channels open, allowing a rapid influx of positive ions and driving the potential up to about +40 mV [36]. This transient voltage spike propagates along the axon from the hillock to the terminal, triggering communication with other neurons. If depolarization remains below threshold, no spike occurs, and the membrane potential gradually returns to its resting value (~ –70 mV) via sodium–potassium pump activity.

Neural systems contain many types of cells beyond neurons. Among them, glial cells (or neuroglia) are non-neuronal cells that provide essential support and regulation within the nervous system. Major classes include astrocytes, oligodendrocytes, microglia, and Schwann cells. Astrocytes are the most abundant, offering structural and metabolic support to neurons, maintaining extracellular ion balance, and participating in tissue repair. Oligodendrocytes in the central nervous system (CNS) and Schwann cells in the peripheral nervous system (PNS) produce myelin, the insulating sheath that enables rapid electrical conduction along axons. Microglia act as resident immune cells of the nervous system, clearing debris and responding to injury or infection. Although glial cells do not generate action potentials like neurons, recent evidence suggests they exhibit computational and signaling roles. For instance, astrocytes communicate bidirectionally with neurons to modulate synaptic transmission [37], and oligodendrocytes can influence neuronal excitability and plasticity [38]. Although the exact mechanisms underlying these glial computations remain under investigation, it is clear that glial cells are indispensable to maintain, regulate, and potentially process information within the nervous system.

2.2. Biological synapses

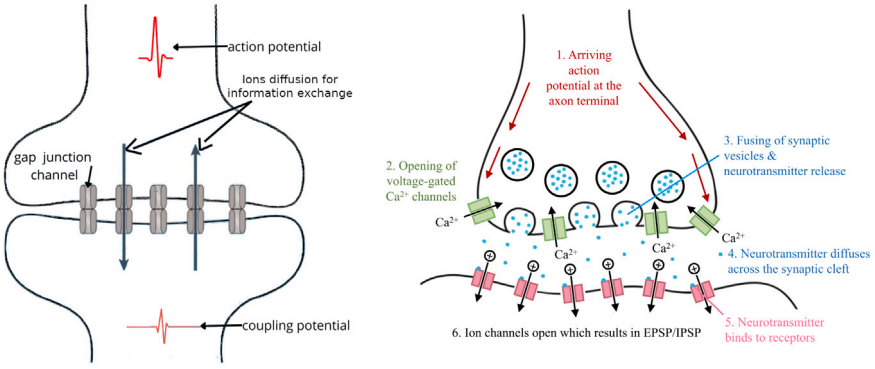
Synapses are sites at which information is carried, from the presynaptic neuron (which sends the action potential) to the postsynaptic neuron (which receives the action potential). A single axon can have multiple branches, allowing it to connect to various postsynaptic cells through synapses as shown by the synaptic end bulbs at the neuron axon terminal in Fig. 1a. Similarly, a neuron can receive thousands of synaptic inputs from many different presynaptic neurons through synaptic connections on its dendrites (indicated by the dendrites terminals in Fig. 1a). Synaptic transmission can be either electrical or chemical, as shown in Fig. 2:

- **Electrical synapses** (shown in Fig. 2a) are specialized direct physical connections between presynaptic and postsynaptic neurons. The structures that provide these connections are called gap junctions, which allow various chemicals and ions to flow directly from one neuron into another for information exchange. When the membrane potential of one cell changes, ions diffuse from one cell to the next and depolarize or hyperpolarize the postsynaptic cell as shown in Fig. 1b.
- **Chemical synapses** (shown in Fig. 2b), on the other hand, are crucial for biological computations. In chemical synapses, there is no direct physical connection between neurons like electrical synapses as they transmit information in the form of chemical messengers called neurotransmitters. Electrical synapses transmit signals faster than chemical synapses. They are essential in neural systems that need fast responses, e.g., defensive reflexes to help an organism escape from a predator. Unlike electrical synapses, chemical synapses lack in signal amplification, as the signal at the postsynaptic neuron is typically the same or smaller in strength than that of the presynaptic neuron [39]. However, the signal of chemical synapses can be modified, which makes it an essential element of memory and learning. Moreover, chemical synapses can have different effects, such as an excitatory or inhibitory effect on the postsynaptic neurons.

Synaptic plasticity it the ability of synapses to strengthen or weaken over time and underpins learning and memory in neural systems [40]. Hebb [41] first proposed that co-activated neurons reinforce their connection, a principle formalized in Hebbian learning. Plasticity occurs on multiple timescales: long-term potentiation and depression (LTP/LTD) produce lasting changes in synaptic efficacy, while short-term plasticity (STP) transiently modulates neurotransmitter release through facilitation or depression [42]. A prominent timing-dependent form, spike-timing-dependent plasticity (STDP), adjusts synaptic weights based on the precise timing of pre- and postsynaptic spikes, strengthening connections when the presynaptic spike precedes the postsynaptic one and weakening them otherwise [43]. Modern models extend STDP to include rate, voltage, and neuromodulatory effects, linking local plasticity to higher-level learning behavior [44].

2.3. Conclusions

The brain's computational power arises from the coordinated activity of neurons, synapses, and supporting cells, all operating through



(a) Electrical synapse (b) Chemical Synapse

Fig. 2. Synaptic connectivity and synapse's type (a) Electrical synapses for direct physical connection between neurons and (b) Chemical synapses based on neurotransmitters. Details in Section 3.2.

electrochemical signaling. Understanding how neurons generate spikes, how synapses modulate and transmit signals, and how plasticity enables learning provides the essential biological foundation for neuromorphic engineering and brain-inspired computation.

3. Bio-plausible models of neurons and synapses

A neuron model describes how neurons integrate input spikes and produce output spikes, while a synaptic model characterizes how synapses convert presynaptic action potentials into postsynaptic currents that drive dendritic and somatic integration [45]. Realistic neuron and synapse modeling involves developing phenomenological (i.e., based on observations instead of being derived from first principles or theory) mathematical models and electronic circuits and finding substrate materials that can emulate, with their physical properties, the complex electrical and chemical signaling that occurs in biological neural networks.

Because the brain exhibits highly nonlinear, heterogeneous, and adaptive behavior, no single model can capture all aspects of neural computation. As a result, many neuron and synapse models have been developed, each offering a different trade-off between biological realism, computational efficiency, and suitability for hardware implementation. For instance, detailed biophysical models reproduce biological behavior accurately but are computationally heavy, while simplified models facilitate large-scale simulations and neuromorphic hardware implementations at the cost of realism.

This section reviews prominent models of biological neurons and synapses that have been developed during the 20th century. Section 3.1 introduces various neuron models, while Section 3.2 presents models of biological synapses. Finally, Section 3.3 provides an overview of brain-inspired SNN architectures.

3.1. Neuron models

Neuron models differ mainly in their level of abstraction. Biophysically detailed models simulate ion-channel dynamics and membrane conductances, while simplified models capture essential spiking behavior with minimal parameters. The choice of model thus depends on the target application, whether it aims to reproduce biological mechanisms, to perform efficient computation, or to enable hardware realization.

3.1.1. Spike response model

The Spike Response Model (SRM) [46] is the most suitable model to illustrate the essential components of a neuron model and how it interacts with other neurons. The SRM is a general model that describes neuronal dynamics using transfer functions instead of electrical circuit models or differential equations. The SRM model can be expressed as shown in Eq. (1) [46].

$$u(t) = \eta(t - \hat{t}) + \int_0^\infty k(t - \hat{t}, s) I(t - s)\, ds \tag{1}$$

Eq. (1) describes the membrane potential as a function of time t. When the membrane voltage crosses the dynamic threshold, $k(t - \hat{t})$ enforces the firing of the neuron, where $\hat{t}$ is the firing time of the last spike. η and k are post-firing and response parameters to the synaptic input $I(t - s)$.

As shown in Fig. 3 [45], the SRM model consists of five functions or filters:

- Membrane Filter (κ): describes how the neuron membrane responds to an input spike.
- Nonlinearity ($f(u - \theta)$): defines the spiking condition.
- Stochastic Spiking: introduces randomness in firing behavior.
- Spike After-Potential (η): models the reset and refractory period after spiking.
- Moving Threshold (θ_1): models threshold adaptation affecting firing frequency.

Overall, the SRM model provides a flexible and computationally efficient framework to describe and simulate a wide range of neuronal properties and behaviors. However, since it abstracts away the underlying ion-channel mechanisms, it is less biologically detailed compared to conductance-based models such as Hodgkin–Huxley [47], which explicitly model ionic currents but at a significantly higher computational cost.

3.1.2. Linear integrate-and-fire models

Integrate-and-fire models [48] are among the most widely used neuron models due to their simplicity and efficiency in capturing essential neuronal behaviors such as bursting, adaptation, and refractoriness. They are particularly suitable for hardware implementations and SNN applications, offering an excellent trade-off between biological accuracy and computational efficiency.

The Integrate-and-Fire (I&F) model assumes the neuron behaves as a perfect integrator without leakage, accumulating incoming current until the membrane potential reaches a fixed threshold, after which it emits a spike and resets to its resting potential.

The Leaky Integrate-and-Fire (LIF) model extends this concept by including a leakage term, accounting for the natural decay of membrane potential over time. This addition enables a more realistic representation of neuronal behavior while maintaining computational simplicity.

The Adaptive Leaky Integrate-and-Fire (ALIF) model introduces spike-frequency adaptation, mimicking the decreasing firing rate under sustained stimulation by dynamically adjusting its firing threshold. This

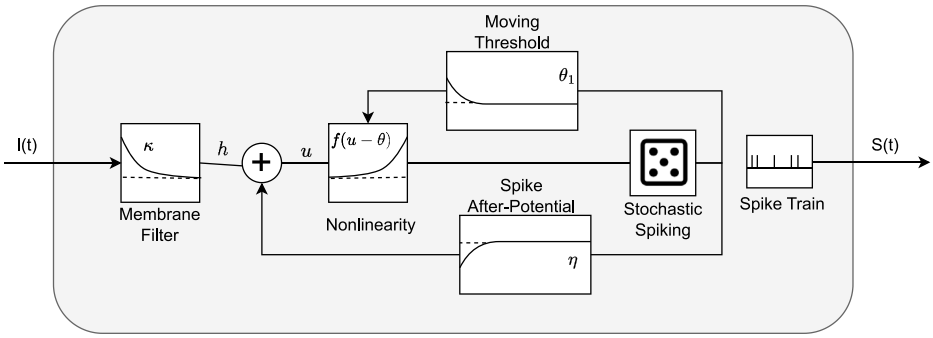


Fig. 3. SRM block diagram as adopted from [45]. Input current is filtered with filter κ. Output spikes are fed back in two distinct ways; they generate contributions (η) on the neuron's potential and (θ_1) on the neuron's threshold potential.

adaptation increases the dynamic range and temporal sensitivity of the neuron model.

When compared to other neuron models, the Hodgkin–Huxley model remains the most biologically accurate, explicitly modeling ion-channel dynamics but at a high computational cost. The Spike Response Model (SRM) provides a more abstract yet flexible framework, capable of capturing temporal dynamics without the biophysical detail of Hodgkin–Huxley. In contrast, the integrate-and-fire family offers the simplest and most hardware-efficient formulation, ideal for large-scale neuromorphic systems, albeit at the expense of detailed channel-level representation. The choice of neuron model therefore depends on the desired balance between biological realism, computational efficiency, and hardware implementability.

A wide spectrum of neuron models exists because each aims to capture neural behavior at a different level of abstraction. The SRM provides an efficient, general framework for temporal integration, the Hodgkin–Huxley model offers unmatched biological detail, and the integrate-and-fire family achieves scalable and efficient computation ideal for neuromorphic hardware. Those models represent the continuum from biophysical realism to hardware efficiency that underpins modern spiking neural network design.

3.2. Synapse models

Synapse models describe how presynaptic activity influences postsynaptic responses. In most cases, they are phenomenological: the synapse is treated as a black box that transforms incoming spikes into a postsynaptic current or potential. The complex biochemical sequence underlying this process—neurotransmitter release, diffusion, and receptor binding—is often abstracted unless those mechanisms are the focus of study [49,50].

Two main types of synapses are modeled in neuroscience and neuromorphic computing: chemical and electrical. Chemical synapses dominate in biology and involve neurotransmitter release across a synaptic cleft, leading to excitatory or inhibitory postsynaptic potentials. Electrical synapses, by contrast, directly connect neurons through gap junctions that allow current to flow bidirectionally. In computational and neuromorphic systems, synapses often combine aspects of both types: analog or resistive couplings emulate electrical behavior, while variable weights capture the modulatory effects of chemical transmission [51,52].

Real synapses are inherently variable. Even under identical stimulation, neurotransmitter release is probabilistic, leading to fluctuations in the strength and timing of the postsynaptic response. This stochasticity can be modeled statistically and plays an important role in neural coding and learning, helping biological and artificial networks remain adaptive and noise-tolerant [50].

A defining property of synapses is their ability to change strength over time—known as *synaptic plasticity*. Plasticity mechanisms can be broadly divided into two families. In *rate-based* models, synaptic change depends on how frequently pre- and postsynaptic neurons fire together, capturing average co-activation over time. In *spike-timing-dependent plasticity* (STDP), the adjustment depends on the precise temporal order of spikes: if a presynaptic neuron fires just before a postsynaptic one, the connection strengthens (potentiation); if it fires after, it weakens (depression) [43,53]. Modern extensions include dependencies on calcium concentration, membrane voltage, firing rate, and neuromodulatory signals [54–56].

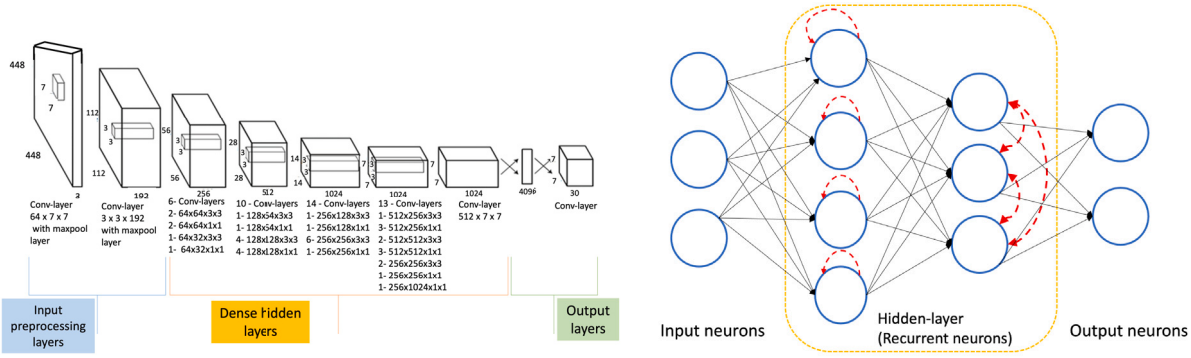
Together, these synaptic mechanisms—stochastic transmission and adaptive plasticity—enable learning and self-organization in spiking neural networks. Rate-based rules describe large-scale population dynamics, while STDP and related mechanisms support event-driven, temporally precise learning that is particularly suitable for neuromorphic hardware. The following section builds on these principles to describe how neurons and synapses combine into complete SNN architectures.

3.3. Spiking Neural Network (SNN) architectures

SNN architectures integrate the neuron and synapse models described in Sections 3.1–3.2 into structured networks capable of computation and learning. Their diversity arises from how neurons are connected, how spikes propagate, and how synaptic plasticity rules, such as those discussed above, modulate connections over time. Similar to conventional Artificial Neural Networks (ANNs), SNNs can be categorized by their connectivity patterns (feed-forward, convolutional, recurrent). However, unlike ANNs, they process information through discrete, temporally precise spike events, giving rise to asynchronous and event-driven computation. This temporal nature introduces new opportunities for energy efficiency and dynamic representation while requiring novel training strategies and hardware support.

Feed-forward SNNs represent the most basic class of architectures, where information flows unidirectionally from input to output [57]. They may include one or more hidden layers: single-layer configurations act as spiking perceptrons, while deeper versions resemble spiking multilayer perceptrons. Each neuron integrates temporal spike inputs and emits an output spike once its membrane potential exceeds a threshold. Stacking multiple layers allows the network to form hierarchical temporal–spatial representations, analogous to deep ANNs, albeit with sparse and event-driven dynamics. Feed-forward SNNs have been widely employed in tasks such as static image recognition and signal classification, particularly when trained via conversion from pre-trained deep neural networks or local learning rules such as Spike-Timing-Dependent Plasticity (STDP). However, the absence of internal feedback limits their ability to capture long-term temporal dependencies.

Convolutional SNNs extend this paradigm by integrating convolutional, pooling, and fully connected layers composed of spiking neurons [58]. By convolving local receptive fields, these architectures exploit spatial correlations in input data while maintaining sparse



(a) YOLO-V6 deep spiking CNN architecture (b) Recurrent SNN

Fig. 4. Example SNN architectures: (a) YOLO-V6 deep spiking CNN architecture for object detection (b) Recurrent spiking neural network architecture.

temporal communication. Convolutional SNNs reduce the number of trainable parameters compared to fully connected networks and are particularly well-suited for event-based vision applications, such as object detection and motion analysis with dynamic vision sensors (DVS). Training strategies include both ANN-to-SNN conversion and direct spike-based learning, enabling real-time visual inference with ultra-low power consumption. Their modular design also facilitates hardware implementation on neuromorphic platforms such as Loihi and DYNAPs (see Section 7). Fig. 4(a) illustrates a CNN inspired deep SNN architecture where convolutional and pooling layers are employed to extract sparse spatiotemporal features and reduce the features the network learns before classification through fully connected spiking layers [59].

Recurrent SNNs incorporate feedback connections among neurons, allowing the network to maintain internal states and process temporal sequences [60]. Such architectures naturally capture the dynamics of biological neural circuits, where recurrent loops and feedback modulate ongoing activity. Recurrent SNNs are capable of sequence learning, working memory, and pattern generation, making them ideal for speech, motor control, and sensory integration tasks. However, their recursive structure complicates learning, often requiring biologically plausible alternatives to backpropagation through time, such as e-prop or local eligibility-trace-based learning.

Among recurrent architectures, the Long Short-Term Memory (LSTM) and gated recurrent models represent important conceptual analogs, introducing gating mechanisms to mitigate gradient-vanishing issues and selectively retain relevant temporal information. Translating these principles to spiking domains—through adaptive thresholds, refractory states, or dynamic synaptic traces—remains an active area of research. Fig. 4(b) illustrates a recurrent SNN where spiking feedback pathways enable temporal context accumulation across layers, facilitating temporal reasoning and prediction.

3.4. Conclusion

Biologically inspired models of neurons and synapses provide the foundation of spiking neural networks, linking neural mechanisms with their computational realizations. From detailed conductance-based formulations to simplified integrate-and-fire abstractions, these models describe how neurons integrate inputs, generate spikes, and adapt over time. Similarly, synaptic models—ranging from deterministic current formulations to stochastic and plasticity-driven dynamics—enable learning and long-term adaptability. Together, they constitute the core building blocks of neuromorphic architectures. Spiking network architectures, in turn, combine spatial processing, temporal dynamics, and sparse communication to form a flexible computational substrate. The choice between feed-forward, convolutional, or recurrent structures depends on the target application, learning paradigm, and hardware constraints, ultimately determining the balance between biological realism, computational efficiency, and scalability.

4. Fundamental properties of neuromorphic computing

Taken literally, "neuromorphic" means systems with the same form, shape, and structure as natural neural systems. Nevertheless, neuromorphic computing systems deviate in form and shape. Today, researchers exploit various devices, circuits, and largely different architectures for constructing sensing and computing systems that mimic some neural computational attributes. And yet, the extent to which a technology can be called neuromorphic is controversial. Nevertheless, we can agree on the *general fundamental* features and principles that need to be researched and understood before we will be able to implement truly artificial neuromorphic systems with silicon and emerging device technologies. Currently, Deep Neural Networks (DNNs) are exceeding the accuracy of biological brains (including the human brain) in several specific domains like video [61] and audio processing [62] and can beat human champions in playing games [63]. However, all of these tasks are performed with digital nanotechnologies without considering many of the main biological features and their related physical restrictions present in natural systems. In nature, biological limitations have steered evolution toward exploiting neural information algorithms capable of coping with the biological neural elements, that are noisy, diverse, and stochastic [64], but that are capable of delivering high energy efficiency. Thus, one of the primary aims of neuromorphic engineering is to understand the principles of neural computation for reaching the brain's energy performance point. This section presents the essential biological features that inspire the development of neuromorphic technologies. As illustrated in Fig. 5, these features are local/in-memory computing, parallelism, non-linearity, asynchronicity, sparsity, continuous learning, probabilistic computing, Scalability and signal representation. These inspiring features are discussed in the following subsections.

4.1. Analog in-memory computing

In the brain, computation happens thanks to the biophysical properties of neurons and synapses. Memory and computation are distributed, and there is little distinction between processing and memory elements. In fact, in biological neural networks, memory traces are found in the state of the synapses and the neurons' membrane potential, in their firing pattern, and in their dynamics [65].

Inspired by these biological principles, neuromorphic in-memory computing emulates the physical properties of neural systems in very-large-scale integration (VLSI) hardware [1] and emerging device technologies [66], where computation occurs either in the memory itself or close to it. By leveraging the characteristics of the medium, similar to those observed in biological systems, these systems enable new computational paradigms that interact with the real world using the inherent physical properties of neural networks.

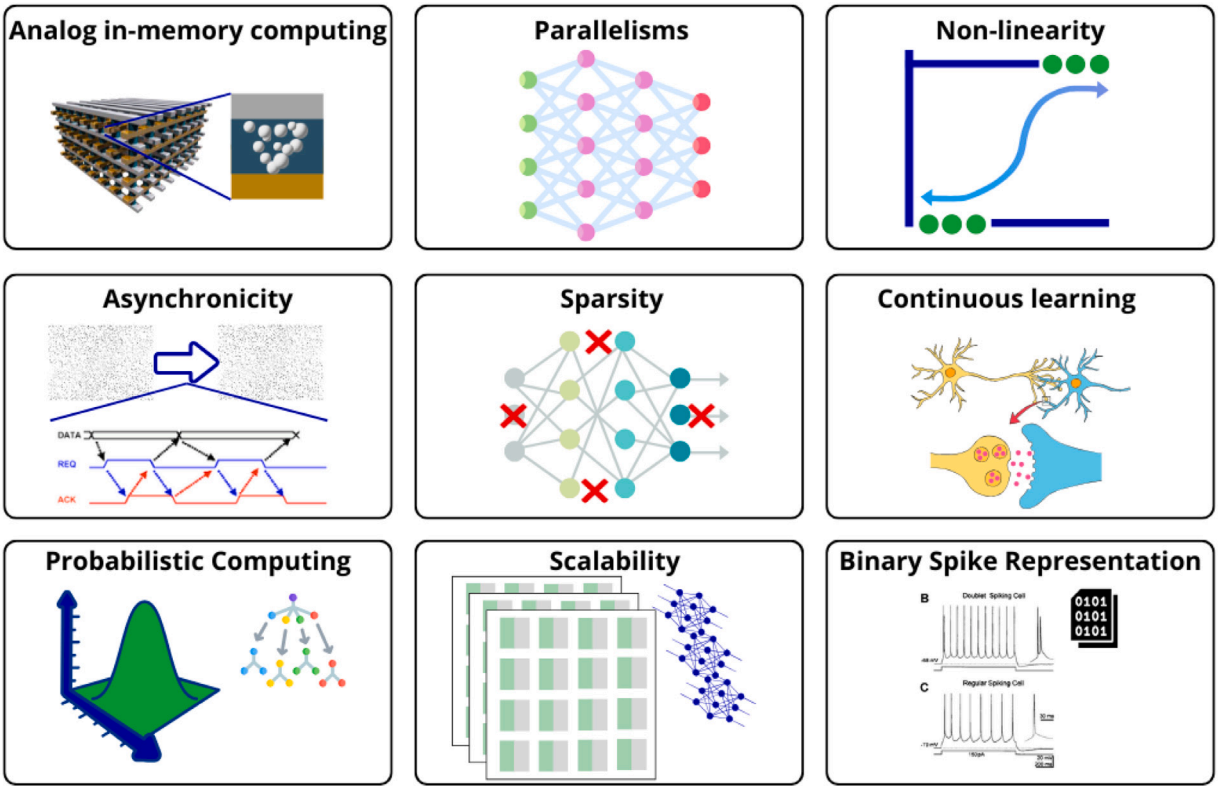


Fig. 5. Attributes of biological neural networks inspire neuromorphic systems, as explained in detail in this section.

More generally, in-memory computing (IMC) refers to a broad range of approaches where computation is performed either directly within the memory or near it. This reduces the need for excessive data movement, which is a major bottleneck in traditional architectures. In the context of deep neural network accelerators, IMC architectures have been proposed to amortize memory access and perform computation on the bit-lines within memory arrays. This technique, applicable in both the analog and digital domains, enhances parallelism and energy efficiency [67], especially for operations such as large-scale vector–matrix multiplication, where improvements in speed, latency, and energy consumption are critical.

4.2. Parallelism

The fabric of the brain is slow. Each neuron fires on average in the range of 1–5 spikes per second. However, thanks to its structure that comprises billions of parallel processing units, brain performance can surpass the performance of many advanced computers. Such neuromorphic systems are typically implemented by means of fine-grain parallelizable systems that can benefit from executing over many small interconnected processing cores (neurons) [68,69]. For many years, improving the speed of a single processing core was the only method to increase the computer's performance. This however is not possible anymore due to the fundamental limitations such as the power, memory and cost wall [70]. Therefore, parallel processing is used as one of the main methods to improve computing performance.

4.3. Non-linearity

The brain is a highly nonlinear system composed of populations of neurons interconnected via feedback loops. Recurrent neural networks, which allow for cycles in the connections, can exhibit varying levels of connectivity. Notably, Douglas [71] introduced the concept of canonical (cortical) microcircuits, an influential theory in neuronal processing. These microcircuits are organized into fundamental neuronal circuits that operate as nonlinear computational units [72]. This organization of microcircuits leads to distinct nonlinear operations in different cortical areas, manifested as specific neural responses [73]. These processes are associated with particular computational roles. For example, many cortical neurons exhibit sigmoid-like response patterns based on stimulus parameters, such as image contrast. A well-known instance is the orientation tuning observed in the primary visual cortex, where neural tuning emerges from the geometric arrangement of the receptive fields of afferent neurons [74]. Another example is the winner-take-all and attractor dynamics used as models for working memory and decision-making in the prefrontal cortex [65], which have been emulated in analog neuromorphic hardware [75,76].

4.4. Asynchronicity

Brains are asynchronous systems. Each neuron works independently from other neurons, without the use of an explicit clock. Still, the signals are propagated as they happen in real-time. Nevertheless, most SNNs are still trained in a synchronous regime, because it is more efficient in our traditional computers (CPUs and GPUs). As a consequence, neurons in the network are required to be evaluated synchronously. This behavior can be a significant bottleneck for scalability. For example, barrier message synchronization in Loihi [69] does not allow for large-scale systems. In contrast, biological neurons do not require explicit synchronization, which allows for efficient scalability. Nevertheless, it is very challenging to obtain robust computation with truly asynchronous systems as we still miss understanding, algorithms, and a formal theory of computation for truly asynchronous computing systems. Additionally, design automation tools for asynchronous systems are lacking behind traditional digital tool-chains, rendering the designs of asynchronous systems harder and mostly driven by academic researchers [77].

4.5. Sparsity (in space and time)

A key property of neuromorphic signal processing is the existence of dimensions where signals exhibit smooth variations. The most straightforward example of such a dimension is time. Neurons in the brain possess short-term memory, allowing them to retain the history of incoming signals. This feature enables the brain to excel in processing natural signals, which are typically highly sparse in time. For instance, although the human brain can efficiently process over 100 frames per second (fps), research has shown that performance drops to below 10 fps when the frames are not temporally correlated [78].

In addition to temporal sparsity, spatial sparsity plays a crucial role in neuromorphic systems. Spatial sparsity refers to the activation of only a small subset of neurons or synapses at any given moment, which reflects the efficient use of neural resources. In the brain, signals are processed locally, meaning that specific neurons or circuits are activated in response to specific features or stimuli, leading to sparsely distributed patterns of activity across the neural population. This spatial sparsity allows for efficient signal representation and reduces energy consumption during processing. For example, neurons in the visual cortex are sensitive to specific orientations of visual stimuli, resulting in sparse activation patterns based on the spatial arrangement of receptive fields.

Furthermore, effective neuromorphic processing relies not only on input-level temporal sparsity but also on context-level temporal sparsity. The brain processes input data by extracting features hierarchically across different stages. If there is insufficient sparsity at each level of feature extraction, it can disrupt the efficient processing of signals. Therefore, both temporal and spatial sparsity are essential for the brain's ability to process complex, high-dimensional data efficiently.

4.6. Continuous learning and plasticity

Online learning, fault tolerance, and adaptability are essential properties of any neural system. Online learning refers to learning and adapting to new inputs and statistics as the system receives information in the physical world, i.e., while further information becomes available, and it refers to a system that updates its knowledge or model continuously as new data comes in, allowing it to learn and adapt in real-time.

Fault tolerance means the system can continue functioning despite faults or damage (e.g., some neurons or synaptic connections are damaged). Finally, adaptability refers to the ability of the system to change and reconfigure itself in response to changing conditions. Moreover, and similarly to online learning, continuous learning emphasizes the ongoing nature of learning without resetting or stopping between tasks. It often refers to lifelong learning where the system retains knowledge from past experiences.

Finally, on-chip Learning refers to learning that occurs directly on the hardware (e.g., a neural chip without the use of external supporting systems as large memories or general purpose computing systems). This usually emphasizes efficient real-time updates with a reduction or elimination of external computation carried in a general purpose computing system.

4.7. Probabilistic computing

Spiking neural networks (SNNs) communicate and process information through action potentials, or *spikes*. These networks utilize specialized neural information processing strategies that include; stochastic recurrent dynamical systems: systems that explore different stable energy states in a probabilistic manner; stochastic resonance: a phenomenon where noise improves signal detection; chaotic systems: systems with unpredictable behavior that can still exhibit underlying patterns. Each of these strategies contributes to the network's ability to handle complex, probabilistic tasks. Some scientists argue that the presence of noise is essential for the correct behavior of the system. For example, the drop-out technique in deep learning is inspired by this feature and improves the neural network's generalization by avoiding overfitting [79]. In summary, the brain is a noisy processing system. Yet, it can process information accurately and efficiently.

The deterministic behavior of modern digital silicon hardware comes at some costs. Highly dense process node silicon (~nm of feature sizes) technologies are noisy. Furthermore, the noise level increases when using a smaller technology node. Allowing a transistor to communicate only binary states (0 and 1) limits the performance of current computers and increases their power consumption. However, the dominance of deterministic computers is the result of the algorithmic requirements. Specifically, deterministic computers (which process binary states—0 and 1—predictably) have become dominant because many algorithms used in modern computing require precise, predictable behavior. These algorithms are designed to operate in environments where every operation has a clear, repeatable outcome. Therefore, deterministic computing ensures that programs run reliably and consistently, which aligns well with the structure of current software and algorithmic needs, despite hardware inefficiencies like noise and power consumption. In contrast, a neuromorphic algorithm does not necessarily require a fully deterministic processing platform to function well. One of the trends in neuromorphic processing is to make noisy analog neurons with the silicon transistors in the ultra-low-power regime (near-threshold) [80] or with entirely new nano-electronic materials [66]. In both cases, the communication between the processing elements (neurons) is done digitally, as the shape of the action potential is stereotypical and it is assumed it does not carry other information than its presence (i.e., binary, is present or not).

4.8. Scalability

Neural systems are intrinsically scalable because they are composed of neurons and synapses that can be replicated and connected in large numbers to form complex networks. This modular organization allows neural systems to scale up or down depending on the computational demands of a particular task. The scalability of neural systems has been recognized in many fields, including artificial intelligence, robotics, and neuroscience. In particular, the development of artificial neural networks has enabled researchers to leverage the scalability of neural systems to perform complex tasks, such as image and speech recognition, with high accuracy and efficiency. For example, in the human brain, the neural cortex is the brain's outer layer, and it plays a critical role in many cognitive functions, such as perception, attention, memory, and language. The cortex is composed of repeating units called cortical columns, which contain thousands of neurons that are arranged in a highly structured manner. Furthermore, these columns are organized hierarchically, with each level of the hierarchy processing increasingly complex features of sensory information.

4.9. Binary spike representation

Signal representation in neural systems can take different forms depending on the level of analysis and the specific neural system being studied. For example, signal representation refers to the way neural systems encode and convey information. Neural signals can be represented in multiple forms using analog representation, digital communication and chemical interactions. Analog representation refers to the continuous changes in the intensity of signals, such as variations in the membrane potential of neurons. Digital communication then refers to the use of a stereotypical pulse signal, i.e. the action potentials (spike), which are binary events (either occurring or not). Finally, many chemical interactions, such as neurotransmitters and other chemical substances that mediate communication between neurons at synapses, modulate signal transmission and processing. Each type

of representation contributes to the complex functionality of neural systems.

At the cellular level, the basic neural processing unit is the neuron, which communicates with other neurons through chemical and electrical signals. These signals are generated by the flow of ions across the neuron's membrane, which creates a voltage difference or electrical potential that can be measured as an analog signal. These analog signals are then converted into digital signals by firing action potentials, which are discrete electrical pulses that neurons use to transmit information.

At the network level, neural systems process information through the interactions of large populations of neurons that are interconnected by synapses. These synapses can modulate the strength of the connections between neurons through various mechanisms, including changes in the release of neurotransmitters or the expression of ion channels. These chemical interactions can also encode information digitally, with the strength of the connection between two neurons representing the presence or absence of a particular feature or pattern of information.

Overall, neural systems rely on a combination of analog and digital representations and chemical interactions to encode and process information. This complex interplay of signals and interactions gives neural systems their remarkable ability to perform complex computations and learn from experience.

4.10. Conclusions

Neuromorphic systems implement the above properties, *in/near-memory computing, parallelism, non-linearity, asynchronicity, sparsity, continuous learning, probabilistic computing, scalability, and spike-based representation*, within concrete SNN architectures (feed-forward, convolutional, recurrent). Feed-forward SNNs leverage temporal sparsity for efficient event-driven pipelines; convolutional SNNs add locality and weight sharing for scalable feature extraction; recurrent SNNs introduce feedback and state for temporal reasoning and online adaptation.

Unlike conventional deterministic processors optimized for dense synchronous arithmetic, neuromorphic substrates co-design device physics, circuits, and network organization around sparse, event-driven computation. This makes them naturally suited for low-power, real-time, and adaptive workloads operating in noisy, uncertain environments. Although today's digital accelerators dominate many benchmark tasks, neuromorphic computing offers a complementary path to energy efficient, scalable, and adaptive intelligence by embodying the above principles in hardware, including silicon CMOS and emerging devices, and aligning the learning rules with local mechanisms on the chip.

5. Learning algorithms for spiking neural networks

Over two decades ago Wolfgang Maass showed in a seminal work that SNNs are universal approximators [81]. Intriguingly, he also argued that spiking neural networks can be computationally more powerful than classical neural networks, since fewer neurons are needed to perform an equivalent computation. Unlocking this capacity, however, requires learning algorithms to instantiate suitable parameter configurations that yield the desired network function. Therefore, this section discusses bio-inspired, spike-based learning algorithms for spiking neural network systems. First the basics of learning algorithms are introduced in Section 5.1. Then, a brief summary of the contemporary deep learning approaches of spiking neural networks is presented in Section 5.2. Finally, the future directions in learning algorithms for spiking neural networks are presented in Section 5.4.

5.1. Learning algorithms

Learning algorithms adjust the network parameters with the goal to achieve a specific behavior or computation in the network. The degree to which the networks show this behavior is calculated using a *loss function*, as in conventional ANNs. Most existing learning algorithms in machine learning are gradient-based and rely on the gradient of the loss, either calculated explicitly, like in gradient descent, or implicitly, like in evolutionary computation. However, most spiking neuron models rely on a discontinuous activation function to implement the spiking threshold (cf. 3.1), which is not smoothly differentiable, and thus poses a challenge for gradient-based optimization in SNNs. Over the past decades, much research effort has therefore been dedicated to evolutionary algorithms and phenomenological plasticity models constrained by plasticity experiments such as STDP.

Phenomenological synaptic plasticity models, such as STDP and Hebbian learning, are commonly used to describe the results of plasticity induction experiments. However, these models are often under-constrained and typically cannot be derived from a sensible loss-function. Thus, their direct application to networks often results in run-away activity rather than useful network behavior. This can be addressed when augmented with additional loss-related terms. For instance, reward modulation STDP can implement reinforcement learning [56,82], or adding normalization terms to Hebbian learning can implement forms of unsupervised learning like Principal Component Analysis (PCA) [83] or blind source separation [84]. For example, Oja's rule is given by:

$$\frac{dw_j}{dt} = \eta x_j y - y^2 w_j \tag{2}$$

where x_i is the neuronal activity of the input neuron i and y is the postsynaptic activity assuming a linear neuron model [83]. The first term in which x and y appear detects coincidences of pre and post-synaptic activity. It makes the learning rule a Hebbian learning rule. However, it is easy to see that with only this first term, the learning rule would be unstable as weights of a highly active neuron y would keep increasing for a positive input x. The second term involving y^2 prevents such run-away activity and ensures that the weight vector $\vec{w}$ remains normalized. Assuming a linear neuron model $y = \sum_j w_j x_j$ and zero mean input the weight vector aligns with the first principal component of the data $\vec{x}$. While the above learning rule can be extended to a population of neurons which, given suitable lateral inhibition, can learn to cover the principal subspace of the data [85], phenomenological local learning rules optimize the activity of single neurons or populations of neurons. While this allows them to effectively implement linear decompositions like PCA, they do not have the capability of optimizing networks end-to-end due to the lack of credit assignment mechanisms, and empirically most of the corresponding local learning rules did not result in interesting computational functions in deep SNNs, although there are some exceptions [86].

To resolve the issue of non-differentiability in more complex SNNs, researchers have explored evolutionary strategies to tune parameters. This approach has proven successful at generating effective small SNNs [87], regardless of the nature of the loss function. However, as with all evolutionary approaches, they are hard to scale to larger problems and deep SNNs.

5.2. Deep learning in SNNs

In deep learning, networks are trained end-to-end using the error-backpropagation algorithm. Combined with unprecedented compute power and the development of large datasets, this algorithm has fueled the deep learning revolution over the past decade.

5.2.1. Backpropagation

Backpropagation of error is an algorithm that allows efficient computation of the gradient in multi-layer networks. Toward that end, backpropagation solves the spatial credit assignment problem, whereby it assigns to every neuron in a multi-layer network a neuron-specific contribution to the error at the output of the network. Importantly, it does so efficiently by prescribing a recursively strategy to compute neuron-specific errors from which loss gradients for all parameters in the network can be easily calculated. While backpropagation is perfectly suited for feedforward networks in which the output errors are back-propagated layer-by-layer along the connections of the network, training Recurrent Neural Networks (RNNs), in which the neuronal activity depends not only on lower layers, but also on activity in previous time steps, e.g., through cycles in the computational graph, require a slightly altered strategy. The self-recursion of the neuronal activity creates a temporal credit assignment problem which is solved by Back-Propagation Through Time (BPTT). BPTT effectively unrolls the network in time and treats each timestep akin to a layer in standard back-propagation, only now weights across these "layers" are tied across time steps. Both standard backpropagation and BPTT are effectively algorithms for applying the chain rule of differentiation to multi-layer networks. For this strategy to work, all functions encountered in the network have to be differentiable.

Training large and complex SNNs also requires end-to-end optimization, but the limited differentiability of SNNs due to their spiking mechanisms, precludes doing so in a straightforward manner using backpropagation. To overcome this limitation in SNNs different strategies have been devised over the years that either convert conventionally trained ANNs to SNNs, perform gradient descent on the firing times of the SNNs, or surrogate gradients. In the following we describe each strategy in more detail.

5.2.2. ANN-to-SNN conversion

To overcome limitations related to differentiability of SNNs, many approaches have focused on converting ANNs trained with error-backpropagation into SNNs, e.g., [88]. Typically, these studies used rate-based approximations and do not capitalize on spiking neuron specific capabilities like timing and sparse spiking, which renders them computationally less efficient [89].

5.2.3. Gradient descent on spike timings

In early work, like SpikeProp [90], error-backpropagation was applied directly to the network of spiking neurons. Here, the spike-discontinuity was overcome by linearizing the gradient at the time of spiking, allowing the individual spike-times in such spiking networks to be trained to desired input–output patterns, illustrated in Fig. 6a. The linearization around spike-times, however, made this approach hard to extend to multiple spikes and sensitive to vanishing spiking — the effect where a neuron that is silenced due to decreased weights will never become activated again. More recent algorithms address the problem of single spike-times and allow computing exact gradients on spike times in large networks [91] using the adjoint method for the Leaky Integrate-and-Fire (LIF) neurons. The effectiveness of the latter approach has been demonstrated on a number of tasks, including MNIST and Yin-Yang. Still, the adjoint method cannot be computed analytically for more complex spiking neuron models, and its inherent dependence on spike timing and the resulting inability to deal with gaining or losing spikes in the hidden layers remains an unresolved challenge [92]. Similarly, Lee et al. [93] apply the implicit function theorem to SNNs and demonstrate that well-defined gradients with respect to firing times can be calculated for SNNs. Implemented in the forward propagation (FP) algorithm, such firing time approaches are able to learn tasks like Yin-Yang and can show fast convergence. While timing based-approaches hold promise for efficient event-based gradient computation, their applicability to large-scale networks is still an unresolved problem and their inherent dependence on spike timing and the resulting inability to deal with gaining or losing spikes in the hidden layers remains challenging [92].

5.2.4. Surrogate gradients

An alternative approach uses the neuron's membrane potential rather than the spike time to calculate the gradient. This Surrogate Gradient (SG) approach replaces the derivative of the discontinuous spike-function with a smooth function of the membrane potential [94]. Using SGs, allows flexibly optimizing any loss function, and no explicit coding assumptions in the hidden layers are mode. This generality allows networks to use spike-timing, rate-coding, or both depending on the task demands [95].

Formally, for a LIF spiking neuron we can model the neuronal dynamics such that the membrane potential u_t updates as $u_t = f(u_{t-1}, x_t, s_{t-1} \parallel W, \tau)$, where W represents the synaptic weights and τ the internal time constants. The membrane potential updates depend on the previous potential, u_{t-1}, the outgoing spike-state $s_{t-1} = \{0, 1\}$ and inputs x_t. A spike, $s_t = 1$, is triggered when the membrane potential u_t crosses a threshold θ from below:

$$s_t = f_s(u_t, \theta) = \begin{cases} 1, & \text{if } u_t \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Finally, upon emitting a spike, the membrane potential is reset to the resting potential u_r: $u_t = (1 - s_t)u_t + u_r s_t$, where usually we use $u_r = 0$. Following this definition, applying BPTT to SNNs amounts to the following: given a prediction $\hat{y}_t$ at timestep t for a training sample (x, y), parameters are optimized by gradient descent to minimize the loss function $\ell_t = \mathcal{L}(y_t, \hat{y}_t)$. This gradient expression is then the sum of the products of the partial gradients:

$$\frac{\partial \ell_t}{\partial W} = \frac{\partial l_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial u_t} \sum_{j=1}^{t} \left(\prod_{m=j}^{t} \frac{\partial u_m}{\partial u_{m-1}} \right) \frac{\partial s_{m-1}}{\partial W} \tag{4}$$

the corresponding computational graph, for a single spiking neuron, is illustrated in Fig. 6b. In this formulation, the partial derivative of spiking $\frac{\partial s_t}{\partial u_t}$ has to go through the discontinuous spiking function. The surrogate gradient approximates this partial derivative term by a well-behaved function $\hat{f}'_s(u_t, \theta)$.

With surrogate gradients, SNNs can be modeled as RNNs using popular deep learning frameworks like PyTorch, Tensorflow and JAX, where the surrogate gradient is implemented in the backward pass of the automatic differentiation functionality. With this technique, learning algorithms for recurrent networks and optimizers developed for conventional ANNs can be directly applied, resulting in greatly increasing performance and accuracy [96] that approaches or exceeds conventional ANNs while relying on sparse communication and low latency. While surrogate gradients have mostly been applied to supervised learning tasks, the approach also extends to self-supervised learning settings [86]. Finally, surrogate gradients work even on analog neuromorphic hardware with in-the-loop learning thereby self-correcting for device mismatch [97].

5.2.5. Complex spiking neuron models

While standard neurons in SNNs have mostly been based on LIF neurons, more complicated spiking neuron models are increasingly being explored. Bellec et al. [98] demonstrated how the inclusion of long adaptation time constants in adaptive LIF neurons improves accuracy in many tasks by expanding the effective memory-window of individual neurons. In related work, Perez et al. [42] found that in general heterogeneity in terms of spiking neuron models in the same network substantially improved task performance. Further, training the internal parameters of spiking neuron models, like the decay time constants of the membrane potential or the adaptation variable, further improves accuracy, likely in part by inducing neural heterogeneity [96]. By endowing connections with Short-Term Plasticity (STP) and training networks of multi-compartment spiking neurons to minimize a feedback-inhibition objective, Keijser et al. [99] recovered biologically observed connectivity properties, thus offering a functional interpretation. Yin et al. [100] also demonstrate how multi-compartment spiking

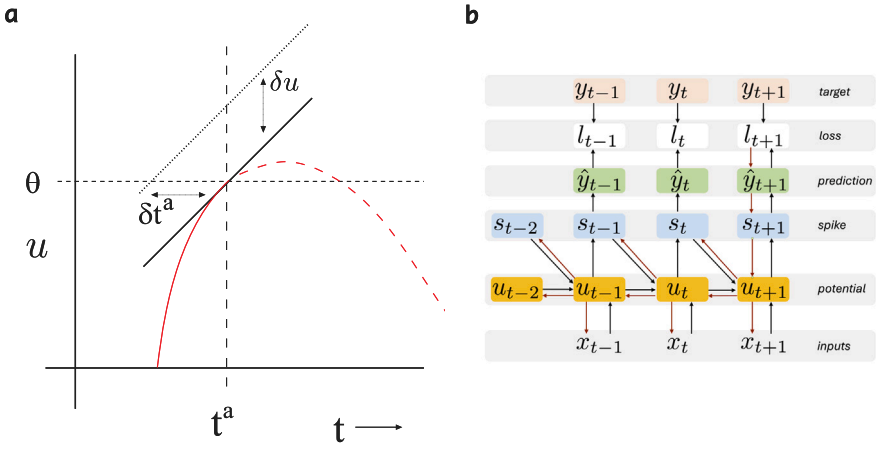


Fig. 6. (a) Linearizing gradient around spike-time: the red potential inside the spiking neuron exceeds the threshold θ from below at time t^a triggering an output spike. Changing the input weights changes the membrane potential u, which in turn changes the spike time; to estimate the change in spike-time we estimate the local slope $\delta t^a/\delta u$. (b) Computational graph linking inputs to outputs for a neuron unrolled in time (black arrows). BPTT traverses the graph in reverse (red arrows) to determine the contributions of each parameter to the losses as measured at every timestep. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

neurons can effectively be used to robustly train large SNNs. Similarly, Harkin et al. [101] used surrogate gradients to train neuron models with complex dendritic morphology. In general however, increasingly more complicated spiking neuron models can interfere with the ability to train the resulting networks. More robust learning rules like Forward-Propagation-Through-Time [100] have been demonstrated to alleviate this problem.

5.3. Online learning

BPTT requires storing the activation history of the entire network for the duration of the entire input sequence before a weight update can be computed. The memory requirements quickly become prohibitive for both large networks and long sequences, especially when a close match with neuromorphic hardware with fine-grained time resolution is needed. For on-chip learning BPTT's memory requirements are usually prohibitive. This limitation has generated substantial interest in online learning rules with a smaller memory footprint. An alternative online learning algorithm which allows to update weights without having to store the activation history is Real-Time Recurrent Learning (RTRL) [102]. However, RTRL has a high computational complexity which again moderates its practical use in the on-chip setting. As a consequence, a number of approximate online learning approaches have been proposed, including SuperSpike [103], eProp [98], DECOLLE [104], Forward-Propagation-Through-Time (FPTT) [100], OSTL [105], and Online-Training-Through-Time (OTTT) [106]. One key feature underlying these approaches is the use of eligibility traces to solve the temporal credit assignment problem. Although successful these methods approximate the gradient as they are approximations of RTRL [107]. Moreover, the notion of eligibility traces do not solve the spatial credit assignment problem. To address this shortcoming, DECOLLE and OSTL extend online learning to deep neural networks by combining spatial back-propagation with temporal forward-propagation through eligibility traces. FPTT relies on another effective approximation allowing it to be applied to deep networks using spatial error-backpropagation, and was shown to be robust for both long-sequence learning and learning in large and deep SNNs on large event-based vision problems like object-localization and classification [100]. Finally, OTTT can conveniently be calculated as three factor Hebbian learning rule, but so far has only been demonstrated for static tasks, like image recognition.

5.4. Conclusions

While there has been much progress in the last years on learning algorithms for SNNs, several important questions remain open. One essential gap is a lack of on-chip learning algorithms that allow neuromorphic systems to learn in the field. Here continual or lifelong learning strategies are needed with performance and stability guarantees. While these are active research areas in the broader deep learning community [108,109], and increasing efforts have also been put into SNNs [110], continual learning remains largely an unresolved problem. Scaling up the size of trainable SNNs is another open question, as current approaches are limited to tens of millions of spiking neurons, whereas popular AI models like large language models (LLMs) are comprised (implicitly) of billions of neurons. Scaling up SNNs to LLMs holds the potential for more energy efficient AI solutions suitable for non-cloud applications when combined with suitable hardware accelerators. A challenge here is also that LLMs are inherently sequential rather than time-continuous like SNNs. Yet, there is active work on smaller SNN-based LLM models [111]. Finally, other active research directions inspired by neurobiology [42,99,100] suggest that more complex synapses, dendrites, and spiking neuron models could increase the computational power of SNNs, where a suitable trade-off will need to be found between the implied memory cost versus other performance metrics.

6. Hardware implementation of spiking neuron and synapse models

Traditionally, neuromorphic engineering focused on emulating the physics of biological neurons and synapses using analog as well as digital circuits using CMOS devices. However, as the field has grown, researchers have begun exploring a more comprehensive range of physical substrates for neuromorphic computing, including memristors, phase-change materials, and organic transistors. These new materials offer unique advantages, such as non-volatility, high scalability, and low power consumption, which would make them well-suited for neuromorphic systems.

This section presents some of the prominent implementations of neural and synaptic models using several technologies, highlighting challenges and differences among these implementations. Thus, analog implementations of neural models are presented in Section 6.1 followed by a discussion on the digital CMOS implementation of neurons in Section 6.2. Finally, Section 6.3 discusses the neuron implementations that leverage emerging Non-Volatile Memory (NVM) devices to model synapses and neurons.

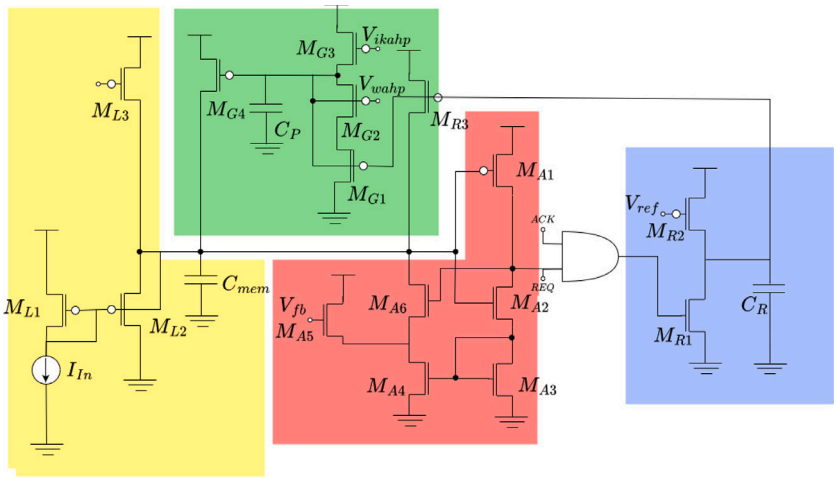


Fig. 7. The LLN circuit comprises a membrane LPF (yellow, M_{L1-L3}), a spike-event generation and positive-feedback element (red, M_{A1-A6}), a reset-refractory pulse generator (blue, $M_{R1,R3}$), and a spike-frequency adaptation LPF (green, M_{G1-G4}). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Source: Reproduced from [80].

6.1. Analog CMOS implementation

Analog CMOS implementations simulate biological neuron models using Metal-Oxide-Silicon Field Effect Transistors (MOSFETs) arranged to model different features of biological neurons and their behavior is parameterized by their design dimensions. Such models can exploit biophysical equivalence between the transport of ions in biological channels and the transport of charge carriers in transistor channels [80]. Analog spiking neurons can emulate complex features of neural computation with high fidelity and low power. Analog neurons are also completely dedicated implementations with no time-multiplexing, offering real-time fully parallel simulation that is scalable and does not suffer from operation load issues/limitations. However, proper calibration of neurons is challenging as they suffer from design variations between them. Hence, analog implementations are suitable for qualitative analysis and large-scale simulations [112] but not for accurate quantitative analysis.

Analog neuron implementations can differ in complexity depending on the complexity of the underlying mathematical model. An analog implementation can be usually split into separate blocks, each implementing a feature or a stage of the neuron model. The first block is typically a temporal integration (synapse) block which converts an incoming action potential into an appropriate post-synaptic input for the neuron's soma. The other blocks implement the neuron's features such as spike/event generation, refractory period mechanism and spike-frequency adaptation.

CMOS neuron implementations have many possible design choices. For example, MOSFETs can operate in sub-threshold region (weak-inversion) where current flows in the transistors by diffusion, or they can operate in above threshold region (strong-inversion) where current flows in the transistors by drift. Input and output signals can also be represented as voltage or current. The analog circuits may operate continuously in a non-clocked fashion or they can implement discrete time signal processing using a switched-capacitor (S-C) design.

6.1.1. The log-domain LPF neuron

The log-domain low-pass filter neuron (LLN) [113] implements many behaviors of the generalized Integrate and Fire (IF) model such as spike bursting and spike-frequency adaptation. The LLN neuron is constructed from a few transistors, operating with low power (50–100 nW), and with easily configurable time constants and conductance by controlling gate voltages. It consists of four stages described by four sub-circuits. Fig. 7 shows a schematic of the LLN neuron. The first stage (yellow, M_{L1-L3}) is responsible for temporal integration (i.e, synaptic response) using a sub-threshold log-domain low-pass filter circuit. The second stage (red, M_{A1-A6}) is responsible for generating an output spike signal. The last two stages are responsible for generating reset and refractory signals (blue, $M_{R1,R3}$) and for LPF spike frequency adaptation (green, M_{G1-G4}).

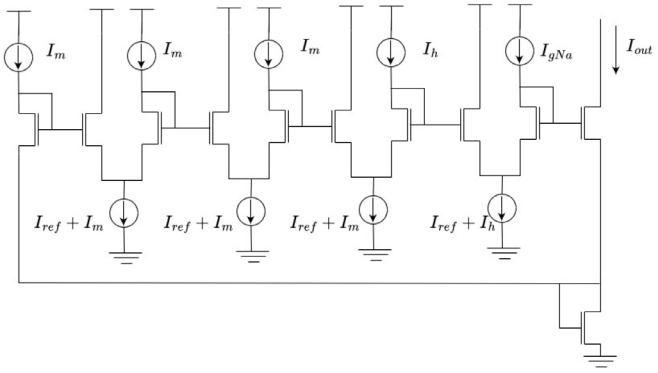


Fig. 8. Translinear circuit implementing a gated conductance of the form x^3yg used to implement H–H conductance equations.

6.1.2. A sub-threshold Hodgkin–Huxley based neuron

In [114], the authors implemented an analog model of the H&H neuron (Section 3) using sub-threshold CMOS circuits. Input spikes are temporally integrated using a log-domain low-pass filter called "Tau-cell" which was first proposed in [115]. The post-synaptic current is passed to a subthreshold circuit as shown in Fig. 8 that implements the nonlinear functions typically used with the gating variables of the H–H model *m, h and n*. In this work, the authors implemented a mixed-signal VLSI chip integrating a biophysical network of four H–H neurons and twelve conductance-based synapses. All gating variables were programmable using digital signals and Digital-to-Analog Converters (DAC).

6.2. Digital implementation

Digital implementations of neurons simulate the neural behavior in discrete time using discrete approximate solutions of the ordinary

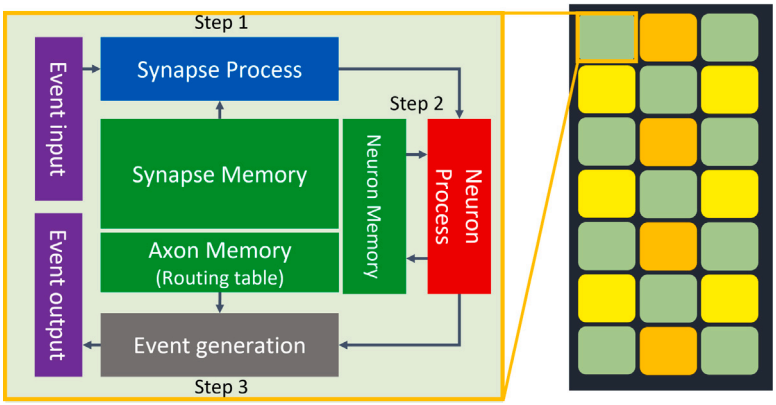


Fig. 9. Left: General overview of an event-driven digital neuro-synaptic core optimized to process spiking neural networks. Right: Scaling up and building a heterogeneous neuromorphic processor by connecting various neuro-synaptic cores with a network-on-chip.

differential equations describing the neural model. Digital implementations usually share compute resources with time-multiplexing.

Digital neurons do not suffer from the design variation challenges of analog neurons, at the expense of more area and power than their analog counterparts. However, they can still benefit from low-precision computation and from the sparsity of spiking neural networks. In addition, time-multiplexing allows the reusing of computational blocks of neurons and synapses, making full digital neuromorphic processors more area-efficient than analog processors. The area efficiency of time-multiplexing is more significant when a complex neuron model is implemented and amortized over many neurons.

Digital neurons can come with different degrees of flexibility depending on their compute units. Some architectures implement only a specific set of neurons with optimized and efficient datapath and compute units [14,116], other architectures implement a more flexible datapath that can implement a family of linear integrate and fire neurons [69], by leveraging fully programmable cores that can be programmed to implement any kind of neuron model [68,117]. In addition, using digital technology provides the flexibility to easily port the design to newer and more advanced technology nodes and therefore benefits more from technology scaling.

Fig. 9 shows three primary steps of processing events in a digital neuromorphic processor. When an event enters the core through a virtual axon, it will broadcast to many neurons, resulting in several post-synaptic neural updates and possible spike firings. The synaptic process in Fig. 9 is the step to apply optional delay on the synaptic events (for example as in [11]), read the relevant synaptic weights, calculate the affected neuron addresses (for example to support convolutional connectivities [69,118]) and updating the synaptic weights (if on-device learning is supported, for example in [69]). The synaptic process may receive feedback from other steps to perform on-device learning. This process can also support interesting features like sparse weight compression [118] or event-decompression [117], which helps to improve synaptic memory usage and event communication efficiency.

The next step is to update the states of the relevant neurons. This process is the core of each neuromorphic processor and defines the type of neurons, data types, timing and mappings. Since each input spike usually requires updating a large population of neurons, the neuron process also has a significant role in the processor's efficiency. Some processors support several neuron types [118] or programmable neurons' equations [119]. The neuron process requires memory for neuron states and other neural parameters (like threshold, refractory times, etc.). Neuron computation can be fully parallel and process the incoming event for all involved neurons in one step [17], fully time-multiplexed (one step for each neuron update) [11,69,118] or partially parallelized [117]. Furthermore, calculating the decay of neuron state can be efficiently approximated [120].

For low-power digital neuromorphic computing, non-linear I&F models pose huge challenges. While linear I&F digital implementations require a response only at input spike time, non-linear I&F models (i.e multi-compartment neurons) require book-keeping (i.e, memory) and computation over multiple discrete timesteps after the input spike time, making them inefficient for low-power event-driven computation [121], although phenomenological approximations are possible [122].

As a result of the neural updates, some neurons may fire spikes which will be passed to the following processing step. In the final step, the event generation process makes the spike packets using the address of the firing neurons and the information in the axon memory. Axon memory includes information related to the destinations of the spikes packet. The spikes in the multi-core systems will travel through the Network-on-Chip to the other neuro-synaptic cores (virtual axons). For smaller scale processors where the layers of neural networks are hardwired to each other [17], axon memory is not required. The event-generator may compress the events, have a feedback path to the synapse process (for recurrent layers and on-device learning), apply a delay to output events to implement skip connection or implement some of the linear layers like average pooling.

6.3. Emerging NVM devices

Emerging Non-Volatile Memory (NVM) devices are attractive candidates for synaptic modeling: they can mimic both *memory* since they are storage elements, and *learning* since they can exhibit plasticity [123]. Furthermore, they have also been explored for neuronal modeling [124–126], and can potentially enable highly dense, distributed, scalable and energy and area efficient neuromorphic systems [66]. In this section, we list some of the most promising NVM candidates that have been employed for modeling synapses and neurons.

6.3.1. Phase-change memory

Phase-Change Memories (PCM) are a class of emerging non-volatile memory devices. They consist of a Chalcogenide structure sandwiched between two electrodes and connected with a resistive electrode (heating element), as shown in Fig. 10. These devices have programmable resistive properties where the resistance is a function of their atomic structure. In a binary PCM cell, the amorphous state has a High-Resistance State (HRS) and the cell is in the RESET operation, while the crystalline state has a Low-Resistance State (LRS) and the cell is in the SET operation. However, these states can be a continuum to represent multiple states, which is the case in multi-level devices. PCMs require heat to transition between amorphous and crystalline states as well as for readout. The PCM crystallizes by applying medium voltage electric pulses. To switch the device back to amorphous state, it requires a

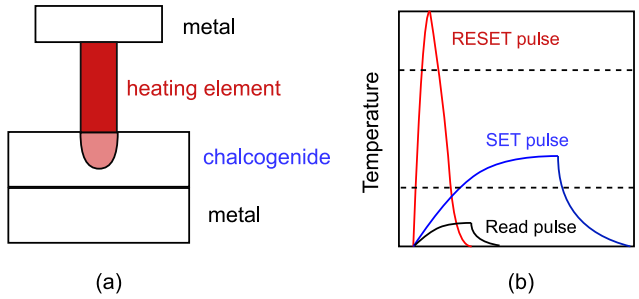


Fig. 10. (a) A phase-change memory and (b) the temperature needed for SET, RESET and READ operations.

strong short voltage pulse. This causes an abrupt melting and quenching of the device followed by rapid cooling [127].

Since the resistivity of crystalline of PCM is programmable, it can be exploited to use the PCM as a synapse element connecting pre- and post-synaptic neurons. Similarly, the amorphous (RESET) state can be used to model the neuron's resting state while the gradual transition from the amorphous to the crystalline state (SET) models the temporal integration of incident spikes. The crystalline state models the neuron's firing state. This creates a strong voltage pulse, which resets it to the amorphous state [128]. In [124], the authors present an I&F neuron with homeostatic regulation using a PCM device. They also demonstrate learning via STDP. In [129], a multi-memristive synaptic architecture is proposed, in which a synapse is implemented using multiple PCM devices. In this approach, a synaptic weight is represented by the combined conductance of the PCM devices.

6.3.2. Resistive RAM

Since the birth of variable-resistance Anodic Oxide materials, the application of Resistive Random Access Memory (RRAM) has been extensively studied [130]. Like PCMs, the operation of RRAM is governed by two conductivity states: LRS and HRS. It has a sandwiched metal–insulator–metal (MIM) structure where the insulator can switch between being an oxide (insulator) and an electrolyte (conductor). Like PCM the states of RRAM can be a continuum to represent multiple states, which is the case in multistate/multi-level devices. An RRAM cell is shown in Fig. 11, where an insulating film (HfO_2) is sandwiched between two conducting electrodes in a MIM structure. The resistive switching process of RRAM devices typically involves either the construction/disruption of conductive filaments, or the modulation of the carrier transport barrier at the electrode/switching layer interface induced by ion migration. These two switching mechanisms are called *filamentary* and *interfacial* switching, respectively. In filamentary switching devices [131,132], a one-time application of a strong electric field upon device operation is required for the initial formation of the conducting filament, which is also known as the electroforming process. A compliance current is necessary to confine the current flows through the local path in LRS. For this reason, there is a substantial interest in the usage of interfacial switching devices for avoiding the electroforming step altogether [133]. Their further advantage is their self-rectifying behavior, which is key in developing selector-free memristive crossbar arrays (MCAs).

The current characteristics (I–V) curve of an RRAM device shows a jump in current at the dielectric breakdown. This jump in current is used to model the output action potential (spike). The variable resistance states of RRAM can mimic synapse behavior. Roy et al. [134] successfully showcased the use of RRAM for modeling synapses. They demonstrated potentiation, depression and STDP using optimized Al-HfO_2 RRAM devices. Similarly, the transition from oxide HRS state to electrolyte LRS state is used to model the temporal integration of spikes. Hence, RRAMs can be used to create I&F neuron models. In [125], authors introduced an all memristor-based stochastic SNN architecture that implemented stochastic I&F neurons using RRAM. They demonstrated 6.4× less energy consumption compared to its CMOS-based counterpart with only 1% loss in accuracy. They use RRAM to implement stochastic I&F neurons.

6.3.3. Magnetoresistive RAM

Magnetoresistive RAM (MRAM) devices are memristors that offer almost unlimited endurance and a fast switching mechanism compared to RRAM and PCM devices, which suffer from degradation problems [135]. Their unlimited endurance is enabled by their switching mechanism, which depends on the spin dynamics and magnetic properties of electrons. These properties do not involve any movement of subatomic particles, which can cause wear-out. Such memristive properties that are a function of the device's magnetic state are classified under magnetoresistance. As magnetic properties are independent of the power supply, the magnetoresistance state can act as an NVM. Magnetic Tunnel Junction (MTJ) was the first device to use electron spin properties for storage [136]. It consists of two ferromagnetic nanomagnets with a dielectric layer in the middle. While one nanomagnet has a fixed magnetic polarization, the other is free. The direction of an applied current can switch the magnetic polarization of the free nanomagnet from parallel to anti-parallel. This creates the ON/SET and OFF/RESET states of the device, which are used to model an I&F's neuron's firing and resting states. Fig. 12 illustrates the switching activity of MTJ under the application of an electric current. Other MRAM devices follow similar principles as MTJ. One of the challenges of employing MRAMs commercially is their low ON/OFF resistance ratio which requires sensitive resistance sensors compared to PCM and RRAM [137]. Similar to PCM and RRAM, MRAM devices can be used to implement synapses and synaptic neurons by controlling the magnetic orientation of the ferromagnetic layers.

Sharad et al. [138] implemented a spin-based integrate-and-fire neuron model using MTJ devices. They also implemented synapses using domain-wall magnets, which also fall under MRAMs. In this work, again, the device's switching activity is used to model the integrate and fire behaviors.

6.3.4. Graphene-based NVM devices

Graphene is one of the promising post-silicon candidates due to its various properties such as flexibility, thinness and ballistic transport. Most importantly, graphene is a bio-compatible material, which makes it favorable for different neuromorphic bio-interfaces when compared to other emerging technologies [139–141]. A basic graphene-based NVM cell is shown in Fig. 13. It consists of a single layer Graphene Nanoribbon (GNR) located above an insulating material and a doped substrate (back gate). When a bias voltage is applied between the source and the drain terminals ($V_d - V_s$) the GNR works as a conduction channel. This conductance can be modulated by changing the graphene sheet geometry, the contacts topology, or by means of external voltages via the top/back gates (i.e., V_g and V_{bg}, respectively) [126].

In [139], graphene-based artificial synapses were implemented that exhibited *tunable* STDP and long-term plasticity. In [141], graphene-based *multi-level* (i.e., >16) memristive synapses are proposed that allow precise (i.e., arbitrarily programmable) weight updates. In [126], an all-graphene-based nonlinear leaky integrate-and-fire neuron was proposed that is made up of 6 GNR units. This neuron showed variability resilience and output-firing regularity for a varying input firing rate from 20 to 200 spikes per second.

6.3.5. Ferroelectric RAM

Ferroelectric Random Access Memory (FeRAM) has attracted increasing interest in the field of ferroelectricity and nonvolatile memories [142]. The key features of FeRAM are (1) nonvolatile data storage capability, (2) lowest power consumption among various semiconductor memories, and (3) faster operation speed, i.e., as fast as that

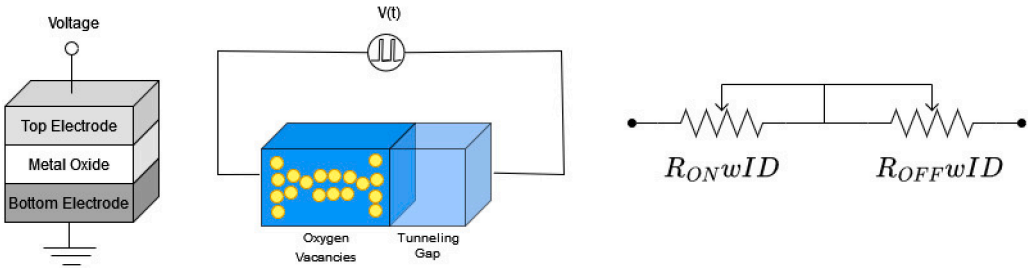


Fig. 11. Basic diagram of an RRAM device with a simplified equivalent circuit.

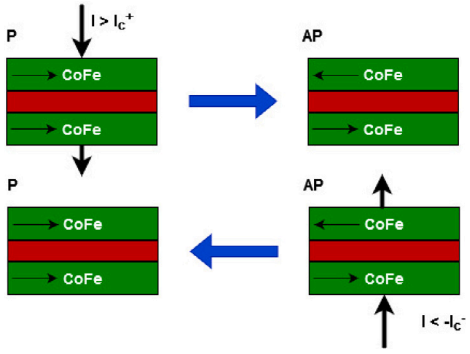


Fig. 12. The spin-MTJ state changes from parallel (P) to anti-parallel (AP) if a positive current density $I > I_{c+}$ is applied. If a negative direction current density $I > I_{c-}$ is applied, its state will return to anti-parallel.

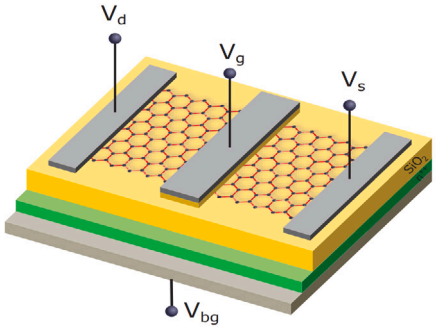


Fig. 13. A basic graphene-based NVM cell.
Source: Reproduced from [126].

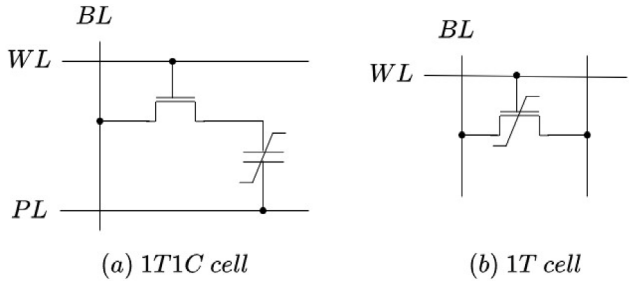


Fig. 14. Bit-cell structure of FeRAM devices (a) 1T1C-type FeRAM (b) 1T-type FeRAM.

of DRAMs (dynamic RAMs) [143]. Depending on their data storage and readout mechanism, FeRAMs are classified into two categories: capacitor-type and FET-type FeRAMs [144]. Capacitor-type FeRAMs use ferroelectric material to store the data with a different polarization state. The polarization state of ferroelectric materials can be tuned under external voltage. A typical cell structure in the capacitor-type FeRAM is a 1T1C-type cell shown in Fig. 14(a), while a typical cell structure in the FET-type FeRAM is a 1T-type cell shown in Fig. 14(b). The 1T1C FeRAM cell, which uses 1 transistor and 1 capacitor with three control signals: Bit line (BL), word line (WL) and Plate Line (PL), is the most common structure, it is similar to a DRAM cell. To write data (e.g., '1') into a 1T1C cell, the BL voltage is raised to V_{dd} so that the negative voltage bias is applied to the FeRAM before PL voltage rises. When the PL voltage rises to V_{dd} the polarization state is switched back to negative remnant polarization (−Pr), that represents data '1'. To write '0', the BL is grounded when the PL voltage is raised to V_{dd} and the polarization state is switched to positive remnant polarization (+Pr), which represents data '0' [145]. The state of the FeRAM cell is read by first pre-charging the BL to zero voltage and applying a V_{dd} pulse to PL while the BL remains floating.

The non-volatility, low-power and CMOS compatibility features of FeRAM can be exploited to utilize FeRAM as the basic cell element of CIM architectures [143]. FeRAM devices can be utilized as the bitcells of a generic crossbar array for CIM operation, where they can be used to perform the weighted sum operation in a similar fashion like RRAM, or PCM-based CIM crossbar arrays. Example works include [123,142], which have implemented synapses for CIM using FeRAMs.

6.4. Conclusion

Hardware realizations of spiking neurons and synapses span analog CMOS, digital, and emerging NVM substrates, each trading fidelity, programmability, density, and energy. Analog offers real-time, ultra-low-power dynamics but variability; digital provides flexibility and scalability at higher overhead; NVM enables non-volatile, in-memory computation for dense, energy-efficient neuromorphic systems.

7. Hardware architectures for spiking neural networks

To highlight the key tradeoffs at the microarchitecture, chip design, and system levels, we will go through the main design choices for neuromorphic hardware by first covering qualitatively the main trends and options, and then placing them in the current neuromorphic chip landscape. Importantly, most of the design choices analyzed hereafter are directly related to the fundamental aspects of neuromorphic computing covered in Section 4. Yet, hardware efficiency considerations can degrade the granularity at which these fundamental aspects can actually be realized in silicon hardware. The purpose of this section is to highlight these challenges, as well as the diversity of approaches that are being pursued.

7.1. Application-specific or general-purpose?

Whether to build application-specific or general-purpose hardware is among the most pressing questions in the neuromorphic field, where concrete application demonstrations are still at an early stage (see

Section 9.2). Two general trends can be found in the design of neuromorphic systems. On the one hand, the historical approach, since the late 1980s, aimed to replicate biology *in silicon* in a bottom-up fashion that is driven by neuroscience insights, leading to general-purpose systems [1]. However, while such neuromorphic systems shine as experimentation platforms and test-beds for new algorithmic developments, their emphasis on flexibility and the lack of insight on how to best exploit biological primitives restricts their efficiency in real-world use cases.

On the other hand, a more recent trend focuses on the target use case and on demonstrating that application-specific neuromorphic systems can outperform conventional approaches. This top-down approach requires carefully selecting which specific insights from biology are deemed as the most critical ones toward achieving an efficiency advantage [146].

Chip highlights:

- The most famous large-scale neuromorphic platforms are all general-purpose systems: The Neurogrid [10] aims at efficiently exploring the deployment of 1M-neuron cortical microcircuit architectures; BrainScaleS 1 and 2 [112,147] aim at accelerating neuroscience simulations up to billion-neuron supercomputer-scale setups, including bio-physically realistic neuron models and plasticity mechanisms; SpiNNaker 1 and 2 [148,149] are based on multi-core digital architectures and focus on maximizing flexibility with an almost arbitrary support of neuron and synapse models at the expense of efficiency, while also allowing for billion-neuron supercomputer-scale setups; TrueNorth [150] is the largest single-chip system with 1M neurons and 256M synapses and efficiently supports large-scale functional abstractions of neuroscience with a restricted behavior repertoire; Tianjic [151] focuses on exploring hybrid ANN-SNN setups; Loihi [69] aims at a good balance between flexibility and efficiency through a carefully selected neuron behavior repertoire, a programmable plasticity co-processor, and a reconfigurable synaptic fan-in. These large-scale general-purpose systems are complemented by smaller-scale <10k-neuron chips that serve as test-beds for the deployment of edge-computing applications, e.g. ROLLS [76], DYNAPs [16], ODIN [14], MorphIC [116], and μBrain [17], the latter offering reconfigurability at synthesis time.
- Application-specific neuromorphic systems typically have fixed network architecture and neuron/synapse models, thereby trading off flexibility for a higher efficiency in their target use case: a few designs focus on image denoising and sparse coding [152,153], SPOON [154] focuses on adaptive edge detection with event-based cameras, the design from Wang et al. [155] focuses on ultra-low-power always-on keyword spotting, and ReckOn [156] focuses on end-to-end user adaptation over second-long timescales with any event-based sensor.

7.2. Analog or digital?

As explained in Section 4.9, the brain intrinsically is a mixed-signal system. Replicating biology is typically done either by following an *emulation* approach (i.e. leveraging the physics of the silicon substrate and of emerging devices to reproduce the brain dynamics) or a *simulation* approach (i.e. the substrate dynamics are ignored and behavior is replicated at the functional or model level). Sub-threshold analog design is the best suited approach for a biological-time emulation of biological processes. Indeed, the charge carrier flow in the MOS transistor channel in the sub-threshold regime is governed by a diffusion mechanism, as are the brain's ion channels [158]. On the other hand, for analog above-threshold designs, the current flow is governed by a drift mechanism instead of diffusion. These designs thus cannot pursue an emulation approach at the level of the device physics, but can still emulate many neuron and synapse models [159]. Furthermore, as currents are increased by 3–4 orders of magnitude in above-threshold designs, time constants accelerated by 3–4 orders of magnitude compared to biological time can be realized [160]. In terms of memory storage, the area cost of capacitor-based storage is typically considered too high [76], unless a biological-time fully-parallel implementation is aimed for (see Sections 7.3 and 7.4 below). The default option for analog, mixed-signal, and digital designs is thus to resort to standard digital static random-access memories (SRAMs), while the best tradeoffs might be uncovered by exploiting emerging memristive devices (see Section 6.3). Fig. 15 shows a multi-neuron crossbar analog architecture in which neurons and synapses are organized in a neurosynaptic core. Many neurosynaptic cores can be stacked through a digital communication channel.

Despite allowing for an intrinsically efficient emulation approach, analog designs do not fully leverage technology scaling and suffer from device mismatch, noise, and susceptibility to power, voltage and temperature variations. As of today, it is still unclear whether such variations are a bug, as in conventional integrated circuit design, or a feature, as in the brain [146]. Digital designs alleviate these issues at the expense of forgoing the emulation approach for simulation-based one. This design choice tends to degrade the overall efficiency by increasing data movement. However, this decrease is usually compensated for by technology scaling. Indeed, digital SRAM memories exhibit an excellent tradeoff between density, speed, and energy efficiency in advanced nodes [161].

Chip highlights:

- The historic approach to designing neuromorphic chips focused on exploiting the MOS transistor physics to emulate the brain biophysics. This approach is still pursued today, and the most representative designs include ROLLS [76], DYNAPs [16], Neurogrid [10], and Braindrop [162]. These designs typically exhibit record efficiency in biological time with down to 10–100 fJ/SOP at the neuron/synapse levels and follow an *understanding by building* approach that aims to reverse-engineer the brain by designing silicon devices based on its operating principles, although it is still unclear how to best exploit their inherent variability. In terms of synaptic storage, ROLLS followed a fully-parallel capacitor-based implementation at the expense of synapses taking up more than two thirds of the chip area, an issue that was solved in DYNAPs by merging synapses within a ternary content-addressable memory (TCAM)-based routing infrastructure. Neurogrid stores all synaptic weights off-chip, while Braindrop adopts an SRAM-based storage.
- The above-threshold analog design approach is unique to the BrainScaleS 1 and 2 designs [112,147], both of which follow a model-based emulation approach running with acceleration factors of 3–4 orders of magnitude compared to biological time. Synapse storage relies on digital SRAM. To ease programming and control, the BrainScaleS systems support advanced variability compensation techniques [159], although in-the-loop training with surrogate gradients has the potential to reduce the need for such techniques [163].
- Digital designs mainly follow three different types of strategy. First, globally asynchronous and locally synchronous (GALS) designs, such as TrueNorth [150] and SpiNNaker [148], employ synchronous clocked cores for controllability, robustness, and programmability, and an asynchronous routing fabric to optimize for fast low-distortion spike-packet-based communication. Second, fully asynchronous designs, such as Loihi [69], μBrain [17], and the chip from Wang et al. [155] allow for fast and fully event-driven processing at the expense of requiring advanced design techniques to ensure a robust clock-free implementation. Finally, fully synchronous designs, mainly consisting of small-scale designs such as [14,116,164], are best supported by commercial

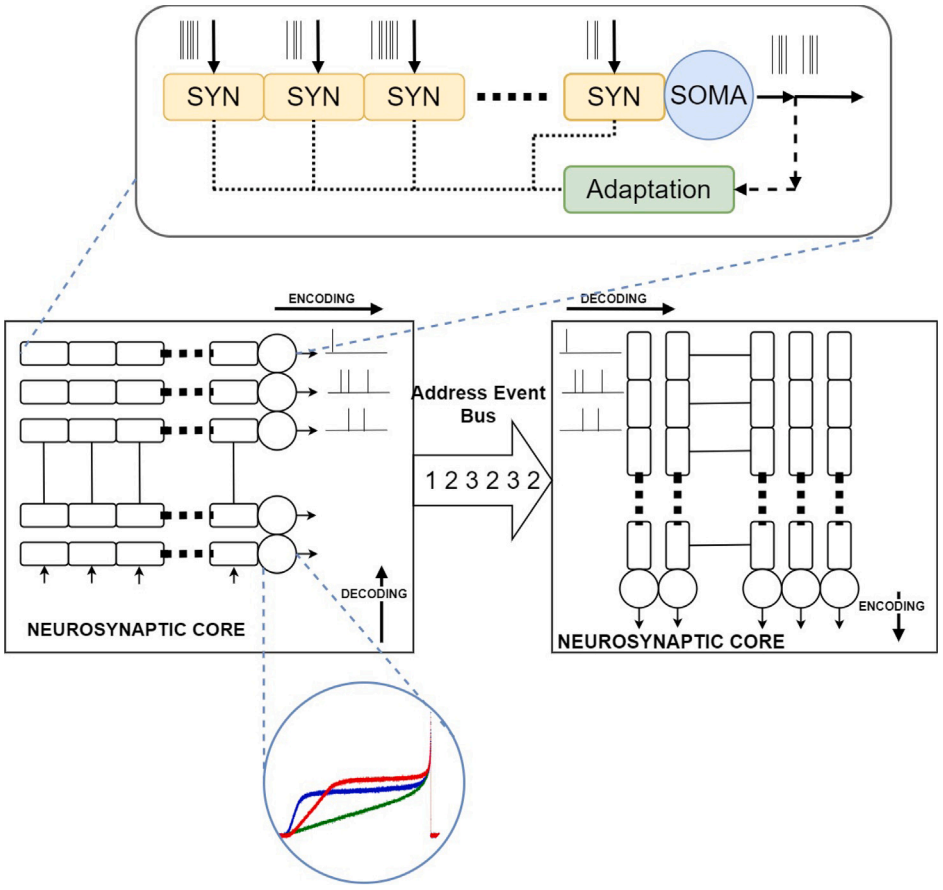


Fig. 15. General overview of a neuromorphic mixed-signal architecture. The processor comprises synapses and a neuron block organized in a neuro-synaptic core. The neurosynaptic core receives input spikes and emits output spikes; the internal computation is carried out in the analog domain. The blue circle shows the integration of current by an analog neuron with different bias settings. On the other hand, communication is carried in the digital domain and by means of the Address Event Representation (AER) protocol. This template is common in mixed-signal neuromorphic processors as [76,157] and implies fully parallel and physical implementations of neurons and synapses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

electronic design automation (EDA) tools, can easily provide full hardware–software equivalence, and techniques such as clock gating can partly mitigate the clock tree penalty for event-driven spike-based processing.

7.3. Biological-time or accelerated?

Whether the design should be optimized for biological- or accelerated-time operation mainly depends on the target use case. Application-specific designs targeting a low-power always-on real-time deployment should rely on biological-time execution, just as biological systems are matched to the time constants of their environment. Similarly, designs aiming at a close emulation of biological systems will also aim for biological-time processing. In contrast, accelerated-time processing is typically chosen for fast offline processing of previously collected data [165], for accelerating hardware-variability-aware training before deployment [166], or for accelerating processes taking place over hour-, day-, or year-long timescales, from homeostatic to evolutionary processes [160].

In terms of chip design, the choice between biological- or accelerated-time operation mainly translates to throughput requirements. There are typically two processing flows in neuromorphic systems: (i) time-stepped operation at a fixed temporal resolution (i.e., time step), or (ii) online continuous-time execution. In the case of time-stepped operation, the design target will be a bound on execution time, as achieving the target acceleration factor requires a guarantee that all computations for worst-case network activity will be carried out within the time step. For online continuous-time execution, as spike events awaiting processing are buffered, the design target will be a specification on the maximum amount of spike-timing distortion. As biological-time operation only implies slow processing speeds when it comes to modern silicon hardware (i.e. each neuron spikes at most at a few hundreds of Hertz), these throughput requirements only require careful optimization when considering accelerated-time operation.

Chip highlights:

- Sub-threshold analog designs such as ROLLS, DYNAPs, Neurogrid and Braindrop typically run in biological time [10,16,76,162], thereby targeting applications such as real-time biosignal [167, 168] and speech processing [169]. In contrast, above-threshold analog systems such as BrainScaleS [112,147] exhibit time constants down to the microsecond, which makes them unfit for real-time sensory processing, where time constants are typically on the order of the millisecond. They are thus typically used for neuroscience simulation acceleration [170] or for accelerated processing of existing benchmarks [166]. In both the sub- and above-threshold cases, analog systems typically follow an online continuous-time execution scheme without global time-stepped synchronization. High-speed asynchronous routing links are thus employed to minimize distortion [16], see Section 4.4.
- For digital systems, the time constant is usually configurable from biological- to accelerated-time, either by adjusting the timestep duration for globally synchronized systems such as TrueNorth

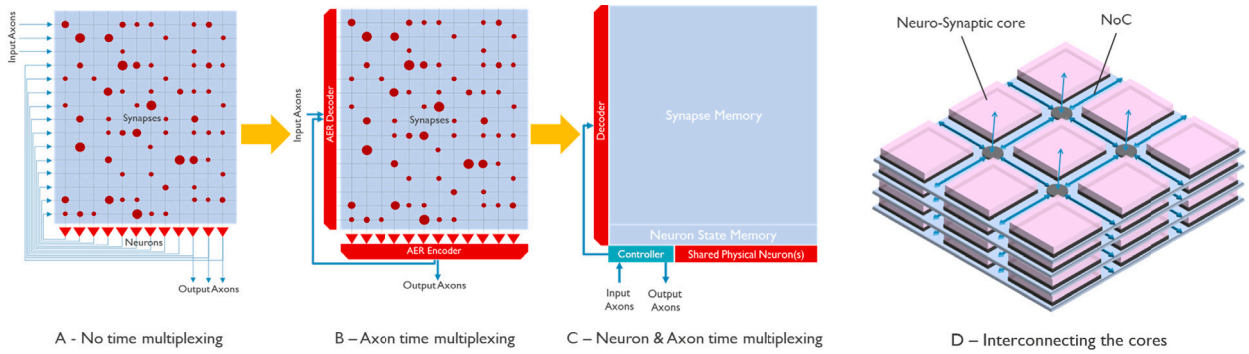


Fig. 16. Overview of main levels of time multiplexing. (A) Fully-parallel implementation of neurons, synapses, and axons. (B) Time-multiplexed axons, fully-parallel neurons and synapses. (C) Time-multiplexed neurons, synapses, and axons. (D) Multi-core overview with time-multiplexed axons.

and Loihi [69,150], or by adjusting the time constants of neuron/synapse models and of the input event stream for online real-time systems [14,156]. This flexibility makes digital systems easy to deploy for a broad range of both online and offline use cases [156,165,171].

7.4. Parallel or time-multiplexed?

Time multiplexing relies on sharing neuron/synapse/dendrite/axon circuitry and accessing their state data in a centralized memory (Fig. 16). Importantly, this technique can be applied both in the digital and the analog domains, e.g. digital logic and SRAM-based storage for the former, and analog circuitry and switched-capacitor or SRAM-based storage for the latter.

Fig. 16(a) outlines a fully-parallel implementation where all neurons, synapses, dendrites, and axons are implemented as independent instances, similarly to the brain. The absence of resource sharing allows for the highest throughput and lowest dynamic power by minimizing data movement. However, this architecture typically has a higher static power penalty and is difficult to scale due to (i) the footprint of all elements, especially for versatile neuron and synapse models, and (ii) wiring constraints between individual elements.

In Fig. 16(b), only the axons are time-multiplexed. This architecture is the most common one in mixed-signal designs aiming for fully-parallel neuron/synapse resources while saving on wiring costs with a digital address-event representation (AER) bus or a dedicated network-on-chip (NoC), where each spike is encoded in a data packet including the address of the source/destination neuron(s) and, optionally, a timestamp [172]. The spike encoding circuit overhead is usually considered negligible, while a high-speed shared digital bus can safely accommodate for tens of thousands of neurons spiking at biological time constants or at low acceleration factors. Nearly all neuromorphic architectures apply axon time multiplexing, which is a necessary condition to scale and easily interface multiple cores, as shown in Fig. 16(d).

A further step consists, in addition to the axons, of time-multiplexing the physical neurons, as shown in Fig. 16(c). In this case, one physical neuron can emulate hundreds of neurons whose states are stored in a centralized memory. This also usually implies some level of time multiplexing in the synapses, e.g. synaptic data is stored in a centralized memory but the afferent synaptic weights of a given neuron can be retrieved in parallel. Time multiplexing neurons and synapses allows drastically cutting their area footprint and is thus a key enabler for large-scale designs that is applied in most digital systems and a few mixed-signal ones, as outlined below in the chip highlights. This comes at the expense of an increase in control complexity and dynamic power due to a neuron/synapse state data movement penalty. Throughput requirements should also be considered, as the higher the number of neurons/synapses emulated by a single physical instance, the more difficult it will be to carry out accelerated- or even biological-time processing.

As outlined above, time multiplexing design choices cannot be considered independently from locality and memory architecture optimization. Centralized memories exhibit the best density, but increase the data movement cost, and thus impact the overall energy efficiency of the design [173]. A careful hierarchical memory architecture design is thus required.

Chip highlights:

- Fully-parallel neuromorphic designs are uncommon, as all chips embed at least some level of axon multiplexing through an AER interface. Some notable designs exploiting a fully-parallel implementation of neurons and synapses include ROLLS [76] and μBrain [17], yet following very different rationales. The former is mixed-signal and targets a continuous-time emulation of all neuron and synapse dynamics, the latter is implemented with asynchronous digital logic and targets a fully event-driven clock-free design allowing for both high-throughput and low-dynamic-power operation. It is worth noting that μBrain comes close to the fully-parallel implementation outlined in Fig. 16(a) as its AER bus is implemented only for inter chip-level interfacing, and axons are not time-multiplexed internally.
- With the exception of μBrain, nearly all digital designs time multiplex neuron and synapse resources [14,69,116,148,150,154,156,164]. However, a few mixed-signal designs also go for this approach, such as [157,174]. Interestingly, [157] time multiplexes synapses while keeping fully-parallel neuron resources, thereby drastically cutting the synaptic array area footprint while maintaining throughput and continuous-time neuronal dynamics.

7.5. Centralized or distributed?

The crossbar core with all-to-all neuron connectivity supports arbitrary neural network topologies. However, maintaining a high resource utilization becomes challenging as the number of neurons per core increases. Scaling up neuromorphic chips thus requires breaking them down into many cores, thereby losing all-to-all connectivity at the system level. Finding the right balance between the core architecture, the number of neurons per core, and the number of cores per chip strikes a tradeoff between hardware utilization, synaptic fan-in, fan-out, and flexibility for an efficient support of deep neural network topologies.

Chip highlights:

- Single-core crossbar designs such as ODIN and ROLLS [14,76] are typically limited in the diversity and complexity of tasks that they can support, and rather serve as a proof-of-concept for the tradeoffs that can be obtained in multi-core systems (see hereafter). In contrast, a few single-core designs adopt a fixed

multi-layer network topology [154–156,175], thereby leading to excellent hardware utilization at the expense of flexibility.

- The tradeoffs resulting from scaling-up large crossbar cores can be illustrated with MorphIC [116], which embeds four 512-neuron crossbars. Such multi-core crossbar architectures allow for large bio-plausible fan-in and fan-out values of 1k and 2k, respectively. However, only fully-connected or recurrent layers allow leveraging these large fan-in and fan-out values with high hardware utilization. Convolutional layers, for example, require inefficiently replicating small kernels over the large synaptic resources of output feature map neurons, leading to poor hardware utilization.
- Allowing for a flexible and efficient deployment of different network topologies thus requires going beyond the crossbar architecture at the core level. For example, Loihi [69] allows reconfiguring the synaptic fan-in by storing connectivity and synaptic weight information in an SRAM memory, while DYNAPs [16] adopts a flexible hierarchical tag-based connectivity scheme, at the expense of a low synaptic fan-in of 64.

7.6. Sparse or dense?

A hallmark of the brain is to exploit sparsity, both in space and time (see Section 4.5). In the space dimension, the brain relies on an approximate small-world connectivity scheme where neurons are densely connected locally and only have sparse long-range connections [176]. In the temporal dimension, the brain relies on a mix of spike codes optimizing between robustness and encoding efficiency [177]. Sparser spike codes, such as time-to-first-spike (TTFS) encoding, exhibit a higher number of information bits per spike but entail more complex operations and are more sensitive to spike loss.

In contrast to the spatial and temporal sparsity of the brain, most of the current neuromorphic systems actually rely on a crossbar core architecture with all-to-all neuron connectivity and on a spike-rate code, which allows for a straightforward ANN to SNN mapping at the expense of failing to exploit sparsity, in both space and time. While multi-core architectures with crossbar cores approach a small-world connectivity scheme, exploiting temporal sparsity appears to be a much bigger challenge that requires co-optimizing the sensor, the neuron model, the spike routing fabric, and the SNN processing algorithm.

Chip highlights:

- Regarding *spatial sparsity*, both MorphIC [116] and DYNAPs [16] are based on a small-world-like hierarchical connectivity scheme, where intra-core connectivity is dense and inter-core/chip connectivity is sparse. The neuromorphic chips in [152,178] propose an alternative architecture based on locally-competitive algorithms (LCAs), where competition is introduced for sparse feature extraction. In both cases, the core architecture still strongly restricts the connectivity schemes that are supported (see also Section 7.5), and further research is required to efficiently support connectivity schemes that are both sparse and flexible.
- Regarding *temporal sparsity*, going beyond rate-code-based ANN-to-SNN demonstrations, which nearly all current neuromorphic chips support, requires an extended range of neuron/synapse behaviors and time constants, as well as spike-time-aware training algorithms. A few works in this area include SPOON [154], which is based on TTFS encodings without requiring pre-processing of neuromorphic retina event streams, and ReckOn [156], which supports spike-code-agnostic learning with the e-prop training algorithm [98] and enforces code sparsity with activity regularization. Several sparse coding experiments with Loihi are also reported in [171].

7.7. Static or plastic?

The performance of inference-only neuromorphic devices with static synaptic weights meets design specifications only if the input data is in accordance with the distribution that was used during the off-chip off-line training phase. This implies that synaptic plasticity is a key enabler for a robust long-term performance in uncontrolled environments, in which out-of-distribution data is frequently encountered and/or the data distribution shifts over time [179]. Interestingly, the brain excels at adapting to its environment in an online and continuous fashion based on a wide range of mechanisms [180], as outlined in Section 4.6. However, implementing plasticity in hardware, let alone the brain's full diversity of plasticity mechanisms, is a tough challenge as it usually entails not only complex circuitry to compute weight updates, but also dedicated memory architectures to handle specific memory access patterns. Carefully selecting the target plasticity algorithm, matching it with the target use case in a hardware-algorithm co-design fashion, and ensuring locality in both space and time, are thus necessary steps toward endowing neuromorphic devices with the ability to adapt to their environment [146].

Chip highlights:

- Given the challenging nature of implementing synaptic plasticity in hardware, a large variety of neuromorphic devices rely on static weights, and are thus only able to perform inference, such as Neurogrid [10], DYNAPs [16], TrueNorth [150], and μBrain [17]. Any change in the environment or task specifications will require a retraining and reprogramming these devices.
- Among the neuromorphic chips that implement synaptic plasticity mechanisms, the majority is based on bio-inspired STDP rules (Section 4.6). Key examples include the chip from Seo et al. [164], the chip from Chen et al. [153], BrainScaleS-1 [147], ROLLS [76], ODIN [14], and the chip from Mayr et al. [174]. Importantly, the three last chips are based on a learning rule that relies on the state of the post-synaptic neuron at the time of the pre-synaptic spike, which allows for a formulation that is *local in time*, as opposed to the conventional STDP rule that relies on a spike time difference. Finally, Loihi [69,119], SpiNNaker [68], and BrainScaleS-2 [112] support programmable forms of spike-timing-based learning rules, yet this flexibility comes at the expense of incurring a dedicated plasticity co-processor.
- It is notably difficult to reach decent accuracy levels with STDP rules directly grounded on neuroscience observations [146]: as these rules are typically unsupervised and based on local neuron activities, they need to be modified in order to minimize the network-level error. To address this issue, some recently proposed neuromorphic hardware designs implement variants of the error backpropagation algorithm that have been modified to increase biological plausibility. The chip from Park et al. [175] and SPOON [154] allow for efficient on-chip learning of non-temporal data (i.e. *static data*): they are based on feedback-alignment algorithms [181,182] that simplify the required memory architecture by removing the need to access both the weight matrices and their transposes. In order to learn *temporal data* instead, ReckOn [156] implements a forward-in-time approximation of the backpropagation through time (BPTT) algorithm [98] and enables on-chip training with temporal dependencies ranging from milliseconds to seconds.
- It is worth mentioning that, while all plasticity-enabled designs mentioned above can learn in an on-chip and online fashion, data efficiency and learning robustness remain open challenges. Early work in this direction includes the designs of [183,184], which are based on a bio-plausible implementation of *continual learning* and allow learning tasks sequentially while alleviating catastrophic forgetting, which is a dramatic loss of performance on previously learned tasks as new ones are learned with a different data distribution.

- Emerging nanoscale devices such as resistive RAM (RRAM) are increasingly considered for implementing synaptic dynamics due to their non-volatile, analog storage capabilities. At the same time, there is growing interest in hardware models that extend beyond point neurons to capture dendritic processing and local non-linearities, which are critical for biological computation [185].

7.8. Conclusion

Neuromorphic hardware spans a spectrum of design choices: general purpose vs. application specific, analog vs. digital, biological time vs. accelerated, parallel vs. time-multiplexed, centralized vs. distributed, sparse vs. dense, and static vs. plastic, each with distinct trade-offs in efficiency, flexibility, and scalability. The "best" architecture is inherently dependent on the use case and is based on careful hardware-algorithm co-design, with memory and communication costs often dominating system efficiency.

8. Mapping and compilers for neuromorphic systems

This section will present neuromorphic applications, compilation and mapping tools to targeted neuromorphic hardware. Executing a program on hardware involves three key steps: compilation, mapping, and run-time management. Although apparent for mainstream computers, these steps are challenging and not very well defined when executing an SNN-based machine learning application on a neuromorphic hardware device. Analogous to a mammalian brain, synapses of an SNN can be categorized into local and global synapses, based on the distance information (a spike) is conveyed. Local synapses are short distance links, where pre- and post-synaptic neurons are located in the same vicinity. Global synapses are those where pre-and post-synaptic neurons are further apart.

To reduce power consumption of the hardware implementation of SNNs, the following two principles are widely adopted in neuromorphic engineering.

- The number of point-to-point local synapses is limited to a reasonable dimension. One example is a $N \times N$ crossbar, which consists of N pre-synaptic neurons connected to N post-synaptic neurons using N^2 synapses.
- Instead of point-to-point global synapses (which are of long distance) as found in a mammalian brain, the hardware implementation usually consists of a time-multiplexed interconnect shared between global synapses.

Typically, the value of N is limited between 128 and 256; realistic SNN applications use amounts of neurons and synapses that are well beyond the capacity of a single crossbar. Therefore, an SNN model of an application must be partitioned into clusters, where each cluster is implemented on a crossbar: the partitioning step must thus take into account the resource constraint of a crossbar. Still, spikes communicated between the clusters (i.e., between the crossbars when these clusters are implemented on them) may create congestion on the global interconnect, which communicates spikes between crossbars. Spike congestion increases latency and energy and may also impact application performance. In the following, a general workflow is presented to map SNN applications to crossbar-based many-core neuromorphic hardware.

8.1. Partitioning large SNN models to clusters

Partitioning a large SNN into clusters is essentially a graph partitioning problem. Many efficient solutions have been proposed over the years to partition directed graphs.

Fig. 17a shows an example of an SNN application with 8 neurons and 13 synaptic connections. The number on each edge represents the total number of spikes communicated on the corresponding synapse for a given input. Later in this section, an overview is provided on how to extract spike information from a given machine learning model. Fig. 17b represents the partitioned SNN application with 4 clusters. The number on each edge represents the total number of spikes between the clusters. These are the spikes that are communicated between neurosynaptic cores when the clustered SNN application is mapped to the hardware. Since inter-core communication happens via a shared interconnect such as a Segmented Bus [186] or a Network-on-Chip (NoC) [187], the inter-core communication can constitute a significant fraction of the total energy consumption if there are too many spikes to communicate. Inter-cluster spikes also lead to an increased potential for network congestion, which may cause performance issues due to a change in the inter-spike interval (ISI), an encoding technique commonly used in SNN applications.

The performance of a machine-learning model can be expressed in terms of accuracy, Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) depending on the specific problem that is being addressed by the model. For SNNs, these performance metrics are defined in terms of ISI. If $\{t_1, t_2, \ldots, t_K\}$ denote a neuron's firing times in the time interval $[0, T]$, the average ISI of these spike trains is:

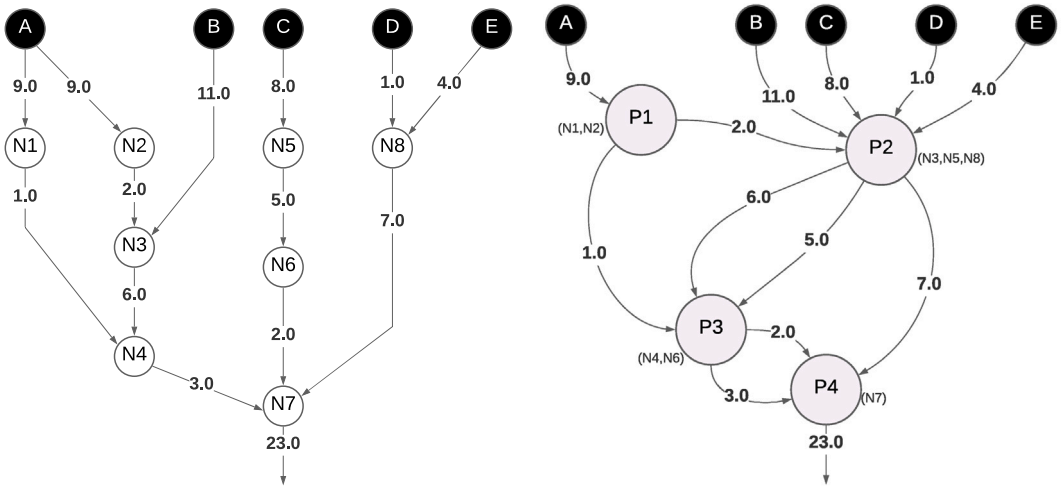
$$\mathcal{I} = \sum_{i=2}^{K} (t_i - t_{i-1})/(K-1). \tag{5}$$

To illustrate how a change in ISI, called *ISI distortion*, impacts model performance, we consider the example of a small SNN (Fig. 18a), where three input neurons are connected to an output neuron. Fig. 18b illustrates the impact of ISI distortion on the output spike. In the top sub-figure, a spike is generated at the output neuron at 22 μs due to spikes from the input neurons. In the bottom sub-figure, the second spike from input 3 is delayed, i.e., it has an ISI distortion. Due to this distortion, there is no output spike generated. Missing spikes can impact model performance.

Fig. 19 shows an example of the impact of ISI distortion on application performance, where the SNN in Fig. 18a is used for an image smoothing application. Fig. 19a shows the input image, which is fed to the SNN. Fig. 19b shows the output with ISI distortion due to variation in latency in the hardware. PSNR of this output with respect to the input is 19, which indicates a degradation in the image quality.

From this example, it is quite clear that spike latency plays an important role in determining the application performance. Fortunately, the latency can be controlled efficiently during SNN partitioning, i.e., by reducing the number of inter-cluster spikes. The problem draws parallel to finding a set of partitions of a weighted graph with the objective of minimizing a cost function. For the case of SNNs, weights are the number of spikes, while the cost function is the number of inter-cluster spikes. To this end, a fundamental graph partitioning algorithm is the Kernighan–Lin algorithm [188]. This is a heuristic partitioning solution which starts with some initial partition that satisfies the size constraints. It then repeatedly swaps nodes between the partitions to reduce the cost function. This is illustrated for an example SNN in Fig. 20, where after a certain number of steps, the number of inter-cluster spikes is reduced from 22 (initial allocation on the left) to 8 (in the right).

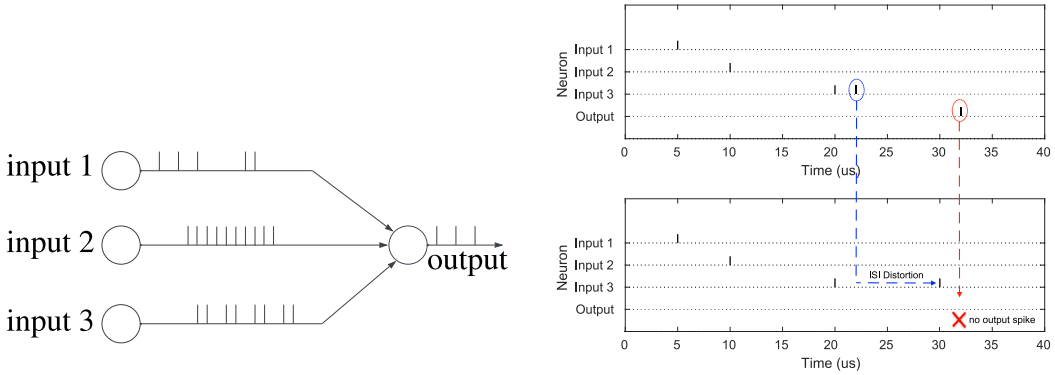
The Kernighan–Lin algorithm is used extensively for SNN partitioning [189,190]. To perform this SNN partitioning it is necessary to extract the spike information, which will be used as weights for edges (i.e., synapses) of the graph representation of an SNN. This process of spike extraction is also called workload generation. The general workflow to generate an SNN workload is to simulate the SNN using an application-level simulator such CARLSim [191] or Brian [192] with representative training examples. The workload consists of the following information.



(a) An SNN with 8 neurons and 13 synaptic connections.

(b) The SNN partitioned into 4 clusters.

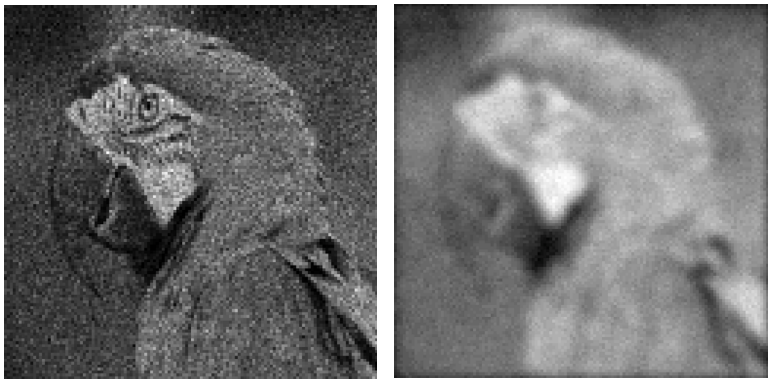
Fig. 17. Illustration of partitioning an SNN with 8 neurons and 13 synapses (left) to 4 clusters (right).



(a) An SNN with 3 neurons connected to an output neuron via their synaptic connections.

(b) No output generation due to increase in spike latency.

Fig. 18. A small SNN with three input and one output neurons (left). Impact of ISI distortion on the output spike.



(a) Original image sent to the image smoothing application.

(b) Output image generated from the image smoothing application. The SNN model has ISI distortion.

Fig. 19. Impact of ISI distortion on image smoothing application.

- *Spike Data:* the exact spike times of all neurons in the SNN model. We let $spk(i)$ represents a list of spike times of the ith neuron in the model.
- *Model Parameters:* the synaptic strength of all synapses in the SNN model. We let $p(i, j)$ represents the synaptic strength of the connection between the ith and jth neurons in the SNN model.

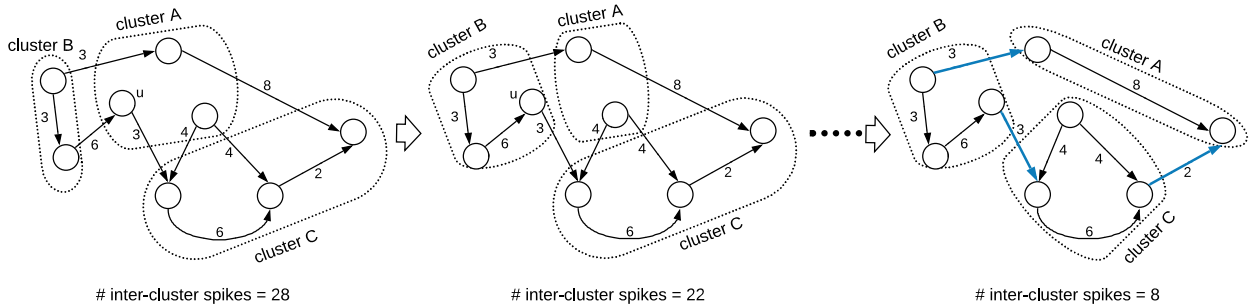


Fig. 20. An illustration of the steps of Kernighan–Lin clustering algorithm to partition an SNN into clusters.

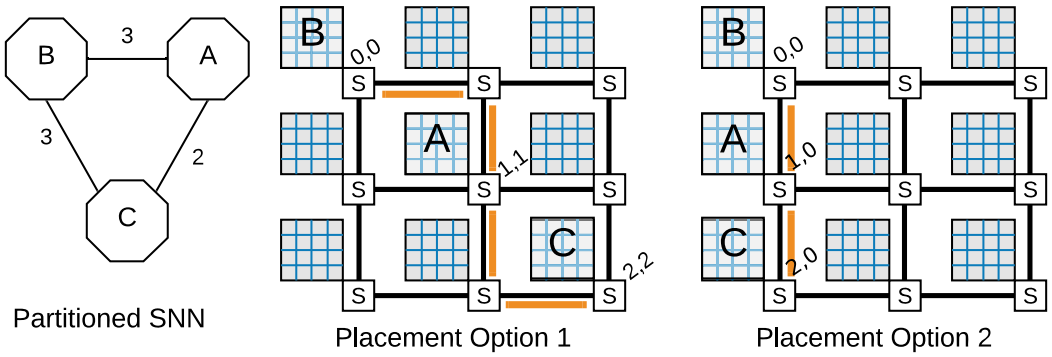


Fig. 21. Illustrating the impact of different placements of clusters of a clustered SNN on a neuromorphic hardware.

Spike data and model parameter weights can then be used to create a directed graph from the SNN. Such a directed graph can be represented as $G_{\text{SNN}} = (N, S)$, where N is the set of nodes representing neurons, and S is the set of edges representing synapses of the SNN model. Each edge $s_{ij} \in S$ connects neurons n_i and n_j. s_{ij} is associated with a synaptic strength p_{ij} and spike trains $spk(i)$. The weight of this synapse (for the purpose of partitioning) is $|spk(i)|$.

8.2. Determining the placement of clusters to cores of many-core neuromorphic hardware

Networks-on-chip (NoC) is the de-facto onchip communication infrastructure for many-core neuromorphic hardware platforms comprising of the physical layer, the data link layer and the network layer of the Open Systems Interconnection (OSI) protocol stack. Inside a NoC, neurosynaptic cores are connected to channels via switches, which are organized as mesh (a Manhattan-like structure). The addressing scheme used is based on Planar Cartesian coordinates, where each core specified using a pair of real numbers (called coordinates). Mapping an SNN to a NoC-based hardware consists of identifying the specific coordinate where each clusters of the SNN is to be placed.

Cluster placement on hardware plays an important role in energy and latency. To illustrate this Fig. 21 (left) shows a clustered SNN with three clusters (A, B, C). The figure also shows the number of spikes between these clusters. Imagine that the three clusters are mapped to the hardware using placement option 1 (middle) and placement option 2 (right).

For placement option 1, A is mapped to coordinate (1,1), B to (0,0), and C to (2,2). In this placement scheme, the three spikes between B and A travel via two hops, i.e., (0,0) → (0,1) and (0,1) → (1,1). Similarly, the two spikes between A and C via two hops, i.e., (1,1) → (2,1) and (2,1) → (2,2). Finally, the three spikes between B and C travel via four hops, i.e., (0,0) → (1,0), (1,0) → (2,0), (2,0) → (2,1), and (2,1) → (2,2).

Consider E_s and L_s be the energy and latency of each switch, while E_l and L_l be that for each NoC link, respectively. The average spike latency is

$$L = \frac{1}{3+2+3} \cdot \begin{cases} 3 \times (3 \cdot L_s + 2 \cdot L_l) + \\ /* \texttt{for spikes between A and B} \\ 2 \times (3 \cdot L_s + 2 \cdot L_l) + \\ /* \texttt{for spikes between A and C} \\ 3 \times (5 \cdot L_s + 4 \cdot L_l) \\ /* \texttt{for spikes between B and C} \end{cases} = \frac{30 \cdot L_s + 22 \cdot L_l}{8} \tag{6}$$

The total energy is

$$E = \begin{cases} 3 \times (3 \cdot E_s + 2 \cdot E_l) + \\ /* \texttt{for spikes between A and B} \\ 2 \times (3 \cdot E_s + 2 \cdot E_l) + \\ /* \texttt{for spikes between A and C} \\ 3 \times (5 \cdot E_s + 4 \cdot E_l) \\ /* \texttt{for spikes between B and C} \end{cases} = 30 \cdot E_s + 22 \cdot E_l \tag{7}$$

The same computations can be performed for placement option 2 (right). The average latency and total energy are

$$L = \frac{19 \cdot L_s + 11 \cdot L_l}{8} \qquad E = 19 \cdot E_s + 11 \cdot E_l \tag{8}$$

From this simple example we see that both average latency and total energy is lower for placement option 2 compared to placement option 1, illustrating that placement plays an important role in mapping SNNs to a neuromorphic hardware. Typically, heuristic solutions are used to obtain the placement of clusters to cores [190].

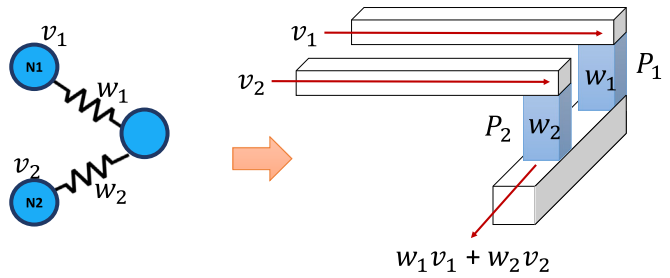


Fig. 22. Implementing a cluster on a crossbar.

8.3. Implementing each cluster on a crossbar

A cluster is implemented by placing its neurons and synapses on to the resources of a crossbar. Fig. 22 illustrates this implementation using a simple example of a 2×1 SNN on a 2×2 crossbar. Synaptic weights w_1 and w_2 are programmed into the NVM cells P1 and P2, respectively. The output spike voltages, v_1 from N1 and v_2 from N2, inject current into the crossbar, which is obtained by multiplying a pre-synaptic neuron's output spike voltage with the NVM cell's conductance at the cross-point of the pre- and post-synaptic neurons (following Ohm's law). Current summations along columns are performed in parallel using Kirchhoff's current law, and implement the sums $\sum_j w_i v_i$, needed for forward propagation of neuron excitation.

8.4. Mapping on limited precision hardware

Majority of SNN models presume that neuromorphic hardware can support the required precision and their mapping strategy mainly focuses on achieving communication cost reduction and performance improvements by employing various partitioning and spatiotemporal mapping [23]. However, neuromorphic hardware accelerator can have limited precision for various reasons. On one hand, the sparsity and inherent fault tolerance properties of SNNs are exploited to improve the area and energy-efficiency of the hardware by supporting limited precision such as approximation [193]. On the other hand, precision is limited due to various reliability and non-ideality issues resulting in imprecise/limited precision hardware. Therefore, a precision-aware mapping scheme can exploit the algorithm/model inherent error tolerance to map an SNN model into a limited precision neuromorphic hardware. Although limited precision hardware mapping is not explored adequately, there are a few mapping schemes such as quantization-aware mapping [194], and an un-balanced bit slicing and mapping [195] targeting limited precision hardware. Quantization-aware mapping quantize the SNN weights to match the precision of the targeted hardware. Whereas the un-balanced bit slicing targets non-ideal memristive-based CIM crossbar by employing a non-ideality aware un-balanced weight slicing technique and maps the slices into varying accuracy/precision memristive devices.

8.5. Recent advancement on SNN mapping

Mapping techniques for SNNs are largely adapted from those for traditional artificial neural networks (ANNs), with the additional consideration of the neuron's state, such as membrane potential, alongside input, output, and weights. Usually, SNN mapping focuses on partitioning neurons across distributed multi-core systems without shared memory, aiming to optimize resource utilization and minimize inter-core communication—an NP-hard problem [196]. Common heuristics include hill climbing [197], particle swarm optimization [198], and graph partitioning [199], with some methods considering hardware reliability and endurance [200]. Synchronous Dataflow Graphs (SDFG) are also used to estimate throughput but current methods struggle to handle SNNs exceeding on-chip memory or to effectively utilize shared on-chip memory.

For example, NeuProMa [201] splits, partitions, and maps SNNs to fit on-chip resources through time-multiplexing, optimizing intercore spike communication, but is limited by its splitting strategies and lack of shared memory consideration. Similarly, SMART [202] targets mapping DSNNs on resource-constrained neuromorphic systems with CPUs, addressing throughput estimation, operation and memory mapping, and parallel scheduling, though it lacks interlayer and intracore optimization. Advanced tools like Algorithm-to-Hardware Mapping (AHM) frameworks optimize layer mapping using analytical cost models, with Stream [203] extending Zigzag [204] to multilayer, multiaccelerator systems using a graph-based approach and genetic algorithms, enabling efficient exploration of Pareto-optimal mappings for substantial performance gains.

Recently, [205] introduced the first approach to map DSNNs onto digital neuromorphic architectures with inter-layer mapping optimization in space (depth-first) and time, improving hardware utilization and minimizing off-chip traffic. Applied to event-based vision benchmarks such as Gen4 [206] and CIFAR10-DVS [207], this method reduced external memory traffic by 12×, energy by 5×, and hidden states by 20× without accuracy loss, demonstrating significant efficiency gains for spatiotemporal feature learning under stringent on-chip memory constraints.

Deployment of SNNs in hardware introduces unique challenges, primarily due to the increased on-chip memory requirements for stateful neurons and synapses, which maintain persistent states. The limitations of on-chip memory, coupled with the high energy costs associated with off-chip memory access, have led to hybrid deployment strategies. In these approaches, persistent stateful elements are selectively used only where they significantly enhance system performance. For instance, in many vision processing tasks, the initial feature extraction layers of the neural network can be implemented using memory-less neurons and synapses, particularly in convolutional layers. This strategy significantly reduces on-chip memory usage while maintaining computational efficiency [208].

Another key practical challenge in deploying large-scale SNNs on multi-core neuromorphic processors is achieving efficient synchronization. Most existing spiking neuron models rely on time-step synchronization, as this facilitates training on GPUs. However, enforcing synchronization across neuromorphic cores introduces additional signaling overhead and can lead to stalling in dataflows. This adds complexity to the mapping process and requires careful architectural and algorithmic co-design to balance performance and energy efficiency [209]. Moreover, the limited precision of neuromorphic hardware poses an additional constraint, as conventional mapping strategies often assume sufficient numerical precision. Recent approaches such as QTMS [210] propose a quantization-aware mapping scheme that optimizes multiple timescale dynamics in SNNs to mitigate precision limitations while maintaining functional accuracy.

8.6. Conclusions

Despite advances in SNN mapping, key challenges persist. Existing heuristics, such as graph partitioning and evolutionary algorithms, provide approximate solutions but struggle with scalability, and current methods fail to fully leverage shared on-chip memory, leading to inefficiencies. Furthermore, the lack of standardized benchmarking methodologies makes it difficult to compare mapping techniques across different hardware platforms. Efforts such as NeuroBench [211] provide a unified framework for evaluating neuromorphic algorithms and hardware, but further refinement is needed for dynamic and continuous-learning applications. Future work should focus on hybrid mapping techniques that integrate deep reinforcement learning and optimization methods, alongside evolving benchmarks for closed-loop neuromorphic tasks and energy-aware metrics to ensure a fair and comprehensive performance evaluation.

9. Spiking Neural Networks deployment & applications

This section presents the recent developments on the adoption, deployment, and application of Spiking Neural Networks in different sectors. Section 9.1 presents the section that discusses the encoding of the input signal for SNN applications, which is an essential step to represent the input signal in usable format of SNN. Then, the application of neuromorphic systems in various domains is presented in Section 9.2.

9.1. Input signal encoding in SNN

The first step for implementing an SNN is to encode signals into a stream of spikes using either a rate-based model [45], some form of temporal coding [212], or population coding [213]. Unlike biological neural networks, artificial SNNs are highly simplified as the action potential generation, and the synaptic dynamics are usually simplified with binary events without including their complex temporal dynamics. Spike trains in a network of spiking neurons are propagated through weighted synaptic connections. A synapse can have a positive (excitatory) or negative (inhibitory) effect on the postsynaptic neuron. This means that it can increase or decrease the neuron's membrane potential. The strength of the synapse (weight) can be changed due to learning. The learning rule of an SNN is among the most challenging component for developing efficient and effective SNNs; this aspect is discussed in Section 5.

In general, the network topology describes how different neurons interact. Various factors determine the network model, such as the level of biological inspiration, their target application, and the limitations of the neuromorphic hardware.

Spiking neural networks (SNNs) can be used for a wide range of applications, including image recognition, speech recognition, robotics, and more. The specific topology of the SNN can depend on the specific application and the requirements of the system.

Feedforward SNNs are commonly used in image and speech recognition tasks, where the input signal is processed layer by layer to produce an output. Recurrent SNNs are often used in tasks that require memory or sequential processing, such as speech recognition [96] or language processing.

Bio-inspired SNN topologies aim to replicate the organization and structure of the brain. These topologies are often used for cognitive tasks, such as decision making [75], learning, and memory. Convolutional SNNs are commonly used in image and speech recognition tasks, where the input signal is processed in a way that mimics the organization of the visual or auditory cortex.

Gated-networks, such as long short-term memory (LSTM) networks, are used for tasks that require the network to maintain memory of previous inputs. This can be useful for tasks such as speech recognition or language processing.

Overall, the choice of topology depends on the specific application and the requirements of the system. Each topology has its advantages and disadvantages, and researchers are continually exploring new topologies and techniques to improve the performance of SNNs for a wide range of applications.

9.2. Neuromorphic applications

SNNs have also been shown to be energy-efficient and can be implemented in hardware using neuromorphic chips, making them attractive for applications in embedded systems and the Internet of Things (IoT). The neuromorphic community is currently engaged in defining benchmarks and application metrics to evaluate various neuromorphic solutions [24]. This evaluation considers both algorithmic complexity and hardware efficiency, helping to determine which tasks and applications make the neuromorphic engineering approach attractive. In addition several neuromorphic applications have demonstrated the benefits of the neuromorphic approach; these include:

- Event-based machine vision: neuromorphic event-based vision is probably the most developed application of neuromorphic technologies [214]. Today, event-based cameras find use in several applications in robotics, autonomous vehicles, and augmented reality. In robotics, event-based vision can be used for tasks such as object tracking and motion planning, where fast and accurate sensing is critical. In autonomous vehicles, event-based vision can help with obstacle avoidance and real-time decision-making. In augmented reality, event-based vision can provide a more natural and immersive user experience by allowing for real-time tracking and updating of the visual scene. One key advantage of event-based vision is its ability to operate in low-light conditions, which can be challenging for traditional cameras. This is because the spiking nature of the sensors allows them to operate more efficiently in low light, making them well-suited for applications such as surveillance and security.
- Speech recognition: SNNs have been used to recognize spoken words and phrases by analyzing the temporal patterns of neural activity associated with speech signals.
- Object recognition: SNNs have been applied to image recognition tasks, where they can learn to recognize objects based on their visual features and temporal patterns.
- Robotics: SNNs have been used to control robots and enable them to perform complex tasks, such as object manipulation and navigation.
- Computing: SNNs have been used to implement neuromorphic computing systems, which aim to emulate the energy-efficient and parallel processing capabilities of biological neural networks.
- Control applications: neuromorphic computing can be utilized to develop the control applications such as video games [215] and autonomous robot navigation [216].
- Prosthetics and brain-machine interfaces: neuromorphic-based systems have been proposed to smoothly interface with neural systems directly through the use of spikes, thus, they have the potential to realize low-latency and low-power neurprosthesics [167].

Overall, brain-inspired sensing and computing systems provide a promising method for modeling and comprehending biological neural systems while creating novel forms of artificial intelligence that can solve demanding tasks more efficiently and potentially with lower power consumption, using a more natural approach.

10. Challenges and future directions

A key obstacle in advancing neuromorphic computing systems lies in our incomplete understanding of the brain. Neuromorphic architectures draw inspiration from how the brain's structure and dynamics give rise to efficient, adaptive computation, so neuroscience discoveries directly shape the potential of neuromorphic technology. Gaining deeper insights into how the brain processes information, particularly its dynamical signal representations, will be crucial to refining neuromorphic systems.

Neuromorphic learning systems, which strive to learn and adapt as biological networks do, face several hurdles. The design of efficient learning algorithms is especially challenging: these algorithms must seamlessly handle large-scale, highly parallel, and time-varying neural activity while remaining energy efficient and scalable. Hardware limitations compound this issue; specialized neuromorphic chips and FPGAs often provide only limited memory and computational resources, restricting network size and complexity. Moreover, the inherent noise and variability in neuromorphic hardware complicate robust and reliable learning, while recurrent loops over time in spiking neural networks make interpreting learned representations notably difficult.

Developing analog mixed-signal circuits adds further complexity at the hardware core level. Noise and variability can undermine accuracy, requiring strategies such as self-healing and adaptive calibration.

At the same time, managing power consumption without sacrificing performance is critical, especially when working with novel nanoscale devices that can increase circuit complexity. In addition, ensuring interoperability with other hardware and software platforms necessitates well-defined standards and interfaces.

Memory constraints and scalability pose major architectural-level challenges. The memory subsystem often dominates power consumption and chip area, making Non-Volatile Memory (NVM) technologies attractive for their compact form factor and zero static power usage. However, high write power, low endurance, and read inefficiencies in current NVM solutions limit their practical deployment in large-scale neuromorphic systems.

Despite promising energy-efficiency achievements, for example, up to 195 Tera Operations per Joule [217] compared to roughly 10 Tera Synaptic Operations per Joule for the human brain [218] state-of-the-art neuromorphic processors remain far behind their biological counterparts. Intel's Pohoiki Springs system (100 million neurons and 99 billion synapses) can draw approximately 500 W [219], whereas the human brain, with tens of billions of neurons and trillions of synapses, operates only 10 to 20 W. Emerging technologies such as 3D integrated circuits can help close this gap by mitigating I/O overhead and improving system scalability [220].

Bridging the hardware–software divide is another critical step forward. While neuromorphic hardware emulates brain-like properties, powerful and flexible software frameworks are essential to harness those capabilities effectively. Finally, developing comprehensive applications and benchmarks will guide progress in neuromorphic sensing and computing [211]. Although initial successes in pattern recognition and robotics have been promising, ongoing research is needed to identify and test novel and sophisticated use cases. By continuing to push the boundaries in neuroscience, hardware innovations, software tools, and application development, the field of neuromorphic computing can move closer to the robust adaptability and efficiency seen in biological neural systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partially funded by the Dutch Organization for Scientific Research (NWO) through the Self-Healing Neuromorphic Systems (SNS) project (KICH1.ST04.22.021) and the NWA ORC project (NWO-NWA grant NWA.1292.19.298), and by the Swiss National Science Foundation, Switzerland (Grant Number PCEFP3_202981).

Data availability

No data was used for the research described in the article.

References

[1] C. Mead, M. Ismail, Analog VLSI Implementation of Neural Systems, Springer Science & Business Media, 1989, p. 80.
[2] G. Cauwenberghs, Reverse engineering the cognitive brain, Proc. Natl. Acad. Sci. 110 (39) (2013) 15 512–15 513.
[3] G. Indiveri, S.-C. Liu, Memory and information processing in neuromorphic systems, Proc. IEEE 103 (8) (2015) 1379–1397.
[4] T. Delbrück, B. Linares-Barranco, E. Culurciello, C. Posch, Activity-driven, event-based vision sensors, in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, IEEE, 2010, pp. 2426–2429.
[5] S.-C. Liu, A. Van Schaik, B.A. Minch, T. Delbruck, Event-based 64-channel binaural silicon cochlea with q enhancement mechanisms, in: 2010 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2010, pp. 2027–2030.
[6] C. Bartolozzi, Neuromorphic circuits impart a sense of touch, Science 360 (6392) (2018) 966–967.
[7] F. Corradi, D. Zambrano, M. Raglianti, G. Passetti, C. Laschi, G. Indiveri, Towards a neuromorphic vestibular system, IEEE Trans. Biomed. Circuits Syst. 8 (5) (2014) 669–680.
[8] J. Schemmel, A. Grübl, S. Hartmann, A. Kononov, C. Mayr, K. Meier, S. Millner, J. Partzsch, S. Schiefer, S. Scholze, et al., Live demonstration: A scaled-down version of the brainscales wafer-scale neuromorphic system, in: ISCAS, 2012.
[9] S. Furber, F. Galluppi, S. Temple, L.A. Plana, The SpiNNaker project, in: Proc. of the IEEE, 2014.
[10] B. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A.R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J.V. Arthur, P.A. Merolla, K. Boahen, Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, in: Proceedings of the IEEE, 2014.
[11] M.V. Debole, B. Taba, A. Amir, et al., TrueNorth: Accelerating from zero to 64 million neurons in 10 years, Computer (2019).
[12] L. Shi, J. Pei, N. Deng, D. Wang, L. Deng, Y. Wang, Y. Zhang, F. Chen, M. Zhao, S. Song, et al., Development of a neuromorphic computing system, in: IEDM, 2015.
[13] M. Davies, N. Srinivasa, T.H. Lin, et al., Loihi: A neuromorphic manycore processor with on-chip learning, IEEE Micro (2018).
[14] C. Frenkel, M. Lefebvre, J.-D. Legat, D. Bol, A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS, TBCAS (2018).
[15] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, Y. Xie, PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory, in: ISCA, 2016.
[16] S. Moradi, N. Qiao, F. Stefanini, G. Indiveri, A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs), TBCAS (2017).
[17] J. Stuijt, M. Sifalakis, A. Yousefzadeh, F. Corradi, μBrain: An event-driven and fully synthesizable architecture for spiking neural networks, Front. Neurosci. (2021).
[18] C.D. James, J.B. Aimone, N.E. Miner, C.M. Vineyard, F.H. Rothganger, K.D. Carlson, S.A. Mulder, T.J. Draelos, A. Faust, M.J. Marinella, et al., A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications, Biol. Inspir. Cogn. Archit. 19 (2017) 49–64.
[19] Y. Chen, H.H. Li, C. Wu, C. Song, S. Li, C. Min, H.-P. Cheng, W. Wen, X. Liu, Neuromorphic computing's yesterday, today, and tomorrow–an evolutional view, Integration 61 (2018) 49–61.
[20] F. Staudigl, F. Merchant, R. Leupers, A survey of neuromorphic computing-in-memory: Architectures, simulators, and security, IEEE Des. Test 39 (2) (2022) 90–99.
[21] M. Musisi-Nkambwe, S. Afshari, H. Barnaby, M. Kozicki, I.S. Esqueda, The viability of analog-based accelerators for neuromorphic computing: a survey, Neuromorphic Comput. Eng. 1 (1) (2021) 012001.
[22] H.F. Langroudi, T. Pandit, M. Indovina, D. Kudithipudi, Digital neuromorphic chips for deep learning inference: a comprehensive study, in: Applications of Machine Learning, Vol. 11139, SPIE, 2019, pp. 84–95.
[23] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, E. Beigne, Spiking neural networks hardware implementations and challenges: A survey, ACM J. Emerg. Technol. Comput. Syst. (JETC) 15 (2) (2019) 1–35.
[24] Z. Yu, A.M. Abdulghani, A. Zahid, H. Heidari, M.A. Imran, Q.H. Abbasi, An overview of neuromorphic computing for artificial intelligence enabled hardware-based hopfield neural network, IEEE Access 8 (2020) 67 085–67 099.
[25] D. Auge, J. Hille, E. Mueller, A. Knoll, A survey of encoding techniques for signal processing in spiking neural networks, Neural Process. Lett. 53 (6) (2021) 4693–4710.
[26] K. Demertzis, G.D. Papadopoulos, L. Iliadis, L. Magafas, A comprehensive survey on nanophotonic neural networks: Architectures, training methods, optimization, and activations functions, Sensors 22 (3) (2022) 720.
[27] C.D. Schuman, T.E. Potok, R.M. Patton, J.D. Birdwell, M.E. Dean, G.S. Rose, J.S. Plank, A survey of neuromorphic computing and neural networks in hardware, 2017, arXiv preprint arXiv:1705.06963.
[28] J.A. Gallego, M.G. Perich, L.E. Miller, S.A. Solla, Neural manifolds for the control of movement, Neuron 94 (5) (2017) 978–984.
[29] N. Axmacher, F. Mormann, G. Fernández, M.X. Cohen, C.E. Elger, J. Fell, Sustained neural activity patterns during working memory in the human medial temporal lobe, J. Neurosci. 27 (29) (2007) 7807–7816.
[30] J.F. Miller, M. Neufang, A. Solway, A. Brandt, M. Trippel, I. Mader, S. Hefft, M. Merkow, S.M. Polyn, J. Jacobs, et al., Neural activity in human hippocampal formation reveals the spatial context of retrieved memories, Science 342 (6162) (2013) 1111–1114.
[31] P. Viswanathan, A. Nieder, Neuronal correlates of a visual sense of number in primate parietal and prefrontal cortices, Proc. Natl. Acad. Sci. 110 (27) (2013) 11 187–11 192.
[32] X.-J. Wang, Probabilistic decision making by slow reverberation in cortical circuits, Neuron 36 (5) (2002) 955–968.

[33] M. Bear, B. Connors, M.A. Paradiso, Neuroscience: Exploring the Brain, Enhanced Edition: Exploring the Brain, Jones & Bartlett Learning, 2020.
[34] M. Van Spronsen, C.C. Hoogenraad, Synapse pathology in psychiatric and neurologic disease, Curr. Neurol. Neurosci. Rep. 10 (3) (2010) 207–214.
[35] T. Yamashita, A. Pala, L. Pedrido, Y. Kremer, E. Welker, C.C. Petersen, Membrane potential dynamics of neocortical projection neurons driving target-specific signals, Neuron 80 (6) (2013) 1477–1490.
[36] B.P. Bean, The action potential in mammalian central neurons, Nature Rev. Neurosci. 8 (6) (2007) 451–465.
[37] S. Paixão, R. Klein, Neuron–astrocyte communication and synaptic plasticity, Curr. Opin. Neurobiol. 20 (4) (2010) 466–473.
[38] W. Xin, Y.A. Mironova, H. Shen, R.A. Marino, A. Waisman, W.H. Lamers, D.E. Bergles, A. Bonci, Oligodendrocytes support neuronal glutamatergic transmission via expression of glutamine synthetase, Cell Rep. 27 (8) (2019) 2262–2271.
[39] A.E. Pereda, Electrical synapses and their functional interactions with chemical synapses, Nature Rev. Neurosci. 15 (4) (2014) 250–263.
[40] L. Khacef, P. Klein, M. Cartiglia, A. Rubino, G. Indiveri, E. Chicca, Spike-based local synaptic plasticity: A survey of computational models and neuromorphic circuits, 2022, arXiv preprint arXiv:2209.15536.
[41] D. Hebb, The Organization of Behavior, wiley, New York, 1949.
[42] N. Perez-Nieves, V.C. Leung, P.L. Dragotti, D.F. Goodman, Neural heterogeneity promotes robust learning, Nat. Commun. 12 (1) (2021) 5791.
[43] G.-q. Bi, M.-m. Poo, Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type, J. Neurosci. 18 (24) (1998) 10 464–10 472.
[44] M.S. Halvagal, F. Zenke, The combination of hebbian and predictive plasticity learns invariant object representations in deep sensory networks, 2022, bioRxiv, pp. 2022–03.
[45] W. Gerstner, W.M. Kistler, R. Naud, L. Paninski, Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition, Cambridge University Press, 2014.
[46] W. Gerstner, Spike-response model, Scholarpedia 3 (12) (2008) 1343, revision #91800.
[47] A. Hodgkin, A. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. 117 (1952) 500–544.
[48] A.N. Burkitt, A review of the integrate-and-fire neuron model: I. homogeneous synaptic input, Biol. Cybernet. 95 (2006).
[49] A. Roth, M.C. van Rossum, et al., Modeling synapses, Comput. Model. Methods Neurosci. 6 (2009) 139–160.
[50] J.S. Rothman, Modeling Synapses, Springer New York, New York, NY, 2013, pp. 1–15.
[51] S. La Barbera, D.R. Ly, G. Navarro, N. Castellani, O. Cueto, G. Bourgeois, B. De Salvo, E. Nowak, D. Querlioz, E. Vianello, Narrow heater bottom electrode-based phase change memory as a bidirectional artificial synapse, Adv. Electron. Mater. 4 (9) (2018) 1800223.
[52] C.D. Schuman, S.R. Kulkarni, M. Parsa, J.P. Mitchell, B. Kay, et al., Opportunities for neuromorphic computing algorithms and applications, Nat. Comput. Sci. 2 (1) (2022) 10–19.
[53] H. Markram, J. Lübke, M. Frotscher, B. Sakmann, Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps, Science 275 (5297) (1997) 213–215.
[54] C. Clopath, L. Büsing, E. Vasilaki, W. Gerstner, Connectivity reflects coding: a model of voltage-based stdp with homeostasis, Nature Neurosci. 13 (3) (2010) 344–352.
[55] F. Zenke, W. Gerstner, Hebbian plasticity requires compensatory processes on multiple timescales, Phil. Trans. R. Soc. B 372 (1715) (2017) 20160259, [Online]. Available: http://rstb.royalsocietypublishing.org/content/372/1715/20160259.
[56] N. Frémaux, W. Gerstner, Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules, Front. Neural Circuits 9 (2016).
[57] D. Zhao, Y. Zeng, T. Zhang, M. Shi, F. Zhao, Glsnn: A multi-layer spiking neural network based on global feedback alignment and local stdp plasticity, Front. Comput. Neurosci. 14 (2020) 576841.
[58] A. Samadzadeh, F.S.T. Far, A. Javadi, A. Nickabadi, M.H. Chehreghani, Convolutional spiking neural networks for spatio-temporal feature extraction, 2020, arXiv preprint arXiv:2003.12346.
[59] B. Yin, F. Corradi, S.M. Bohté, Accurate online training of dynamical spiking neural networks through forward propagation through time, Nat. Mach. Intell. 5 (5) (2023) 518–527.
[60] V. Demin, D. Nekhaev, Recurrent spiking neural network learning based on a competitive maximization of neuronal activity, Front. Neuroinformatics 12 (2018) 79.
[61] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[62] U. Kamath, J. Liu, J. Whitaker, Deep Learning for NLP and Speech Recognition, Springer, 2019, p. 84.
[63] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[64] T. Branco, K. Staras, The probability of neurotransmitter release: variability and feedback control at single synapses, Nature Rev. Neurosci. 10 (5) (2009) 373–383.
[65] E.T. Rolls, Memory, Attention, and Decision-Making, OUP, 2008, (Chapter 2).
[66] M.L. Varshika, F. Corradi, A. Das, Nonvolatile memories in spiking neural network architectures: Current and emerging trends, Electronics 11 (10) (2022) 1610.
[67] N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, P. Deaville, In-memory computing: Advances and prospects, IEEE Solid-State Circuits Mag. 11 (3) (2019) 43–55.
[68] S.B. Furber, F. Galluppi, S. Temple, L.A. Plana, The spinnaker project, Proc. IEEE 102 (5) (2014) 652–665.
[69] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S.H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al., Loihi: A neuromorphic manycore processor with on-chip learning, IEEE Micro 38 (1) (2018) 82–99.
[70] H. Esmaeilzadeh, E. Blem, R.S. Amant, K. Sankaralingam, D. Burger, Dark silicon and the end of multicore scaling, in: 2011 38th Annual International Symposium on Computer Architecture, ISCA, IEEE, 2011, pp. 365–376.
[71] R.J. Douglas, K.A. Martin, D. Whitteridge, A canonical microcircuit for neocortex, Neural Comput. 1 (4) (1989) 480–488.
[72] U. Rutishauser, R.J. Douglas, J.-J. Slotine, Collective stability of networks of winner-take-all circuits, Neural Comput. 23 (3) (2011) 735–773.
[73] M. Kouh, T. Poggio, A canonical neural circuit for cortical nonlinear operations, Neural Comput. 20 (6) (2008) 1427–1451.
[74] D.H. Hubel, T.N. Wiesel, Integrative action in the cat's lateral geniculate body, J. Physiol. 155 (2) (1961) 385.
[75] F. Corradi, H. You, M. Giulioni, G. Indiveri, Decision making and perceptual bistability in spike-based neuromorphic vlsi systems, in: 2015 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2015, pp. 2708–2711.
[76] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, G. Indiveri, A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses, Front. Neurosci. 9 (2015) 141.
[77] N. Imam, K. Wecker, J. Tse, R. Karmazin, R. Manohar, Neural spiking dynamics in asynchronous digital circuits, in: The 2013 International Joint Conference on Neural Networks, IJCNN, IEEE, 2013, pp. 1–8.
[78] S. Thorpe, D. Fize, C. Marlot, Speed of processing in the human visual system, Nature 381 (6582) (1996) 520–522.
[79] S. Wager, S. Wang, P.S. Liang, Dropout training as adaptive regularization, Adv. Neural Inf. Process. Syst. 26 (2013).
[80] G. Indiveri, B. Linares-Barranco, T.J. Hamilton, A.v. Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, et al., Neuromorphic silicon neuron circuits, Front. Neurosci. 5 (2011) 73.
[81] W. Maass, Networks of spiking neurons: the third generation of neural network models, Neural Netw. 10 (9) (1997) 1659–1671.
[82] R. Legenstein, D. Pecevski, W. Maass, A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback, PLoS Comput. Biol. 4 (10) (2008) e1000180.
[83] E. Oja, Simplified neuron model as a principal component analyzer, J. Math. Biol. 15 (3) (1982) 267–273, [Online]. Available: http://www.springerlink.com/content/u9u6120r003825u1/abstract/.
[84] A.J. Bell, L.C. Parra, Maximising information yields spike timing dependent plasticity, in: NIPS, 2004, pp. 121–128.
[85] C. Pehlevan, T. Hu, D.B. Chklovskii, A Hebbian/Anti-Hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data, Neural Comput. 27 (7) (2015) 1461–1495, [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/NECO_a_00745.
[86] S. Halvagal, F. Zenke, The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks, 2022, [Online]. Available: https://www.biorxiv.org/content/10.1101/2022.03.17.484712.
[87] C.D. Schuman, J.S. Plank, A. Disney, et al., An evolutionary optimization framework for neural networks and neuromorphic architectures, in: Joint Conference on ..., 2016.
[88] D. Zambrano, R. Nusselder, H.S. Scholte, S.M. Bohté, Sparse computation in adaptive spiking neural networks, Front. Neurosci. 12 (2018) 987.
[89] S. Davidson, S.B. Furber, Comparison of artificial and spiking neural networks on digital hardware, Front. Neurosci. 15 (2021) publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2021.651141/full.
[90] S.M. Bohte, J.N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (1–4) (2002) 17–37.
[91] T.C. Wunderlich, C. Pehle, Event-based backpropagation can compute exact gradients for spiking neural networks, Sci. Rep. 11 (1) (2021) 12829.
[92] T. Nowotny, J.P. Turner, J.C. Knight, Loss shaping enhances exact gradient learning with EventProp in spiking neural networks, 2022, arXiv:2212.01232 [cs]. [Online]. Available: http://arxiv.org/abs/2212.01232.

[93] J. Lee, S. Haghighatshoar, A. Karbasi, Exact gradient computation for spiking neural networks, in: OPT 2022: Optimization for Machine Learning, NeurIPS 2022 Workshop, arXiv preprint arXiv:2210.15415.

[94] S.M. Bohte, Error-backpropagation in networks of fractionally predictive spiking neurons, Artif. Neural Netw. Mach. Learn. (2011).

[95] J. Kim, K. Kim, J.-J. Kim, Unifying activation- and timing-based learning rules for spiking neural networks, 2020, 19 534–19 544.

[96] B. Yin, F. Corradi, S.M. Bohté, Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks, Nat. Mach. Intell. 3 (10) (2021) 905–913.

[97] F. Zenke, T.P. Vogels, The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks, Neural Comput. 33 (4) (2021) 899–925.

[98] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, W. Maass, A solution to the learning dilemma for recurrent networks of spiking neurons, Nat. Commun. 11 (1) (2020) 1–15.

[99] J. Keijser, H. Sprekeler, Optimizing interneuron circuits for compartment-specific feedback inhibition, PLoS Comput. Biol. 18 (4) (2022) e1009933.

[100] B. Yin, F. Corradi, S.M. Bohte, Accurate online training of dynamical spiking neural networks through forward propagation through time, Nat. Mach. Intell. (2023).

[101] E.F. Harkin, P.R. Shen, A. Goel, B.A. Richards, R. Naud, Parallel and recurrent cascade models as a unifying force for understanding subcellular computation, Neuroscience 489 (2022) 200–215, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306452221003808.

[102] R.J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. 1 (2) (1989) 270–280.

[103] F. Zenke, S. Ganguli, SuperSpike: Supervised learning in multilayer spiking neural networks, Neural Comput. 30 (6) (2018) 1514–1541, [Online]. Available: http://dx.doi.org/10.1162/neco_a_01086.

[104] J. Kaiser, H. Mostafa, E. Neftci, Synaptic plasticity dynamics for deep continuous local learning (DECOLLE), Front. Neurosci. 14 (2020) [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00424/full?report=reader.

[105] T. Bohnstingl, S. Woźniak, A. Pantazi, E. Eleftheriou, Online spatio-temporal learning in deep neural networks, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–15, conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[106] M. Xiao, Q. Meng, Z. Zhang, D. He, Z. Lin, Online training through time for spiking neural networks, 2022.

[107] F. Zenke, E.O. Neftci, Brain-inspired learning on neuromorphic substrates, Proc. IEEE 109 (5) (2021) 935–950, conference Name: Proceedings of the IEEE.

[108] F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, Proc. Mach. Learn. Res. 70 (2017) 3987–3995.

[109] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019300231.

[110] N. Imam, T.A. Cleland, Rapid online learning and robust recall in a neuromorphic olfactory circuit, Nat. Mach. Intell. 2 (3) (2020) 181–191.

[111] R.-J. Zhu, Q. Zhao, J.K. Eshraghian, SpikeGPT: Generative pre-trained language model with spiking neural networks, 2023.

[112] C. Pehle, S. Billaudelle, B. Cramer, J. Kaiser, K. Schreiber, Y. Stradmann, J. Weis, A. Leibfried, E. Müller, J. Schemmel, The brainscales-2 accelerated neuromorphic system with hybrid plasticity, Front. Neurosci. 16 (2022).

[113] J. Arthur, K. Boahen, Recurrently connected silicon neurons with active dendrites for one-shot learning, in: 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541), Vol. 3, 2004, pp. 1699–1704, vol. 3.

[114] T. Yu, G. Cauwenberghs, Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics, IEEE Trans. Biomed. Circuits Syst. 4 (3) (2010) 139–148, [Online]. Available: http://dx.doi.org/10.1109/TBCAS.2010.2048566.

[115] R.T. Edwards, G. Cauwenberghs, Synthesis of Log-Domain Filters from First-Order Building Blocks, Springer US, Boston, MA, 2000, pp. 71–80, [Online]. Available: http://dx.doi.org/10.1007/978-1-4757-6414-7_5.

[116] C. Frenkel, J.-D. Legat, D. Bol, Morphic: A 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning, IEEE Trans. Biomed. Circuits Syst. 13 (5) (2019) 999–1010.

[117] A. Yousefzadeh, G.-J. Van Schaik, M. Tahghighi, P. Detterer, S. Traferro, M. Hijdra, J. Stuijt, F. Corradi, M. Sifalakis, M. Konijnenburg, Seneca: Scalable energy-efficient neuromorphic computer architecture, in: 2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems, AICAS, IEEE, 2022, pp. 371–374.

[118] O. Moreira, A. Yousefzadeh, F. Chersi, G. Cinserin, R.-J. Zwartenkot, A. Kapoor, P. Qiao, P. Kievits, M. Khoei, L. Rouillard, et al., Neuronflow: a neuromorphic processor architecture for live ai applications, in: 2020 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2020, pp. 840–845.

[119] M. Davies, Taking neuromorphic computing to the next level with loihi 2, in: Intel Technology Brief, 2021.

[120] G. Di Patrizio Stanchieri, A. De Marcellis, M. Faccio, E. Palange, I. Isiksalan, T. Tufan, U. Guler, Decay time processing technique by digital architecture for noninvasive optical monitoring of biosignals in healthcare applications, in: 2025 IEEE Sensors Applications Symposium, SAS, 2025, pp. 1–6.

[121] M. Ward, O. Rhodes, Beyond lif neurons on neuromorphic hardware, Front. Neurosci. 16 (2022) [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2022.881598.

[122] C. Frenkel, J.-D. Legat, D. Bol, A compact phenomenological digital neuron implementing the 20 izhikevich behaviors, in: 2017 IEEE Biomedical Circuits and Systems Conference, BioCAS, 2017, pp. 1–4.

[123] S. Boyn, J. Grollier, G. Lecerf, B. Xu, N. Locatelli, S. Fusil, S. Girod, C. Carrétéro, K. Garcia, S. Xavier, et al., Learning through ferroelectric domain dynamics in solid-state synapses, Nat. Commun. 8 (1) (2017) 1–7.

[124] I. Muñoz-Martin, S. Bianchi, S. Hashemkhani, G. Pedretti, O. Melnic, D. Ielmini, A brain-inspired homeostatic neuron based on phase-change memories for efficient neuromorphic computing, Front. Neurosci. 15 (2021) [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2021.709053.

[125] P. Wijesinghe, A. Ankit, A. Sengupta, K. Roy, An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain, IEEE Trans. Emerg. Top. Comput. Intell. 2 (5) (2018) 345–358.

[126] H. Wang, N.C. Laurenciu, Y. Jiang, S.D. Cotofana, Ultra-compact, entirely graphene-based nonlinear leaky integrate-and-fire spiking neuron, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2020, pp. 1–5.

[127] H.-S.P. Wong, S. Raoux, S. Kim, J. Liang, J.P. Reifenberg, B. Rajendran, M. Asheghi, K.E. Goodson, Phase change memory, Proc. IEEE 98 (12) (2010) 2201–2227.

[128] A. Sebastian, M. Le Gallo, G.W. Burr, S. Kim, M. BrightSky, E. Eleftheriou, Tutorial: Brain-inspired computing using phase-change memory devices, J. Appl. Phys. 124 (11) (2018) 111101.

[129] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, E. Eleftheriou, Neuromorphic computing with multi-memristive synapses, Nat. Commun. 9 (1) (2018) 2514.

[130] H. Akinaga, H. Shima, Resistive random access memory (reram) based on metal oxides, Proc. IEEE 98 (12) (2010) 2237–2251.

[131] X. Xu, et al., Superior retention of low-resistance state in conductive bridge random access memory with single filament formation, IEEE Electron Device Lett. 36 (2015) 129–131.

[132] A. Siemon, T. Breuer, N. Aslam, S. Ferch, W. Kim, J. van den Hurk, V. Rana, S. Hoffmann-Eifert, R. Waser, S. Menzel, E. Linn, Realization of boolean logic functionality using redox-based memristive devices, Adv. Funct. Mater. 25 (2015) 6414–6423.

[133] N. Du, et al., Field-driven hopping transport of oxygen vacancies in memristive oxide switches with interface-mediated resistive switching, Phys. Rev. Appl. 10 (5) (2018) 054025.

[134] S. Roy, G. Niu, Q. Wang, Y. Wang, Y. Zhang, H. Wu, S. Zhai, P. Shi, S. Song, Z. Song, et al., Toward a reliable synaptic simulation using al-doped hfo2 rram, ACS Appl. Mater. Interfaces 12 (9) (2020) 10 648–10 656.

[135] J.-G. Zhu, Magnetoresistive random access memory: The path to competitiveness and scalability, Proc. IEEE 96 (11) (2008) 1786–1798.

[136] W. Zhao, E. Belhaire, C. Chappert, F. Jacquet, P. Mazoyer, New non-volatile logic based on spin-mtj, Phys. Status Solidi (A) 205 (6) (2008) 1373–1377, [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/pssa.200778135.

[137] R. Patel, E. Ipek, E.G. Friedman, 2T–1r stt-mram memory cells for enhanced on/off current ratio, Microelectron. J. 45 (2) (2014) 133–143.

[138] M. Sharad, C. Augustine, G. Panagopoulos, K. Roy, Spin-based neuron model with domain-wall magnets as synapse, IEEE Trans. Nanotechnol. 11 (4) (2012) 843–853.

[139] H. Wang, N.C. Laurenciu, Y. Jiang, S. Cotofana, Graphene-based artificial synapses with tunable plasticity, ACM J. Emerg. Technol. Comput. Syst. (JETC) 17 (4) (2021) 1–21.

[140] X. Wang, W. Xie, J.-B. Xu, Graphene based non-volatile memory devices, Adv. Mater. 26 (31) (2014) 5496–5503.

[141] T.F. Schranghamer, A. Oberoi, S. Das, Graphene memristive synapses for high precision neuromorphic computing, Nat. Commun. 11 (1) (2020) 1–11.

[142] E. Covi, H. Mulaosmanovic, B. Max, S. Slesazeck, T. Mikolajick, Ferroelectric-based synapses and neurons for neuromorphic computing, Neuromorphic Comput. Eng. (2022).

[143] J. Okuno, T. Kunihiro, K. Konishi, H. Maemura, Y. Shuto, F. Sugaya, M. Materano, T. Ali, K. Kuehnel, K. Seidel, et al., Soc compatible 1t1c feram memory array based on ferroelectric hf0. 5zr0. 5o2, in: 2020 IEEE Symposium on VLSI Technology, IEEE, 2020, pp. 1–2.

[144] Y. Arimoto, H. Ishiwara, Current status of ferroelectric random-access memory, Mrs Bull. 29 (11) (2004) 823–828.

[145] Y. Luo, Y.-C. Luo, S. Yu, A ferroelectric-based volatile/non-volatile dual-mode buffer memory for deep neural network accelerators, IEEE Trans. Comput. (2021).

[146] C. Frenkel, D. Bol, G. Indiveri, Bottom-up and top-down approaches for the design of neuromorphic processing systems: tradeoffs and synergies between natural and artificial intelligence, Proc. IEEE 111 (6) (2023) 623–652.

[147] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, S. Millner, A wafer-scale neuromorphic hardware system for large-scale neural modeling, in: 2010 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2010, pp. 1947–1950.

[148] E. Painkras, L.A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D.R. Lester, A.D. Brown, S.B. Furber, Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation, IEEE J. Solid-State Circuits 48 (8) (2013) 1943–1953.

[149] C. Mayr, S. Hoeppner, S. Furber, Spinnaker 2: A 10 million core processor system for brain simulation and machine learning, 2019, arXiv preprint arXiv: 1911.02385.

[150] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, et al., Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 34 (10) (2015) 1537–1557.

[151] L. Deng, G. Wang, G. Li, S. Li, L. Liang, M. Zhu, Y. Wu, Z. Yang, Z. Zou, J. Pei, et al., Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation, IEEE J. Solid-State Circuits 55 (8) (2020) 2228–2246.

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

[153] G.K. Chen, R. Kumar, H.E. Sumbul, P.C. Knag, R.K. Krishnamurthy, A 4096-neuron 1 m-synapse 3.8-pj/sop spiking neural network with on-chip stdp learning and sparse weights in 10-nm finfet cmos, IEEE J. Solid-State Circuits 54 (4) (2018) 992–1002.

[154] C. Frenkel, J.-D. Legat, D. Bol, A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2020, pp. 1–5.

[155] D. Wang, P.K. Chundi, S.J. Kim, M. Yang, J.P. Cerqueira, J. Kang, S. Jung, S. Kim, M. Seok, Always-on, sub-300-nw, event-driven spiking neural network based on spike-driven clock-generation and clock-and power-gating for an ultra-low-power intelligent device, in: 2020 IEEE Asian Solid-State Circuits Conference, A-SSCC, IEEE, 2020, pp. 1–4.

[156] C. Frenkel, G. Indiveri, Reckon: A 28 nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales, in: 2022 IEEE International Solid-State Circuits Conference, ISSCC, vol. 65, IEEE, 2022, pp. 1–3.

[157] S. Moradi, G. Indiveri, An event-based neural network architecture with an asynchronous programmable synaptic memory, IEEE Trans. Biomed. Circuits Syst. 8 (1) (2013) 98–107.

[158] E. Chicca, F. Stefanini, C. Bartolozzi, G. Indiveri, Neuromorphic electronic circuits for building autonomous cognitive systems, Proc. IEEE 102 (9) (2014) 1367–1388.

[159] S.A. Aamir, Y. Stradmann, P. Müller, C. Pehle, A. Hartel, A. Grübl, J. Schemmel, K. Meier, An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture, IEEE Trans. Circuits Syst. I. Regul. Pap. 65 (12) (2018) 4299–4312.

[160] J. Schemmel, S. Billaudelle, P. Dauer, J. Weis, Analog neuromorphic computing, in: Analog Circuits for Machine Learning, Current/Voltage/Temperature Sensors, and High-Speed Communication, Springer, 2022, pp. 83–102.

[161] X. Peng, S. Huang, H. Jiang, A. Lu, S. Yu, Dnn+ neurosim v2. 0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (11) (2020) 2306–2319.

[162] A. Neckar, S. Fok, B.V. Benjamin, T.C. Stewart, N.N. Oza, A.R. Voelker, C. Eliasmith, R. Manohar, K. Boahen, Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model, Proc. IEEE 107 (1) (2018) 144–164.

[163] B. Cramer, S. Billaudelle, S. Kanya, A. Leibfried, A. Grübl, V. Karasenko, C. Pehle, K. Schreiber, Y. Stradmann, J. Weis, J. Schemmel, F. Zenke, Surrogate gradients for analog neuromorphic computing, Proc. Natl. Acad. Sci. 119 (4) (2022) [Online]. Available: https://www.pnas.org/content/119/4/e2109194119.

[164] J.-s. Seo, B. Brezzo, Y. Liu, B.D. Parker, S.K. Esser, R.K. Montoye, B. Rajendran, J.A. Tierno, L. Chang, D.S. Modha, et al., A 45 nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons, in: 2011 IEEE Custom Integrated Circuits Conference, CICC, IEEE, 2011, pp. 1–4.

[165] B. Rueckauer, C. Bybee, R. Goettsche, Y. Singh, J. Mishra, A. Wild, Nxtf: An api and compiler for deep spiking neural networks on intel loihi, ACM J. Emerg. Technol. Comput. Syst. (JETC) 18 (3) (2022) 1–22.

[166] J. Göltz, L. Kriener, A. Baumbach, S. Billaudelle, O. Breitwieser, B. Cramer, D. Dold, A.F. Kungl, W. Senn, J. Schemmel, et al., Fast and energy-efficient neuromorphic deep learning with first-spike times, Nat. Mach. Intell. 3 (9) (2021) 823–835.

[167] F. Corradi, G. Indiveri, A neuromorphic event-based neural recording system for smart brain-machine-interfaces, IEEE Trans. Biomed. Circuits Syst. 9 (5) (2015) 699–709.

[168] F.C. Bauer, D.R. Muir, G. Indiveri, Real-time ultra-low power ecg anomaly detection using an event-driven neuromorphic processor, IEEE Trans. Biomed. Circuits Syst. 13 (6) (2019) 1575–1582.

[169] J. Büchel, D. Zendrikov, S. Solinas, G. Indiveri, D.R. Muir, Supervised training of spiking neural networks for robust deployment on mixed-signal neuromorphic processors, Sci. Rep. 11 (1) (2021) 1–12.

[170] S. Billaudelle, Y. Stradmann, K. Schreiber, B. Cramer, A. Baumbach, D. Dold, J. Göltz, A.F. Kungl, T.C. Wunderlich, A. Hartel, et al., Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2020, pp. 1–5.

[171] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G.A.F. Guerra, P. Joshi, P. Plank, S.R. Risbud, Advancing neuromorphic computing with loihi: A survey of results and outlook, Proc. IEEE 109 (5) (2021) 911–934.

[172] J. Park, T. Yu, S. Joshi, C. Maier, G. Cauwenberghs, Hierarchical address event routing for reconfigurable large-scale neuromorphic systems, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2016) 2408–2422.

[173] M. Horowitz, 1.1 computing's energy problem (and what we can do about it), in: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers, ISSCC, IEEE, 2014, pp. 10–14.

[174] C. Mayr, J. Partzsch, M. Noack, S. Hänzsche, S. Scholze, S. Höppner, G. Ellguth, R. Schüffny, A biological-realtime neuromorphic system in 28 nm cmos using low-leakage switched capacitor circuits, IEEE Trans. Biomed. Circuits Syst. 10 (1) (2015) 243–254.

[175] J. Park, J. Lee, D. Jeon, A 65-nm neuromorphic image classification processor with energy-efficient training through direct spike-only feedback, IEEE J. Solid-State Circuits 55 (1) (2019) 108–119.

[176] D.S. Bassett, E. Bullmore, Small-world brain networks, Neurosci. 12 (6) (2006) 512–523.

[177] S. Thorpe, A. Delorme, R. Van Rullen, Spike-based strategies for rapid processing, Neural Netw. 14 (6–7) (2001) 715–725.

[178] F.N. Buhler, P. Brown, J. Li, T. Chen, Z. Zhang, M.P. Flynn, A 3.43 tops/w 48.9 pj/pixel 50.1 nj/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40 nm cmos, in: 2017 Symposium on VLSI Circuits, IEEE, 2017, pp. C30–C31.

[179] C. Bartolozzi, G. Indiveri, E. Donati, Embodied neuromorphic intelligence, Nat. Commun. 13 (1) (2022) 1024.

[180] D. Kudithipudi, M. Aguilar-Simon, J. Babb, M. Bazhenov, D. Blackiston, J. Bongard, A.P. Brna, S. Chakravarthi Raja, N. Cheney, J. Clune, et al., Biological underpinnings for lifelong learning machines, Nat. Mach. Intell. 4 (3) (2022) 196–210.

[181] A. Nøkland, Direct feedback alignment provides learning in deep neural networks, Adv. Neural Inf. Process. Syst. 29 (2016).

[182] C. Frenkel, M. Lefebvre, D. Bol, Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks, Front. Neurosci. 15 (2021) 629892.

[183] F.T. Zohora, V. Karia, A.R. Daram, A.M. Zyarah, D. Kudithipudi, Metaplasticnet: Architecture with probabilistic metaplastic synapses for continual learning, in: 2021 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2021, pp. 1–5.

[184] V. Karia, F.T. Zohora, N. Soures, D. Kudithipudi, Scolar: A spiking digital accelerator with dual fixed point for continual learning, in: 2022 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2022, pp. 1372–1376.

[185] S. D'agostino, F. Moro, T. Torchet, Y. Demirağ, L. Grenouillet, N. Castellani, G. Indiveri, E. Vianello, M. Payvand, Denram: neuromorphic dendritic architecture with rram for efficient temporal processing with delays, Nat. Commun. 15 (1) (2024) 3446.

[186] A. Balaji, P.K. Huynh, F. Catthoor, N.D. Dutt, J.L. Krichmar, A. Das, Neusb: A scalable interconnect architecture for spiking neuromorphic hardware, IEEE Trans. Emerg. Top. Comput. (2023).

[187] X. Liu, W. Wen, X. Qian, H. Li, Y. Chen, Neu-noc: A high-efficient interconnection network for accelerated neuromorphic systems, in: 2018 23rd Asia and South Pacific Design Automation Conference, ASP-DAC, IEEE, 2018, pp. 141–146.

[188] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J. 49 (2) (1970) 291–307.

[189] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, S. Schaafsma, Mapping of local and global synapses on spiking neuromorphic hardware, in: 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2018, pp. 1217–1222.

[190] A. Balaji, A. Das, Y. Wu, K. Huynh, F.G. Dell'Anna, G. Indiveri, J.L. Krichmar, N.D. Dutt, S. Schaafsma, F. Catthoor, Mapping spiking neural networks to neuromorphic hardware, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 28 (1) (2019) 76–86.

[191] L. Niedermeier, K. Chen, J. Xing, A. Das, J. Kopsick, E. Scott, N. Sutton, K. Weber, N. Dutt, J.L. Krichmar, Carlsim 6: An open source library for large-scale, biologically detailed spiking neural network simulation, in: 2022 International Joint Conference on Neural Networks, IJCNN, IEEE, 2022, pp. 1–10.

[192] D.F. Goodman, R. Brette, The brian simulator, Front. Neurosci. (2009) 26.

[193] A. Gebregirogis, M. Tahoori, Approximate learning and fault-tolerant mapping for energy-efficient neuromorphic systems, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 26 (3) (2020) 1–23.

[194] M.P.E. Apolinario, A.K. Kosta, U. Saxena, K. Roy, Hardware/software co-design with adc-less in-memory computing hardware for spiking neural networks, IEEE Trans. Emerg. Top. Comput. 12 (1) (2023) 35–47.
[195] S. Diware, A. Singh, A. Gebregiorgis, R.V. Joshi, S. Hamdioui, R. Bishnoi, Accurate and energy-efficient bit-slicing for rram-based neural networks, IEEE Trans. Emerg. Top. Comput. Intell. 7 (1) (2022) 164–177.
[196] P.K. Huynh, M.L. Varshika, A. Paul, M. Isik, A. Balaji, A. Das, Implementing spiking neural networks on neuromorphic architectures: A review, 2022.
[197] T. Titirsha, A. Das, Thermal-aware compilation of spiking neural networks to neuromorphic hardware, in: International Workshop on Languages and Compilers for Parallel Computing, Springer, 2020, pp. 134–150.
[198] A. Das, Y. Wu, K. Huynh, F. Dell'Anna, F. Catthoor, S. Schaafsma, Mapping of local and global synapses on spiking neuromorphic hardware, in: 2018 Design, Automation & Test in Europe Conference & Exhibition, DATE, 2018, pp. 1217–1222.
[199] S. Eissa, S. Stuijk, H. Corporaal, Dnasim: Evaluation framework for digital neuromorphic architectures, in: 2022 25th Euromicro Conference on Digital System Design, DSD, 2022, pp. 438–445.
[200] T. Titirsha, S. Song, A. Das, J. Krichmar, N. Dutt, N. Kandasamy, F. Catthoor, Endurance-aware mapping of spiking neural networks to neuromorphic hardware, IEEE Trans. Parallel Distrib. Syst. 33 (2) (2021) 288–301.
[201] C. Xiao, J. Chen, L. Wang, Neuproma: A toolchain for mapping large-scale spiking convolutional neural networks onto neuromorphic processor, in: S. Liu, X. Wei (Eds.), Network and Parallel Computing, Springer Nature Switzerland, Cham, 2022, pp. 129–142.
[202] A. Das, Real-time scheduling of machine learning operations on heterogeneous neuromorphic soc, 2022.
[203] A. Symons, L. Mei, S. Colleman, P. Houshmand, S. Karl, M. Verhelst, Stream: A modeling framework for fine-grained layer fusion on multi-core dnn accelerators, in: 2023 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, 2023, pp. 355–357.
[204] L. Mei, P. Houshmand, V. Jain, S. Giraldo, M. Verhelst, Zigzag: Enlarging joint architecture-mapping design space exploration for dnn accelerators, IEEE Trans. Comput. 70 (8) (2021) 1160–1174.
[205] S. Eissa, S. Stuijk, F. de Putter, A. Nardi-Dei, F. Corradi, H. Corporaal, Stems: Spatial–temporal mapping for spiking neural networks, IEEE Trans. Comput. 74 (9) (2025) 2991–3002.
[206] E. Perot, P. de Tournemire, D. Nitti, J. Masci, A. Sironi, Learning to detect objects with a 1 megapixel event camera, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020.
[207] H. Li, H. Liu, X. Ji, G. Li, L. Shi, Cifar10-dvs: An event-stream dataset for object classification, Front. Neurosci. 11 (2017) [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2017.00309.
[208] Y. Xu, K. Shidqi, G.-J. van Schaik, R. Bilgic, A. Dobrita, S. Wang, R. Meijer, P. Nembhani, C. Arjmand, P. Martinello, et al., Optimizing event-based neural networks on digital neuromorphic architecture: a comprehensive design space exploration, Front. Neurosci. 18 (2024) 1335422.
[209] R. Koopman, A. Yousefzadeh, M. Shahsavari, G. Tang, M. Sifalakis, Overcoming the limitations of layer synchronization in spiking neural networks, 2024, arXiv preprint arXiv:2408.05098.
[210] S. Eissa, F. Corradi, F. de Putter, S. Stuijk, H. Corporaal, Qmts: Fixed-point quantization for multiple-timescale spiking neural networks, in: International Conference on Artificial Neural Networks, Springer, 2023, pp. 407–419.
[211] J. Yik, K. Van den Berghe, D. den Blanken, Y. Bouhadjar, M. Fabre, P. Hueber, W. Ke, M.A. Khoei, D. Kleyko, N. Pacik-Nelson, et al., The neurobench framework for benchmarking neuromorphic computing algorithms and systems, Nat. Commun. 16 (1) (2025) 1545.
[212] S. Park, S. Kim, B. Na, S. Yoon, T2fsnn: Deep spiking neural networks with time-to-first-spike coding, in: 2020 57th ACM/IEEE Design Automation Conference, DAC, IEEE, 2020, pp. 1–6.
[213] S.M. Bohte, H. La Poutré, J.N. Kok, Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer rbf networks, IEEE Trans. Neural Netw. 13 (2) (2002) 426–435.
[214] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A.J. Davison, J. Conradt, K. Daniilidis, et al., Event-based vision: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (1) (2020) 154–180.
[215] C. Tang, Z. Wang, X. Sima, L. Zhang, Research on artificial intelligence algorithm and its application in games, in: 2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture, AIAM, 2020, pp. 386–389.
[216] J.D. Schaffer, Evolving spiking neural networks for robot sensory-motor decision tasks of varying difficulty, in: Proceedings of the Neuro-Inspired Computational Elements Workshop, NICE '20, Association for Computing Machinery, New York, NY, USA, 2020, [Online]. Available: http://dx.doi.org/10.1145/3381755.3381757.
[217] C.-X. Xue, J.-M. Hung, H.-Y. Kao, Y.-H. Huang, S.-P. Huang, F.-C. Chang, P. Chen, T.-W. Liu, C.-J. Jhang, C.-I. Su, et al., 16.1 a 22 nm 4 mb 8b-precision reram computing-in-memory macro with 11.91 to 195.7 tops/w for tiny ai edge devices, in: 2021 IEEE International Solid-State Circuits Conference, ISSCC, vol. 64, IEEE, 2021, pp. 245–247.
[218] T. Delbruck, S.-C. Liu, Data-driven neuromorphic dram-based cnn and rnn accelerators, in: 2019 53rd Asilomar Conference on Signals, Systems, and Computers, IEEE, 2019, pp. 500–506.
[219] Y.S. Yang, Y. Kim, Recent trend of neuromorphic computing hardware: Intel's neuromorphic system perspective, in: 2020 International SoC Design Conference, ISOCC, IEEE, 2020, pp. 218–219.
[220] S. Sinha, X. Xu, M. Bhargava, S. Das, B. Cline, G. Yeric, Stack up your chips: Betting on 3d integration to augment moore's law scaling, 2020, arXiv preprint arXiv:2005.10866.

Dr. Anteneh Gebregiorgis received the Ph.D. degree in computer science from Karlsruhe Institute of Technology (KIT), Germany, in 2019. He is currently an assistant professor in the Department of Quantum and Computer Engineering, Delft University of Technology, The Netherlands. From 2017 to 2018 he was a visiting scholar with the Nanoelectronics Research Laboratory, Purdue University, working on energy-efficient neuromorphic architectures. His research focuses on reliable and energy-efficient system design for neuromorphic applications using emerging devices and unconventional computing paradigms.

Dr. Amirreza Yousefzadeh is an Assistant Professor at the University of Twente, specializing in embedded AI and neuromorphic systems. His research focuses on the co-design of algorithms and hardware for efficient, reliable, and adaptive machine intelligence, with applications in next-generation computing and smart embedded devices.

Sherif Eissa received the B.Sc. and M.Sc. degrees cum laude in electrical and electronics engineering from the German University in Cairo, in 2016, and the University of Stuttgart, in 2019. He is currently a neuromorphic concept engineer at Innatera Nanosystems, as well as a Ph.D. researcher in neuromorphic computing at Eindhoven University of Technology (TU/e). His research interests include edge AI, computer architecture, AI algorithms, and HW-SW co-design.

Dr. Muhammad Ali Siddiqi is an Assistant Professor of Electrical Engineering at the Lahore University of Management Sciences (LUMS), Pakistan. He holds a Ph.D. from Erasmus University Medical Center, Rotterdam, on the security and privacy of Implantable Medical Devices. He has worked as a postdoctoral researcher at TU Delft on in-memory computing for lightweight cryptography and neural implants, and as a Design Engineer at Silicon Labs Norway. He is a TPC member of the Reconfigurable Architectures Workshop (RAW). His research interests include low-power hardware design, brain–machine interfaces, and the cybersecurity of implantable and wearable devices.

Dr. Charlotte Frenkel received the M.Sc. degree (summa cum laude) in electromechanical engineering and the Ph.D. degree in engineering science from the Universite catholique de Louvain (UCLouvain), Louvainla-Neuve, Belgium, in 2015 and 2020, respectively. In 2020, she joined the Institute of Neuroinformatics, UZH and ETH Zurich, Switzerland, as a Postdoctoral Researcher. She is an Assistant Professor with Delft University of Technology, Delft, The Netherlands, since 2022, and holds a Visiting Faculty Researcher position with Google since 2024. Her research aims at bridging the bottom-up (bio-inspired) and top-down (engineering-driven) design approaches toward neuromorphic intelligence, with a focus on hardware algorithm co-design for (Neuro)AI, digital hardware accelerators, and brain inspired on-device learning. She received a best paper award at the IEEE International Symposium on Circuits and Systems (ISCAS) 2020 conference in the Neural Networks track, and her

Ph.D. thesis was awarded the FNRSFWO/Nokia Bell Scientific Award 2021 and the FNRS-FWO/IBM Innovation Award 2021. In 2023, she was awarded prestigious Veni and AiNed Fellowship Grants from the Dutch Research Council (NWO). She presented several invited talks, including keynotes at the tinyML EMEA technical forum 2021 and at the Neuro-Inspired Computational Elements (NICE) neuromorphic conference 2021. She serves or has served as Program Co-Chair of NICE 2023–2024 and of the tinyML Research Symposium 2024, as a Co-Lead of the NeuroBench initiative for benchmarks in neuromorphic computing since 2022, as a TPC Member of IEEE ESSERC for 2022–2024, and as an Associate Editor for IEEE Transation on Biomedical Circuits and Systems since 2022.

Dr. Friedemann Zenke is a research group leader at the Friedrich Miescher Institute for Biomedical Research and an assistant professor at the University of Basel, Switzerland. He received his Ph.D. in computational neuroscience from EPFL in 2014. His research focuses on theoretical neuroscience and neuromorphic algorithms, particularly biologically inspired learning, efficient neural networks, and unsupervised learning.

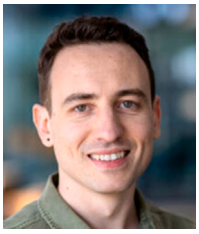
Prof. Dr. Sander M. Bohté heads the CWI Machine Learning group, and is also a part-time full professor of Cognitive Computational Neuroscience at the University of Amsterdam, The Netherlands. He received his Ph.D. in 2003 at CWI on the topic of "Spiking Neural Networks". He was then awarded an NWO TALENT grant, which he spent with Michael Mozer at the University of Colorado in Boulder. In 2004, he rejoined CWI as junior permanent staff to work on distributed spiking neural network models and multi-agent systems. In 2016, he co-founded the CWI Machine Learning group, where his research bridges the field of neuroscience with applications thereof as advanced neural networks. His work has been pioneering in the development of advanced and efficient spiking neural networks, including seminar work on supervised learning with spike-time coded networks. Recent work has also developed biologically plausible deep learning and deep reinforcement learning models for cognition, and spiking neural network versions thereof.

Dr. Abdulqader Mahmoud is a Test Engineer at NXP Semiconductors in The Netherlands, where he leads pre-silicon simulation and testing activities for automotive mixed-signal circuits. He previously worked as an Analog Circuit Designer, developing high-voltage circuits for nextgeneration automotive applications. Dr. Mahmoud earned his B.Sc. and M.Sc. degrees in Electrical and Computer Engineering from Khalifa University, UAE, and completed his Ph.D. in Computer Engineering at Delft University of Technology, The Netherlands, where his research focused on spin-wave-based circuit design. He has authored more than 25 peer-reviewed publications and holds a patent on a two-dimensional charge pump. His research interests include neuromorphic computing, emerging device technologies, and energy-efficient circuit design.

Dr. Anup Das is an Associate Professor at Drexel University. He received a Ph.D. in Embedded Systems from the National University of Singapore in 2014. He held positions at the University of Southampton and IMEC, leading projects on neuromorphic computing. He received the NSF/DARPA RTML award (2019), NSF CAREER (2020), and DOE CAREER (2021). His research focuses on operating systems for neuromorphic hardware, and the dependability and security of neuromorphic computing. He is a Senior Member of IEEE.

Prof. Said Hamdioui received the M.S.E.E. and Ph.D. (Hons.) degrees from Delft University of Technology, The Netherlands. He is Chair Professor of dependable and emerging computer technologies and Head of the CE-Lab, TU Delft. Previously, he worked at Intel (Santa Clara), Philips Semiconductors (France), and Philips/NXP (Nijmegen). He has authored over 250 papers and holds patents in memory testing. His research covers CMOS reliability, hardware security, 3D-ICs, memristors, and in-memory computing. He has received multiple international awards and best paper prizes.

Prof. Henk Corporaal is Professor in Embedded System Architectures at the Eindhoven University of Technology (TU/e) in The Netherlands. He has gained a M.Sc. in Theoretical Physics from the University of Groningen, and a Ph.D. in Electrical Engineering, in the area of Computer Architecture, from Delft University of Technology. His research is on low power multi-processor, heterogeneous processing architectures, their programmability, and the predictable design of soft- and hard real-time systems. This includes research and design of embedded system architectures, including CGRAs, SIMD, VLIW and GPUs, on accelerators, the exploitation of all kinds of parallelism, fault-tolerance, approximate computing, architectures for machine and deep learning, optimizations and mapping of deep learning networks, and the (semi-) automated mapping of applications to these architectures. Corporaal has co-authored over 500 journal and conference papers. Furthermore he invented a new class of VLIW architectures, the Transport Triggered Architectures, which is used in several commercial products, and by many research groups. He initiated the Dutch NWO perspectief program on Efficient Deep Learning (efficient-deeplearning.nl); in this program many research institutes and over 30 companies participated. He also is the PI of the EU project CONVOLVE (convolve.eu) on seamless design of smart edge processors, with 19 partners. For further details see corporaal.org.



Dr. Federico Corradi received the Ph.D. degree in Neuroinformatics from the University of Zurich and the ETH Neuroscience Centre Zurich, Switzerland, in 2015. From 2012 to 2017, he was with iniLabs, Zurich, where he contributed to the design of several neuromorphic integrated processors and event-driven sensors. From 2017 to 2019, he was with IMEC, Eindhoven, The Netherlands, where he worked on neuromorphic architectures and sensor interfaces bridging academic and industrial research. He, is currently an Assistant Professor with the Department of Electrical Engineering at Eindhoven University of Technology (TU/e), The Netherlands, where he leads the Neuromorphic Edge Computing Systems Laboratory. His research interests include neuromorphic computing, asynchronous circuits, and mixed-signal design for spiking neural networks, with applications in robotics and biomedical sensing. He serves as an Associate Editor for Elsevier Microprocessors & Microsystems and as a member of the technical program committees of international conferences such as ICONS, IJCNN, NEWCAS, and ISCAS. He has delivered invited talks at international venues, including the Korea Semiconductor Academy (2025), the BrainInspiration Conference (2024), and the International Conference on Field-Programmable Logic and Applications (FIRE 2025), among others. Dr. Corradi received the ISCAS Honorary Mention of the NSA Track in 2014 and was recognized as a Rising Star in Neuromorphic Computing & Engineering by IOP Science in 2025. He actively promotes open-source neuromorphic design methodologies and interdisciplinary education at the intersection of hardware and brain-inspired computing.