Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware

NITIN RATHI, INDRANIL CHAKRABORTY, and ADARSH KOSTA, School of Electrical and Computer Engineering, Purdue University, USA
ABHRONIL SENGUPTA, School of Electrical Engineering and Computer Science, Pennsylvania State University, USA
AAYUSH ANKIT, School of Electrical and Computer Engineering, Purdue University, USA
PRIYADARSHINI PANDA, Electrical Engineering, Yale University, USA
KAUSHIK ROY, School of Electrical and Computer Engineering, Purdue University, USA

Neuromorphic Computing, a concept pioneered in the late 1980s, is receiving a lot of attention lately due to its promise of reducing the computational energy, latency, as well as learning complexity in artificial neural networks. Taking inspiration from neuroscience, this interdisciplinary field performs a multi-stack optimization across devices, circuits, and algorithms by providing an end-to-end approach to achieving brain-like efficiency in machine intelligence. On one side, neuromorphic computing introduces a new algorithmic paradigm, known as Spiking Neural Networks (SNNs), which is a significant shift from standard deep learning and transmits information as spikes ("1" or "0") rather than analog values. This has opened up novel algorithmic research directions to formulate methods to represent data in spike-trains, develop neuron models that can process information over time, design learning algorithms for event-driven dynamical systems, and engineer network architectures amenable to sparse, asynchronous, event-driven computing to achieve lower power consumption. On the other side, a parallel research thrust focuses on development of efficient computing platforms for new algorithms. Standard accelerators that are amenable to deep learning workloads are not particularly suitable to handle processing across multiple timesteps efficiently. To that effect, researchers have designed neuromorphic hardware that rely on event-driven sparse computations as well as efficient matrix operations. While most large-scale neuromorphic systems have been explored based on CMOS technology, recently, Non-Volatile Memory (NVM) technologies show promise toward implementing bio-mimetic functionalities on single devices. In this article, we outline several strides that neuromorphic computing based on spiking neural networks (SNNs) has taken over the recent past, and we present our outlook on the challenges that this field needs to overcome to make the bio-plausibility route a successful one.

CCS Concepts: • **Theory of computation → Design and analysis of algorithms**; • **Computing methodologies → Bio-inspired approaches;**

N. Rathi and I. Chakraborty contributed equally to this research.

This work was supported in part by the Center for Brain-inspired Computing Enabling Autonomous Intelligence (C-BRIC), one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, in part by the National Science Foundation, in part by Intel, in part by the ONR-MURI program, and in part by the Vannevar Bush Faculty Fellowship.

Authors' addresses: N. Rathi, I. Chakraborty, A. Kosta, A. Ankit, and K. Roy, School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47906; emails: {rathi2, ichakra, akosta, aankit, kaushik}@purdue.edu; A. Sengupta, School of Electrical Engineering and Computer Science, Pennsylvania State University, State College, PA, 16801; email: sengupta@psu.edu; P. Panda, Electrical Engineering, Yale University, New Haven, CT, 06511; email: priya.panda@yale.edu.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2023 Copyright held by the owner/author(s).

0360-0300/2023/03-ART243 $15.00
https://doi.org/10.1145/3571155

Additional Key Words and Phrases: Neuromorphic Computing, Spiking Neural Networks, bio-plausible learning, spike-based backpropagation, event cameras, In-Memory Computing, Non-Volatile Memories, asynchronous communication

ACM Reference format:
Nitin Rathi, Indranil Chakraborty, Adarsh Kosta, Abhronil Sengupta, Aayush Ankit, Priyadarshini Panda, and Kaushik Roy. 2023. Exploring Neuromorphic Computing Based on Spiking Neural Networks: Algorithms to Hardware. *ACM Comput. Surv.* 55, 12, Article 243 (March 2023), 49 pages.
https://doi.org/10.1145/3571155

1 INTRODUCTION

In the seminal book "The Computer and the Brain" [241], John von Neumann discussed how the brain can be viewed as a computing machine. Since then, there have been multitude of works trying to perform brain-like functions with brain-like architectures. Neural networks, specifically Deep Learning, have powered the current era of ubiquitous artificial intelligence, demonstrating unprecedented success, even surpassing humans in several cognitive tasks [116, 214]. But at what cost?

While our fastest parallel computers have enabled deep learning, they are primarily limited by their ability to move data between the compute and memory, which is in stark contrast to the massively parallel, sparse, event-driven, distributed processing capabilities of the brain. Consequently, there is a substantial energy-efficiency gap between the brain and deep learning architectures. With increasing complexity of tasks in the era of in-sensor analytics and **Internet of Things (IoT)** and with the ever-growing size of networks deployed for such tasks, implementing and training such deep neural networks in edge devices with constrained power, energy and computational budgets have become a daunting task [257].

The underlying computational paradigm for Neuromorphic Computing is an emerging discipline of artificial neural networks that attempts to mimic neuronal and synaptic functionalities temporally and in a distributed fashion based on neuron "spikes" or firing events in the brain [47, 70]. Termed as **Spiking Neural Networks (SNNs)** [133], these networks lead to possibilities of sparse, event-driven neuronal computations and temporal encoding–a shift from standard deep learning networks, termed as **Analog Neural Networks (ANNs)**, which process and transmit logically analog information rather than all-or-nothing spikes.

Owing to the unique features of SNNs, there is a need to explore new algorithmic directions that are more amenable to its implicit recurrence, event-driven, and sparse nature of computing. SNNs, by their design, are dynamical recurrent systems; the internal state of the spiking neuron integrates temporal information and maintains a history of previous inputs. Training recurrent networks with binary signals exacerbates the issue of exploding and vanishing gradients. While the event-driven nature of SNNs offers a promising route for achieving lower energy and power consumption for intelligent hardware, it also poses a critical limitation on their learning capability. Integrating temporally encoded statistics of spiking neurons/synapses with standard gradient-descent-based learning algorithms (catered for ANNs that do not encode information in time) presents several challenges [123, 265]. It is difficult to train the layers of a deep SNN architecture globally in an end-to-end manner [181]. Bio-plausible unsupervised [52] and supervised [99] learning have been explored as a viable solution that allows localized learning, and has proven to be computationally more efficient than backpropagation-based algorithms. Although SNNs trained with unsupervised learning algorithms are not yet suitable for challenging cognitive tasks, they find applications in clustering and extracting low-level features from images for recognition. There has also been a significant thrust towards developing scalable gradient-based algorithms that can be adapted to

event-driven, sparse activity in SNNs [16, 166]. Overall, these algorithmic drives promises to scale up the performance of SNNs to levels currently offered by ANNs, while preserving the benefits of sparse event-based computations. Also, current algorithms do not take full advantage of the quintessential "time" parameter, future research can explore these opportunities along with implementing additional bio-plausible functions to further the performance of SNNs.

A parallel research thrust focuses on the development of efficient computing platforms for the evolving SNN algorithms and workloads. Hardware for SNN workloads derives certain motivation from the broad field of Neuromorphic Engineering. The concept of Neuromorphic Engineering was first proposed by Carver Mead in his seminal book [144, 145], where he explored the notion of analog circuits to mimic complex neuronal and synaptic functionalities in the brain. This was later demonstrated in groundbreaking work of implementing a Silicon Neuron [139] and then a Silicon Retina by Mahowald et al. [138]. Various neuromorphic chips have hence been implemented such as ROLLS [187], Dynap-SEL [154], Neurogrid [21], SpiNNaker [171] and so on. These implementations target emulation of the biophysics of neurons and synapses of the brain through CMOS circuits. SNN workloads tend to use simpler neuro-synaptic models, and can potentially take inspiration from aforementioned works that have delved deeper into emulating brain-like characteristics on chip. Besides neuro-synaptic models, another key component of SNN hardware is acceleration of the computations in SNN workloads. The requirements of such SNN accelerators have evolved significantly over the last few years, particularly toward designing large-scale systems effectively leveraging key features of SNN algorithms. The accelerators targeted toward executing standard ANN workloads more efficiently such as **General Purpose Graphics Processing Units (GPGPUs)** and **Tensor Processing Units (TPUs)** [98] are not designed to optimize processing across multiple timesteps. Neuromorphic hardware architectures draw inspiration from two basic principles of SNNs: (i) event-driven sparse computations and (ii) efficient and parallel matrix operations. While ANN accelerators are designed to efficiently perform matrix operations, they fail to leverage temporal sparsity in SNNs. The temporal sparsity has inspired various architectures such as *TrueNorth* [6] and *Loihi* [46], which deploy asynchronous computing systems to reduce compute and communication of data. The second feature is more generic to neural networks, and have overseen significant developments of domain-specific accelerators [5, 163] with intertwined memory and compute elements to overcome the von-Neumann bottleneck. While most large-scale SNN accelerators are based on CMOS technology, several **Non-Volatile Memory (NVM)** technologies have emerged as a promising candidate for building bio-mimetic devices [31, 37, 269]. Such devices can emulate neuronal [172, 206, 234] and synaptic functionalities [28, 84, 97, 111, 185, 224, 226] at a one-to-one level while simultaneously enabling a novel paradigm of "In-Memory" computing, i.e., *in situ* synaptic computations [262] for acceleration of SNN workloads [11, 216]. Thus, there is a need to co-design neuromorphic algorithms and the underlying computing architectures in a multi-disciplinary research [88].

This article outlines recent developments in the domain of Spiking Neural Networks under the umbrella of neuromorphic engineering in terms of driving research directions in algorithms as well as hardware. Whether we will be able to achieve the much eluded energy-efficiency in machine intelligence platforms is a question that is difficult to answer currently. However, we believe a rethinking of brain-inspired computing with a unified hardware-software perspective is essential to enable computationally efficient learning. Combining outlooks from varying fields—computational neuroscience, machine learning, materials, devices, circuits, and architectures—will help outline the challenges posed by these different outlooks and provide a future direction for intelligent platforms with power and energy-efficiency akin to the brain.

2 ALGORITHMS

In this section, we delve into the algorithmic foundations for SNNs. Significant developments in neuron modeling, input encoding, learning algorithms, and network architectures have shaped the progress of neuromorphic computing. Due to the representation of inputs with time domain information, considerable rethinking of the training algorithms, currently in place for ANNs, is required.

2.1 Neuron Models

Neuron models with varying degrees of bio-fidelity have been proposed to mimic the dynamics of biological neurons [69]. Some of the simpler and popular models are **integrate-and-fire (IF)**, **leaky-integrate-and-fire (LIF)**, and **spike response model (SRM)** that are widely used in deep SNNs for image-recognition tasks [52, 96, 199, 209, 213]. The more detailed and complex Hodgin-Huxley model considers the dynamics of different ion channels observed in biological neurons [83]. The IF/LIF and SRM neuron models can be derived from the Hodgin-Huxley model [69]. Although Hodgin-Huxley model is more biologically realistic, current optimization algorithms (such as ANN-to-SNN conversion and surrogate gradient-based learning) perform better with simpler IF/LIF neuron models. Most of the spiking neuron models have few things in common: They have an internal state that accumulates the input stimuli; the neuron generates an output (or fires) when the internal state crosses a threshold value; and sometimes the firing event is followed by a refractory time-period, during which the neuron is dormant and does not respond to input stimuli. Here, we discuss the prevalent IF/LIF model that shows competitive performance on complex tasks and the stochastic neuron model that does not have an internal state. We refer to Reference [69] for details on other spiking neuron models.

2.1.1 LIF/IF Neuron Model. The dynamics of the IF/LIF model is described by the differential equation

$$\tau \frac{du(t)}{dt} = -[u(t) - u_{rest}] + RI(t), \tag{1}$$

where u is the internal state known as the membrane potential, u_{rest} is the resting value of the membrane potential, $R(I)$ is the input resistance (current), and τ is the time constant [69]. The equation represents the behavior of the neuron when the membrane potential (u) is below the threshold potential (v). The membrane potential integrates the input current over time and the neuron fires when the potential crosses the threshold voltage. IF/LIF is a very simple model and does not reflect the overall complex dynamics of a biological neuron. However, its simplicity makes it attractive to optimize in deep learning frameworks. For LIF neuron, the integration is leaky and the membrane potential decays over time in the absence of input stimuli (Figure 1). The firing event is sometimes followed by a refractory period during which the membrane potential does not integrate the input current. This avoids excessive firing of a particular neuron and allows other neurons to participate in the learning [52]. The membrane potential is reset after firing and returns to its rest value after the refractory period (Figure 1).

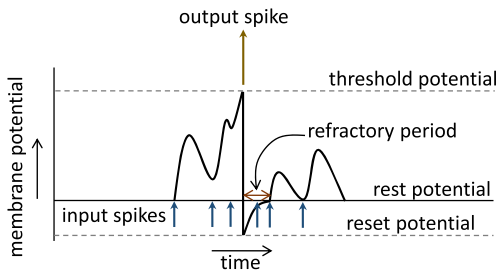

Fig. 1. Dynamics of a leaky-integrate-and-fire (LIF) model in response to input spikes.

The continuous time domain differential equation (Equation (1)) is solved to get an iterative update rule [254] where τ is mapped to λ through integration as

$$u_i^t = \lambda u_i^{t-1} + \sum_j w_{ij} o_j^t - v o_i^{t-1}, \tag{2}$$

$$o_i^{t-1} = \begin{cases} 1, & \text{if } u_i^{t-1} > v \\ 0, & \text{otherwise} \end{cases}, \tag{3}$$

where u is the membrane potential, subscript i and j represent the post- and pre-neuron, respectively, superscript t is the timestep, λ is a constant (<1) responsible for the leak in membrane potential, w is the weight connecting the pre- and post-neuron, o is the output spike, and v is the threshold potential. The second term in Equation (2) integrates the inputs and the last term resets the membrane potential after firing. The reset mechanism reduces the membrane potential by the threshold value instead of resetting it to the reset potential. This reduces information loss and leads to better performance in image classification tasks [77]. Most of the hardware implementations of LIF models employ the reset to ground, however, few of them provide the functionality of reset by subtraction [4].

2.1.2 Stochastic Neuron Model. The deterministic IF/LIF neuron model, discussed above, emits a spike as soon the membrane potential crosses the threshold voltage. If the membrane potential is below the threshold at any timestep, then the computation in next timestep begins with the previous membrane potential. In contrast, a probabilistic neuron model, fires stochastically and the probability of firing at a particular time is a non-linear function of the instantaneous magnitude of the weighted input [19, 206, 242]. The probability of firing of a post-neuron is defined as

$$P(o_i = 1) = \frac{1}{1 + e^{-\sum_j w_{ij} o_j}}, \tag{4}$$

where o_j is the spike input (1 or 0) from pre-neurons and w_{ij} is the synaptic weight connecting pre- and post-neuron. In the absence of any input activity and bias ($\sum_j o_j = 0$) the firing probability is 0.5. A network with stochastic neurons is evaluated over multiple iterations and the output of a layer is computed as the average number of spikes over all iterations. The average analog value can then be rate-coded (more on this in Section 2.2.1) as a Poisson spike train to act as input to the next layer. These stochastic models are difficult to implement in hardware.

2.2 Input Encoding

SNNs compute and communicate information through binary signals (spikes). Therefore, analog inputs such as image pixels or real numbers need to be encoded in binary signals. A single spike is a discrete event represented as "+1" or "−1" and an analog value is encoded in a set of spikes. The encoding mechanism determines the quantization or conversion error in representing the analog value with spikes. The analog value is represented by one or more spikes distributed over a time period. The popular coding methods are rate coding [52, 209], temporal coding [155], and explicitly training an encoding layer [193, 255]. The encoding methods are depicted in Figure 2.

2.2.1 Rate Coding. In rate coding, the information is represented in the mean firing rate of the neuron within a time period. The timing of the individual spikes has no relevance. In Poisson encoding, a type of rate coding, at each timestep the normalized analog value is compared with a random number. The neuron fires (output "1") if the analog value is greater, otherwise stays inactive (output "0"). The number of timesteps[1] determines the discretization error in the

[1]Wall-clock time for 1 "timestep" is dependent on the number of computations performed and the underlying hardware. In simulation, 1 timestep is the time taken to perform 1 forward pass.

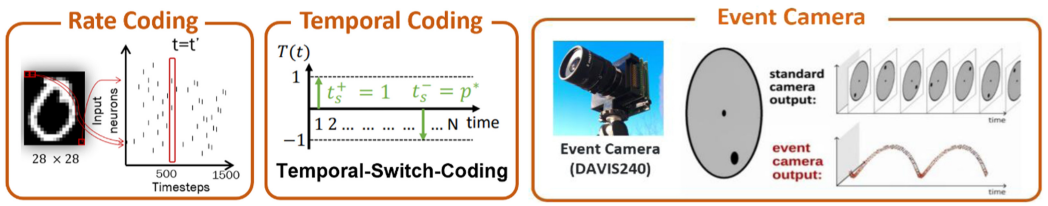

Fig. 2. Different input encoding methods for SNNs. In rate coding, the information is represented in the average number of spikes in a given duration. In Poisson rate coding, at every timestep (t') a random number $[0, 1]$ is generated for each neuron and compared with the corresponding pixel value; the neuron fires if the pixel value is greater than the random number. In Temporal Switch Coding, a form of temporal coding, the time difference between two spikes (t_s^+ and t_s^-) encodes the input data; p^* is proportional to the value being encoded. Event cameras capture the asynchronous changes in log intensity as discrete spikes over time [158].

representation of the analog value by spike-train. This leads to adopting a large number of timesteps for high accuracy at the expense of high inference latency [209]. Rate coding is inefficient due to minimal information content in each spike.

2.2.2 Temporal Coding. Unlike rate coding, in temporal coding, the timing of individual spikes is crucial, as the information is encoded in the timing instances. The Logarithmic Temporal Coding [267] encodes the analog value as a binary number with a predefined number of bits. The number of bits serves as a proxy for time and a spike is generated for each active bit in the binary number. For sparser representations, the spike is generated only for the most significant bit. Rank Order Coding [48] represents the information as the order of firing instances instead of using the precise timing of the spike. The input neurons encoding larger analog values fire earlier compared to neurons encoding smaller values. Time-to-First-Spike [156, 198], a form of Rank Order Coding, restricts each neuron to spike only once. The order or time of spike is inversely proportional to the analog value being encoded. In Temporal Switch Coding [76], the analog value is encoded using two spikes, and the time difference between the spikes is proportional to the encoded value. It achieves better energy-efficiency, since at most two memory accesses and two addition computations are performed for each synapse. Although temporal coding methods can encode information with less number of spikes, the lack of appropriate training algorithms for temporally coded SNNs results in sub-optimal performance compared to rate-coded SNNs [198].

2.2.3 Encoding Layer. Rate and temporal coding, discussed earlier, are fixed formula-based, non-parameterized coding methods. Alternatively, the input encoding can be made part of the training process and, therefore, the encoding function can have parameters that are trained with the input data. The encoding function is modelled as a neural network that receives the analog values and generates a spike train. In some cases, this network is as simple as one convolution layer [132, 193, 199]. The encoding layer is appended to the front of the SNN and trained end-to-end with the entire SNN. The encoding layer consists of IF/LIF neuron that integrates the weighted analog values and generates a spike-train. The input to the encoding layer at every timestep is the same analog value.

2.2.4 Event-based Sensors. All the above discussed encoding methods are designed with the aim to encode the original image frames captured by standard cameras into pixel-wise temporal spikes to construct the inputs for the SNN. However, since the original input had no time information to begin with, this spatio-temporal representation is not very well justified. Standard cameras that are typically used to capture image as well as video information fall victim to a variety of drawbacks for real-world and edge-applications. Their low and fixed sampling rate makes them

prone to motion blur when capturing high-speed motion. Their low dynamic range of operation renders them unable to capture meaningful information in challenging lighting conditions such as low light and **high dynamic range (HDR)** environments. In addition, sampling the entire image frame at regular intervals makes them capture redundant information over and over in nearly static scenes leading to a high power consumption. These shortcomings motivate the need for specialized sensing modalities for efficient operation in challenging real-world environments while being significantly energy-efficient.

Event-based sensors (such as DVS128 [129], DAVIS240 [29], Samsung DVS [217], etc.) aim to address these concerns by offering asynchronous sensing of change in visual information from the environment. Also known as bio-inspired silicon retinas, event cameras detect the log-intensity (I) changes at each pixel element asynchronously and independently and generate a spike event if the change exceeds a threshold (C):

$$\| \log(I_{t+1}) - \log(I_t) \| \geq C. \tag{5}$$

These cameras employ a threshold-variation-insensitive algorithm over the standard asynchronous sigma-delta modulation when handling the input intensity signal. This results in idle output for no change in input intensity. Since only the log intensity changes are monitored and recorded, a very low power consumption can be achieved. Due to the fundamentally different working principle compared to standard cameras, event-cameras provide exceptionally high temporal resolution ($10\mu s$ vs. $3ms$), high dynamic range ($120dB$ vs. $74dB$), and low power consumption ($\sim 10mW$ vs. $3W$). Event-cameras offer advantages compared to standard cameras in scenarios such as real-time human-machine interface systems, robotics, wearable electronics, or vision-based edge-devices in general, where efficient operation in challenging lighting conditions, low latency, and power consumption [130] are paramount. They also find applications in computer vision and robotics tasks such as object detection and tracking [151], gesture recognition [10], optical flow/depth, egomotion estimation [260, 270, 272], and so on.

Event-based encoding results in asynchronous and sparse spatio-temporal data containing both structural and temporal information in the form of a voxel. This type of input representation can be naturally handled by asynchronous event-driven models such as SNNs, as will be discussed in later sections. Nevertheless, most of the research using event cameras has been carried out either in conjunction with traditional computer vision methods or with ANNs. This requires constructing event frames in place of image frames (as with standard cameras) by accumulating events over a certain time interval and subsequently considering them equivalent to image frames thereafter. For example, works such as References [260, 270, 272] utilize these event frames as channels serving as input to an ANN. This leads to the loss of essential temporal information within the accumulation interval as well as the temporal ordering of individual frames. Although, they show promising potential when compared with approaches using standard cameras, they still do not completely utilize the fundamental benefits of event cameras. This is because the asynchronous and discrete nature of event camera data makes it incompatible to work with with traditional ANNs in their native form that rely on frame-based information. In addition, ANN-based methods are typically designed for pixel-based images following the photo-consistency assumption (color and brightness of an object remains the same over image sequences). Certain works such as References [22] and [23] explore using events directly for estimating visual-flow, however, they are limited in terms of scalability to more complex problems. In light of this, SNNs inspired by the biological neuron model, offer direct handling of event data providing asynchronous computations while also exploiting the rich spatio-temporal dynamics and inherent input sparsity. Moreover, using event data with SNNs eliminates the need for having any complicated spike encoding stage required for

ANN-based methods. The working principle of spiking neurons naturally offers compatibility for event-based asynchronous processing distributed across SNN layers and, combined with computation on specialized neuromorphic hardware such as IBM's TrueNorth [6] or Intel's Loihi [46], provides high energy-efficiency. Section 3.4 discusses some works combining event cameras with SNNs along with the benefits as well as challenges associated with them.

2.3 Feedforward Networks

Feedforward networks comprise multiple convolutional and/or fully connected layers, where information flows from the input to the output layer, and connections between the neurons do not form a cycle. In this section, we discuss various unsupervised, supervised, and bio-plausible local learning rules for feedforward SNNs.

2.3.1 Unsupervised Learning. Unsupervised learning refers to algorithms that identify patterns from unlabelled data. The data consists of inputs (image, audio, text, etc.) and no corresponding label or targets. Unsupervised learning is desirable to find clusters in raw and unknown data.

Spike Timing Dependent Plasticity: Spike Timing Dependent Plasticity **(STDP)** [25, 69, 101], an unsupervised learning technique, is a biologically plausible mechanism for synaptic learning in SNNs. STDP-based learning rules [218] modify the weight of a synapse interconnecting a pair of pre- and post-synaptic neurons based on the degree of correlation between the respective spike times as specified by

$$\Delta w = \begin{cases} A_+ \, e^{-\frac{\Delta t}{\tau_+}}, & \text{if } \Delta t = t_{post} - t_{pre} > 0 \\ -A_- \, e^{\frac{\Delta t}{\tau_-}}, & \text{if } \Delta t = t_{post} - t_{pre} < 0 \end{cases}, \tag{6}$$

where t_{pre} and t_{post} are the time instant of a pair of pre- and post-spikes, A_+, A_-, τ_+, and τ_- are the learning rates and time constants governing the change, Δw, in the synaptic weight. Another variant of the STDP rule performs both potentiation[2] and depression based on the positive values of Δt (Figure 3(d)) [52, 192]. The weight update is computed as

$$\Delta w = \eta \times \left[e^{\left(\frac{t_{pre} - t_{post}}{\tau}\right)} - offset \right] \times [w_{max} - w]^{\mu}, \tag{7}$$

where Δw is the change in weight, η is the learning rate, t_{pre} and t_{post} are the time instant of pre- and post-synaptic spikes, τ is the time constant, $offset$ is a constant used for depression, w_{max} is the maximum constrained imposed on the synaptic weight, w is the previous weight value, μ is a constant that governs the exponential dependence on previous weight value. The weight update is positive (potentiation) if the post-neuron spikes immediately after the pre-neuron and negative (depression) if the spikes are far apart (Figure 3(d)). There are many other variants of STDP curves in response to spike pair stimulation, including spike triplets and quadruplets [73]. STDP modulates the strength of each synapse independently; while this is very powerful, it also introduces stability problems. Therefore, mechanisms to maintain an appropriate level of distributed activity throughout the network are explored in Reference [3]. In STDP, effective synapses are strengthened and ineffective synapses are weakened, which creates a positive feedback loop and leads to stability issues.

In addition to non-volatile STDP, which is non-adaptive to different learning scenarios (due to the absence of any forgetting mechanism in absence of a spike stimulus), synaptic learning mechanisms such as **Short-Term Plasticity (STP)** and **Long-Term Potentiation (LTP)** [140, 274]

[2]In neuroscience, the increase in synaptic strength (positive change in weight) is called potentiation, and the reverse (negative change in weight) is termed depression.

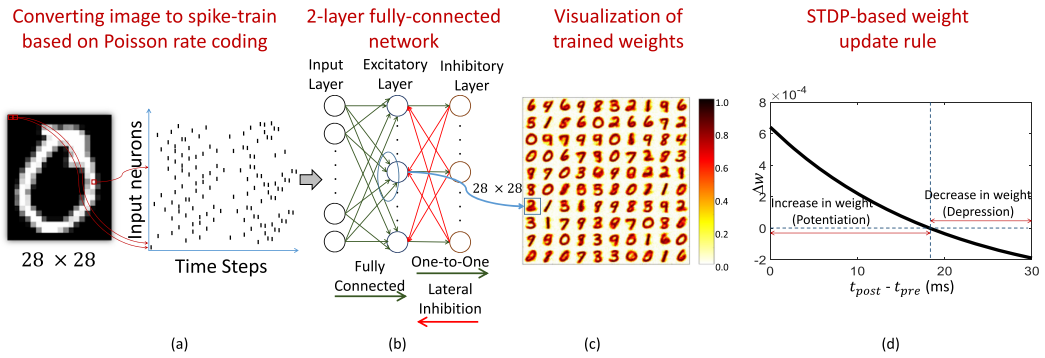

Fig. 3. Unsupervised learning in SNN based on STDP learning rule; (a) the input image is converted to spike-train, where the probability of spiking is proportional to the pixel intensity; (b) a 2-layer fully connected network with inhibitory connections receives spike inputs and the weights are trained based on STDP learning rule. The input-to-excitatory layer weights are trained, whereas the excitatory-to-inhibitory and inhibitory-to-excitatory layer connections are fixed before training [52]; (c) visualization of input-to-excitatory layer weights show that each excitatory neuron learns a unique class; (d) STDP learning rule (Equation (7)) based on temporal correlation between pre- and post-synaptic spikes ($\eta = 0.002, \tau = 20ms, offset = 0.4, w_{max} = 1, w = 0.5, \mu = 0.9$).

bearing resemblance to the concepts of Short-Term and Long-Term Memory [13, 113] have also been explored for online learning that can adapt synaptic weights to dynamically changing environments. Catastrophic forgetting, a phenomena observed in continual learning where knowledge about old tasks is lost when new tasks are learned, is a prevalent issue in neural networks, and SNNs trained with STDP also suffer from it [173]. An adaptive weight decay mechanism [173] that gradually forgets less important weights and learns new information in its place can effectively learn new data and avoid catastrophic forgetting.

SNNs are being actively explored for unsupervised pattern recognition due to their ability to learn input representations using STDP-based localized learning rules. The unsupervised feature learning capability of SNNs was initially demonstrated using two-layer SNN consisting of an input layer fully connected to neurons in the excitatory (or output) layer followed by an inhibitory layer (Figure 3) [52, 74]. However, the performance of such two-layer SNNs is still limited to simple tasks like handwritten digit recognition. Convolutional SNNs, similar in architecture to their deep learning counterparts [107], were proposed to address the limited scalability of two-layer SNNs [63, 103, 121, 141, 221, 223, 230, 232]. The convolutional layers can be trained using STDP for learning hierarchical input representations in an unsupervised manner, which are then fed to the classifiers trained using supervised learning rules for inference. While STDP-trained convolutional SNNs yield improvement over fully connected two-layered SNNs, the accuracy is still lower than state-of-the-art performance on popular benchmark datasets. The challenge for STDP-trained convolutional SNNs is two-fold. First, it remains to be seen how deep these SNNs can be scaled, since the current networks are limited to few convolutional layers. Second, it is inconclusive if STDP alone can enable the deeper layers to learn complex input representations. STDP is powerful at clustering and extracting low-level features from images but fails to generalize on composing high-level features and, therefore, two or more layers of STDP learning do not provide much benefit [63]. In hierarchical learning with convolutional layers, it is necessary to combine learned features into high-order features that can perform recognition. Some algorithms that can supplement the feature extraction of STDP can address these shortcomings. However, if these grand challenges are

met, then STDP would enable a new generation of low-cost (area/power/resource requirement), local, and unsupervised learning framework, as opposed to their non-spiking counterparts (ANNs) relying on global and supervised learning techniques. The advantages are multitude; unsupervised learning enables the network to adapt to changing environments while local learning assists in implementation of low-power learning circuit primitives. Local learning would remove the need for a global error distribution that is essential for supervised learning, which, in turn, requires hardware expensive circuitry.

Stochastic STDP: The STDP algorithm, described earlier, works well for shallow networks (2–3 layers) with full precision weights. However, networks with low precision, for example binary weights, require a probabilistic learning rule for efficient training and to avoid rapid switching of weights between allowed levels. The difference between the spike times of post- and pre-neuron ($t_{post} - t_{pre}$) is mapped to the switching probability of the connecting binary weight (Figure 4) [222]. The synaptic weight switches from low to high state (Hebbian potentiation) with a constant probability if the time difference is positive and below a certain threshold. If the time difference is negative and above a fixed threshold, then the weight switches from high to low state (Hebbian depression) with a constant probability. Additionally, if the time difference is positive and above a certain value, then the synapse is depressed (anti-Hebbian depression) due to low causality. The authors in Reference [222] mention that anti-Hebbian learning enables the synapses to unlearn features lying outside the receptive field like noisy background in images. The stochastic STDP learning rule enables training of SNNs with binary weights [188] and can achieve similar accuracy as full-precision SNNs trained with STDP [52] but with lower memory requirements [222]. Additionally, Reference [119] proposes a semi-supervised training mechanism with STDP. The network is initialized with pre-trained STDP weights trained in an unsupervised manner. Next, supervised gradient-based learning is employed to fine-tune the weights and improve the accuracy with faster convergence. Even though STDP-based local learning rules are not yet applicable for large-scale problems (like ImageNet), they perform relatively well for energy-efficient clustering tasks.

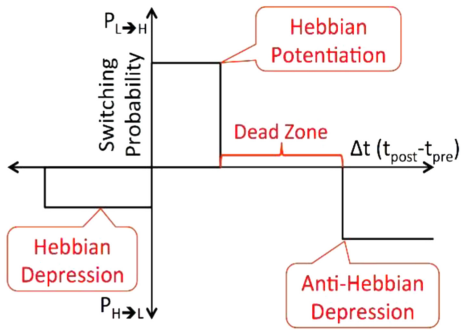

Fig. 4. Stochastic STDP learning rule for binary synapse where the probability of the synapse switching from high to low state (H→L) is a function of the difference between the spike times of post- and pre-neuron ($t_{post} - t_{pre}$).

2.3.2 Supervised Learning. Unlike unsupervised learning that does not require labelled examples for training, supervised learning derives its strength from large corpus of labelled examples. In supervised learning, the network receives an input (image, text, or audio) and produces a prediction score for all possible labels that the input can belong to, i.e., number of classes in the datasets. The prediction is compared with the true label (one-hot vector with "1" for the correct class and "0" for everything else) to determine the error/loss, and the network parameters (weights, bias, etc.) are updated based on the gradients of the loss function with respect to the parameters. Generally, in ANNs the supervised methods perform extremely well if a large number of labelled samples are available. However, direct implementation of gradient-based methods in SNNs is challenging because of the discontinuous and non-differentiable nature of the spiking neuron.

To circumvent this problem, researchers proposed methods to convert a trained ANN to an SNN for inference [34, 53, 199, 209]. In addition, surrogate gradients that approximate the discontinuous derivative as a continuous function are used to train SNNs with end-to-end backpropagation [16, 166, 213]. Recently, a combination of ANN-to-SNN conversion and surrogate gradient-based learning was employed to train deep SNNs for image-recognition tasks [193, 194]. Additionally, bio-plausible methods that learn from local information present at a synapse are also proposed for efficient training [165, 202].

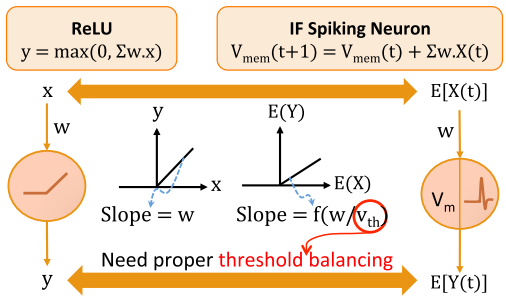

Fig. 5. Mapping a ReLU neuron to IF neuron during ANN-to-SNN conversion.

ANN-to-SNN conversion: Although SNNs trained with local learning rules are hardware-friendly and energy-efficient, they suffer from sub-optimal accuracy on challenging datasets [63, 174]; whereas, SNNs trained with conversion frameworks achieve accuracy similar to that of ANNs [34, 53, 85, 199, 209]. In ANN-to-SNN conversion, first an ANN with ReLU neurons is trained with state-of-the-art supervised algorithms with some restrictions (no bias, average pooling, no batch normalization), although some works have shown that some of the restrictions can be relaxed [199]. Next, an SNN with IF neurons and iso-architecture as ANN is initialized with the weights of the trained ANN. The underlying principle of this process is that a ReLU neuron can be mapped to an IF neuron with minimum conversion loss. The mapping is performed by adjusting the firing threshold of the IF neuron so its average firing rate is similar to the activation of the ReLU neuron (Figure 5). The major bottleneck of this method is to determine the firing threshold of the IF neurons that can balance the accuracy-latency tradeoff. Generally, the threshold is computed as the maximum pre-activation of the IF neuron resulting in high inference accuracy at the cost of high inference latency (~1,000 timesteps) [209]. In recent work, the authors showed that instead of using the maximum pre-activation, a certain percentile of the pre-activation distribution reduces the inference latency (60–80 timesteps) with minimal accuracy drop [132]. Researchers in References [199, 209] have demonstrated deep SNNs trained with conversion methods on standard deep learning architectures such as VGG [215], ResNet [79], and Inception [228], exhibiting state-of-the-art performances on complex datasets like ImageNet [200]. A "soft reset" mechanism (Section 2.1) that retains the residual membrane potential at spike-events was proposed to further reduce the conversion loss in ANN-to-SNN conversion frameworks [77]. Deep architectures with sparse, event-driven computations can potentially yield energy reductions compared to their ANN counterparts, since the neuron spiking sparsity increases drastically with network depth [209]. While conversion frameworks establish the effectiveness of spike-based inference and show as proof-of-concept that SNNs bear comparable computational power as their ANN counterparts, it has a major drawback: the absence of the timing information. The quintessential parameter "time" is not utilized in the conversion process, which leads to higher inference latency. The spike-based gradient descent methods, described next, perform credit assignment with **backpropagation through time (BPTT)** and achieve lower inference latency by incorporating temporal information.

Spike-based backpropagation: ANNs are primarily trained with gradient-descent-based methods that update the network parameters (weights, biases, etc.) based on the partial derivative of the loss function with respect to the parameter ($\partial L/\partial W$). In SNNs, the spike activation function

of the IF/LIF neuron does not have a continuous derivative. The derivative of the spike function (Dirac-delta) is not defined at the time of spike and "0" otherwise. To overcome this issue, surrogate gradients or pseudo-derivatives (Figure 6) of the spike function are proposed that approximate the real gradient as a continuous function [16, 166, 213]. Since SNNs compute over multiple timesteps in the forward pass, the gradients are computed by unrolling the network in time and performing BPTT. Several recent works have employed spike-based gradient-descent learning on SNNs to perform various classification tasks [16, 96, 119, 120, 123, 155, 165, 166, 254]. SNNs trained with spike-based gradient-descent can achieve lower inference latency compared to ANN-to-SNN converted networks (Figure 7). The primary reason is the integration of temporal information in training that is lacking in the ANN-to-SNN conversion process. Thus, gradient descent-based training achieves better latency but requires more training effort (both computations and memory). A single feed-forward pass in ANN corresponds to multiple forward passes in SNN that is proportional to the number of timesteps. Also, the backward pass requires the gradients to be integrated over the total number of timesteps, which increases the computation and memory complexity. The multiple-iteration training effort with exploding memory requirement has limited the applicability of spike-based backpropagation methods to small datasets (such as MNIST and CIFAR10) on simple few-layered convolutional architectures.

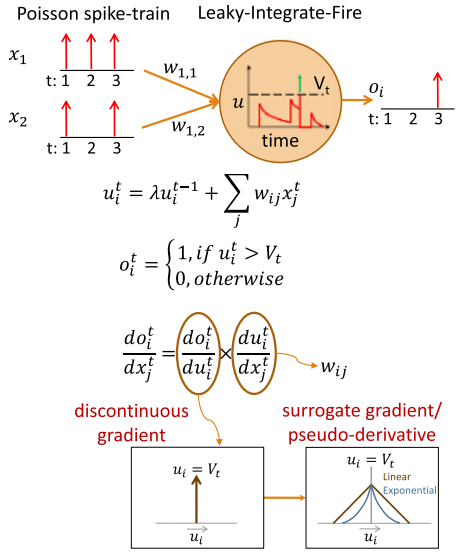

Fig. 6. Discontinuous gradient of spike function approximated as a continuous pseudo derivative. x_1 and x_2 represents the Poisson spike-train of two input neurons connected with weights $w_{1,1}$ and $w_{1,2}$ to a LIF neuron, u is the membrane potential, λ is the leak coefficient, and o is spike output of the LIF neuron. The derivative of the LIF output o with respect to its input x is discontinuous and is approximated by a pseudo-derivative to enable credit assignment through backpropagation.

Researchers in Reference [16] showed that damping the surrogate gradient enhanced the performance of gradient descent during larger time spans. Furthermore, Reference [166] provides a comparative study of employing various surrogate gradients as an approximation for the discontinuous real gradients for performing gradient descent in SNNs.

Hybrid learning: The efficacy of the gradient descent algorithms have thus far been validated on MNIST [117] and CIFAR datasets [106] for SNNs with few layers. The scalability of the supervised algorithms and their ability to achieve training convergence for much deeper SNNs remain a challenge (since this requires error backpropagation with neurons generating outputs as a temporal spike sequence). Additionally, the memory requirement for performing gradient descent in SNNs is higher compared to the conversion frameworks, because the number of computations linearly increase with time. To address the scalability of gradient descent methods, Reference [194] proposed a hybrid training mechanism (Figure 8) that solves both the issue of high latency (conversion frameworks) and high training cost (gradient descent in SNN). They first perform an ANN-to-SNN conversion using the framework from Reference [209] followed by a gradient descent-based training in SNN with surrogate gradient (Algorithm 1). The conversion framework acts as a good

ALGORITHM 1: Supervised learning in SNNs: ANN-to-SNN conversion, and spike-based backpropagation

ANN training
Input: Dataset (D), ANN model (N_a), initial weights (W_a)
while *stopping criterion not met* **do**
  sample mini-batch of input (X) - target (Y) pairs from D $\hat{Y} = N_a(X)$ // Forward propagation
  $Loss = CrossEntropy(Y, \hat{Y})$
  $W_a \leftarrow W_a - \epsilon \frac{dLoss}{dW_a}$ // Weight update
end

ANN-to-SNN conversion
Input: Trained ANN weights (W_a), mini-batch input (X) - target (Y) pairs from D, SNN model (N_s), Timesteps (T)
// Initialize SNN weights with trained ANN weights
$W_s \leftarrow W_a$
V: threshold voltage
// compute the threshold for all layers sequentially
for l ***in*** N_s **do**
  for $t = 1$ **to** T **do**
    // pre-activation of layer l
    $A_l = N_s(X)$
    // K is generally chosen between 90–100
    if K^{th} *percentile of* $A_l > V_l$ **then**
      $V_l = K^{th}$ percentile of A_l
    end
  end
end

Spike-based backpropagation on converted SNN
Input: Dataset (D), Converted SNN (N_s), SNN weights (W_s)
while *stopping criterion not met* **do**
  sample mini-batch of input (X) - target (Y) pairs from D, U: membrane potential, O: spike output, λ: membrane leak
  for $t = 1$ **to** T **do**
    $O_0 = X$ // direct input encoding
    for $l = 1$ **to** $L-1$ **do**
      // accumulate the output of previous layer in U, soft reset when spike occurs
      $U_l^t = \lambda_l U_l^{t-1} + W_{s_l} O_{l-1}^t - V_l O_l^{t-1}$
      // generate spike if U exceeds V
      if $U_l > V_l$ **then**
        $O_l = 1$
      end
    end
    // only accumulation in the final layer
    $U_L^t = U_L^{t-1} + W_{s_L} O_{L-1}^t$
  end
  $Loss = CrossEntropy(Y, U_L^T)$
  $W_a \leftarrow W_a - \epsilon \frac{dLoss}{dW_a}$ // Weight update
  $V \leftarrow V - \epsilon \frac{dLoss}{dV}$ // Threshold update
  $\lambda \leftarrow \lambda - \epsilon \frac{dLoss}{d\lambda}$ // Leak update
end

initialization and the gradient descent converges to a good setting within few epochs. The inference latency and accuracy on image classification tasks can be improved by training the leak and threshold along with the weights of the network [193].

Bio-plausible local learning: Spike-based gradient methods, described earlier, achieve good accuracy, but the backpropagation of global loss from output to input layer results in significant memory overhead. However, STDP-based unsupervised learning rules (Section 2.3.1) are compute- and memory-efficient but do not achieve competitive accuracy. In addition, local learning rules are more compatible with event-driven neuromorphic hardware (more on this in Section 4). Thus, researchers have explored gradient-based methods that do not require end-to-end backpropagation and perform weight updates based on local information. Generally, the global error (E) is the function of the output of the final layer (O_L) and the target (Y)

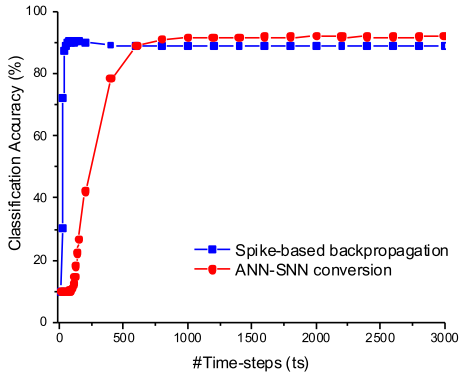

Fig. 7. Comparing classification accuracy vs. timesteps for SNNs trained with ANN-to-SNN conversion and spike-based backpropagation on VGG9 for CIFAR10 dataset [120]. For ANN-to-SNN conversion the threshold is computed as the maximum pre-activation of that layer. The timesteps for ANN-to-SNN conversion can be reduced by setting the threshold as a certain percentile of the pre-activation values [132, 193, 199].

$$E = f(O_L, Y). \tag{8}$$

To update the weights of a hidden layer (W_l) the gradient is computed as

$$\frac{\partial E}{\partial W_l} = \frac{\partial E}{\partial O_l}\frac{\partial O_l}{\partial U_l}\frac{\partial U_l}{\partial W_l}, \tag{9}$$

where U_l is the membrane potential. The term $\partial E/\partial O_l$ is the backpropagated error and is computed from all downstream synaptic weights; the other two terms are based on local information. The works based on bio-plausible local learning eliminate the dependence on this backpropagated term to perform weight updates. Hebbian three-factor learning rules [66] describe a method to update parameters locally based on pre-synaptic activity, post-synaptic variables, and neuromodulators. In the context of SNNs, neuromodulators are replaced with a local error signal [17, 99, 157, 264]. These learning rules can be implemented in hardware by designing eligibility traces for each synapse [49]. *DECOLLE*, an SNN equipped with local error functions to perform deep continuous local learning, is one such example [99]. A random readout layer is attached to each layer of the network, and an auxiliary cost function is defined over the readout. The random readout is obtained by multiplying the activations with a random and fixed matrix. Instead of minimizing a global objective function, the training process minimizes many local cost functions [157]. The weight update in each layer depends only on the information available locally, and thus all layers in the network can be trained in parallel.

2.4 Recurrent Networks

In feedforward networks, discussed earlier, a neuron is always connected to a different neuron and does not have any self-connection or loops. Additionally, there are no connections between neurons in the same layer; the neurons in one layer are connected to the neurons in other layers. In

ALGORITHM 2: Training in Spike-FlowNet (adapted from Reference [118])

Input: Spike inputs (*inputs*), #spike-frames in former/latter groups (N), #SNN layers (L_S), #ANN layers (L_A), SNN/ANN output (o/o_A), neuronal membrane potential (V), neuronal firing threshold (V_{th}), ANN non-linear activation (h)

Initialize: $V^l[n] = 0,\ \forall l = 1, \ldots, L_S$

// Forward-pass in SNN-block

for $n \leftarrow 1$ ***to*** N **do**

  $o^1[n] = inputs[n]$

  for $l \leftarrow 2$ ***to*** $L_S - 1$ **do**

    // weighted spike-inputs are integrated into V

    $V^l[n] = V^l[n-1] + w^l o^{l-1}[n]$

    // check if V exceeds V_{th}

    if $V^l[n] > V_{th}$ **then**

      // emit a spike and reset V

      $o^l[n] = 1$

      $V^l[n] = 0$

  // accumulate output at the final SNN layer

  $o_A^{L_S} = V^{L_S}[n] = V^{L_S}[n-1] + w^{L_S} o^{L_S - 1}[n]$

// Forward-pass in ANN-block

for $l \leftarrow L_S + 1$ ***to*** $L_S + L_A$ **do**

  $o_A^l = h(w^l o_A^{l-1})$

// Compute loss: photometric + smoothness

$\mathcal{L} = \text{photo}(o_A^{(L_S+L_A)}) + \text{smooth}(o_A^{(L_S+L_A)})$

// Backward-pass in ANN-blocks

for $l \leftarrow L_S + L_A$ **to** L_S **do**

  $\Delta w^l = \frac{\partial \mathcal{L}}{\partial o_A^l} \frac{\partial o_A^l}{\partial w^l}$

// Backward-pass in SNN-blocks

for $n \leftarrow N$ ***to*** 1 **do**

  for $l \leftarrow L_S - 1$ ***to*** 1 **do**

    // evaluate partial derivatives of loss w.r.t. w_S *by unrolling the SNN over time*

    $\Delta w^l[n] = \frac{\partial \mathcal{L}}{\partial o^l[n]} \frac{\partial o^l[n]}{\partial V^l[n]} \frac{\partial V^l[n]}{\partial w^l[n]}$

contrast, recurrent networks have feedback connections where the output of the neuron is routed back as input with a time delay. Hence, the output is a function of current input and the previous state of the neuron. SNNs implicitly have this relation, as the membrane potential depends on the input and the potential at previous timestep (Equation (2)). Thus, SNNs have implicit recurrence through its internal state [166, 252], as shown in Figure 9. Note, however, in this section, we discuss **recurrent networks of spiking neurons (RSNNs)** that have explicit recurrent connections on top of the intrinsic recurrent dynamics. In ANNs, recurrent connections are particularly important to learn and generate sequences with long-range structure such as text, video, or audio data. RNNs contain feedback loops in their connectivity structure, which allows the network to remember prior computations or history of the input [57, 116, 179, 227]. The computation in RNNs occur by unrolling the network over discrete timesteps. Once unrolled, RNNs can be viewed as deep feedforward networks with shared weights capable of establishing long-range temporal

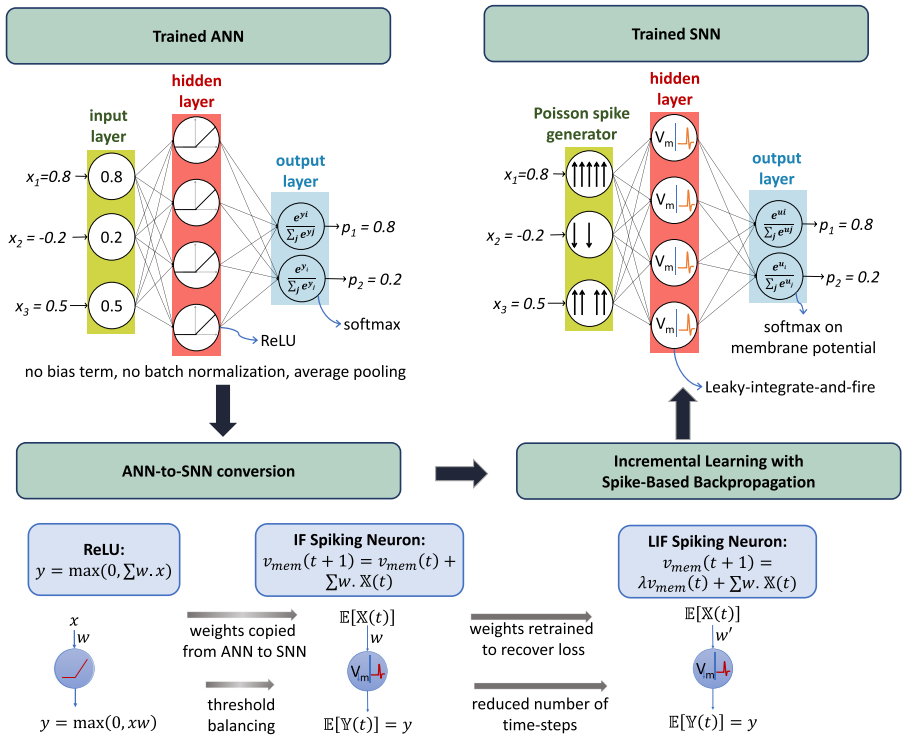

Fig. 8. Hybrid learning algorithm combining ANN-to-SNN conversion and spike-based backpropagation [194]. A trained ANN is converted to SNN by replacing ReLU neurons with IF neurons and threshold balancing, where the threshold for each layer is determined sequentially. The converted SNN is incrementally trained for few epochs (~20) with spike-based gradient backpropagation to reduce the inference latency. The trained SNN demonstrates better accuracy as well as lower inference latency (100–200 timesteps) on image classification tasks. The inference latency can be reduced (5–10 timesteps) by replacing the Poisson spike generator with an encoding layer that accepts analog values and outputs spike-trains [193].

dependencies. However, training RNNs has proven to be very difficult than their deep feedforward counterparts due to exploding or vanishing gradients that deteriorate the overall learning [20, 81]. Today, **Recurrent Neural Networks (RNNs)**, specifically **Long Short-term Memory (LSTM)** networks [82], are widely used to process sequential inputs such as language or speech.

The biological brain is known to be composed of a network of several millions of neurons connected via billions of recurrent synaptic links. These neuronal links are not randomly determined but are optimized for specific tasks and have developed through long-term evolution. RSNNs attempt to realize this neuronal model of the brain. This includes exploring connectivity in the model as well as learning the synaptic weights. RSNNs aim to offer high energy-efficiency compared to RNNs due to the computations being sparse and discrete in nature in the form of spikes. However, the research in this field has been highly limited due to the challenges associated with training. RSNNs generally have sub-optimal performance compared to equivalently trained RNNs, thereby limiting their application to only simplistic tasks.

Research on developing algorithms to enable bio-plausible learning in RSNNs has been an open problem for quite a few years with little success. There have been several works over the past years that aim to train RSNN models in a bio-plausible manner [7, 71, 148, 235]. These aim at learning the non-linear dynamics of RSNNs through feedback-based local learning rules. Authors

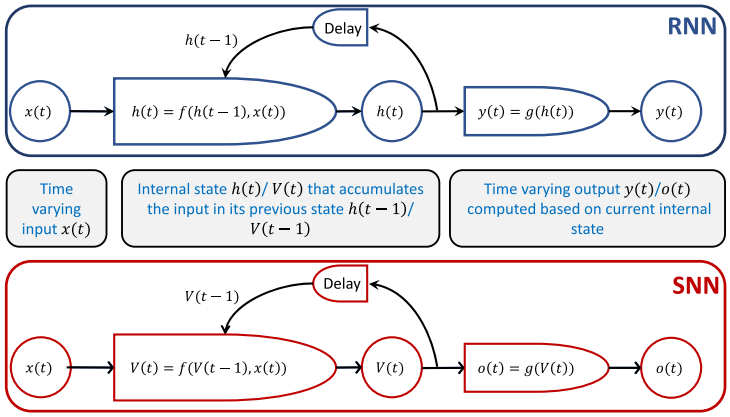

Fig. 9. Implicit recurrence in SNNs is similar to the recurrence relation in traditional RNNs. In RNNs, the hidden state $h(t)$ retains the history of all previous inputs through explicit feedback connections, whereas, in SNNs, the equivalent membrane potential $V(t)$ acts as a memory of past inputs. The output in RNNs is a non-linear function of the hidden state, and in SNNs, the neurons fire based on their membrane potential. Note, however, the inputs and outputs in SNNs are binary, whereas, in RNNs, they are continuous values.

in Reference [71] present **FOLLOW (Feedback-based Online Local Learning Of Weights)**, a local-learning algorithm in which the weights changes depend on the presynaptic activity and the projected error on the postsynaptic neuron. Reference [235] proposes to incorporate a third factor of local dendritic potential besides pre- and postsynaptic activity to modulated plasiticity. They utilize a functional rule seeking to minimize the descrepancies between somatic firings and the local dendritic potential. However, Reference [148] introduces a learning method called **Deep Feedback Control (DFC)**, which uses a feedback controller to drive a neural network towards a desired target output while its control signal is used for credit assignment. Researchers in Reference [51] explore using appropriate "firing-rate" models to train RSNNs for generating complex temporal outputs by utilizing continuous-variable networks to identify training targets. In addition, authors in References [100, 149, 150] explore event-based solutions related to neuro-circuit design for visual-motion perception and scene understanding.

In contrast to surrogate gradient-based BPTT, authors in Reference [18] propose "e-prop," a biologically realistic method based on minimizing a spike-dependent loss function (E) that measures the difference between the actual neuron output and a target output. It is shown that the loss-derivative with respect to the synaptic weights can be represented as the sum of products over the timesteps of RSNN computation with the help of a suitable pseudo-derivative offering an adequately powerful function to enable learning in RSNN models. The results show that "e-prop" can learn nearly as well as BPTT-based methods for tasks such as phenome recognition and reinforcement learning on Atari games compared with A3C [152] algorithm.

2.4.1 Spiking LSTMs. Like traditional LSTMs, spiking LSTM, too, finds applications in sequence modelling tasks such as natural language and speech processing, time-series predictions, and so on. However, training these spiking LSTMs again suffers drawbacks associated with SNNs, as discussed previously. To mitigate these and enable training, ingenious methods need to be developed. Authors in Reference [131] explore approximate loss functions to compute gradients and enable **backpropagation through time (BPTT)** in spiking LSTMs. They evaluate the performance on sequential versions of MNIST [117] and extended MNIST datasets while providing comparison with other feedforward SNNs. The authors in Reference [182], however, propose a hybrid analog

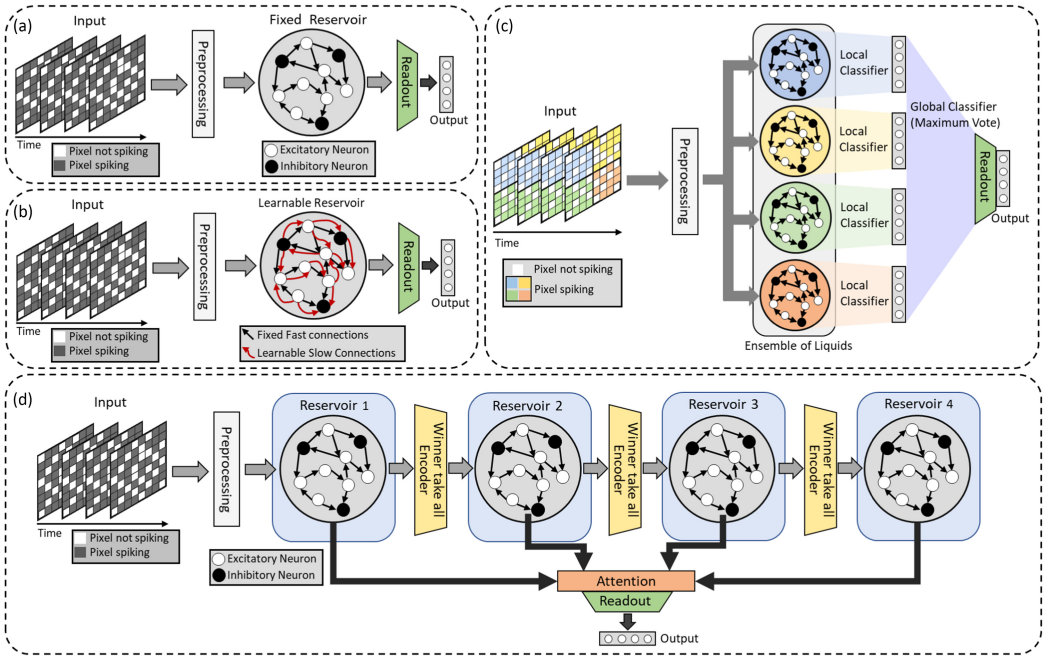

Fig. 10. Algorithmic and architectural optimization approaches for improving learning in liquid state machines (LSMs). (a) Vanilla LSM taking spike stream corresponding to each pixel in the image frame as input and predicting outputs at the readout layer trained using backpropagation. (b) Learning alternate liquid connections [175] to improve performance. Fast connections (having a high decay rate) are fixed, while the slow connections (having a low decay rate) are learned using STDP-based local learning rules. (c) Ensemble of smaller liquids with local classifiers followed by a global classifier for predicting final output leading to better performance at smaller overall liquid size [220]. (d) Deep Hierarchical LSM with attention modulated readout. Each hidden reservoir captures different sets of features that are selectively analyzed at the attention layer [219].

and spiking LSTM where the compute-intensive parts of LSTM are converted to SNN for better energy-efficiency on edge devices. The results showcase significantly lower energy consumption while having a negligible drop in performance for sequential image classification on MNIST [117] dataset and sequence-to-sequence translation on the IWSLT14 [36] dataset.

2.4.2 Liquid State Machines. **Liquid state machines (LSM)** have been explored as lightweight architectures to handle spatio-temporal inputs in a bio-plausible manner [134, 136]. LSM is an RSNN consisting of a sparsely connected reservoir (liquid) of excitatory and inhibitory spiking neurons. The synaptic connections and their weights are randomly initialized and fixed *a priori*. This leads to a simplistic lightweight structure while still inherently capturing the spatio-temporal input information. The liquid essentially projects the inputs to a higher dimensional space while also retaining temporal information through recurrent connections. Given a large enough reservoir with random and sparse interconnects, the high-dimensional representation generated can be linearly classified using a fully connected readout layer. The work by Maass and Markram in 2004 [135] investigates on the computational power of LSMs for real-time computing. Figure 10(a) shows a vanilla LSM consisting of a reservoir of spiking neurons (excitatory and inhibitory) with random connections. Several works have successfully employed LSMs for a variety of applications ranging from gesture recognition [175], video activity recognition [219], reinforcement learning [124, 183],

and so on, with low compute costs. The major challenge with LSM lies in its inability to scale well for real-life complex computing tasks without drastically increasing the reservoir size. Various efforts have been made towards improving the learning capability of LSMs without increasing their size significantly. One approach involves exploring mechanisms for training the liquid connections to improve application accuracy at the cost of added complexity. Authors in Reference [258] explore using heterogeneous neurons with different behaviors and degree of excitability in the liquid to aid learning. Reference [175] employs a Driven/Autonomous model approach [2] coupled with **Recursive Least Squares (RLS)** rule and FORCE training [167] to train the liquid connections. This approach consists of two different types of synaptic connections, namely, fixed fast connections (τ_{fast}) and learnable slow connections ($\tau_{slow} = 10 * \tau_{fast}$. This is shown in Figure 10(b). Other efforts focus on optimizing the network architecture itself. Authors in Reference [220] propose to adopt a "divide and learn" strategy by utilizing multiple small liquids, each learning characteristic patterns corresponding to a segment of input patterns. In addition, the input-to-ensemble connections are trained using STDP-based learning rules. The intermediate outputs from the ensemble of liquids are combined using a global classifier to generate the final output. This approach is shown in Figure 10(c). However, authors in Reference [219] propose an architecture involving multiple layers of liquids to form a deep hierarchical LSM. The hidden layers (liquids) are connected using spiking winner-take-all encoders that extract and propagate the temporal features. The representations generated by the different hidden layers (liquids) are condensed using an attention function before final classification at the readout. Figure 10(d) demonstrates this method. These approaches offer competitive accuracy and faster training time compared to a large single liquid. Similar to these, Reference [162] presents an efficient partitioning method for hierarchical mapping of large SNNs on reconfigurable neuromorphic hardware.

Recurrent networks of spiking neurons in the form of spiking LSTMs and LSMs thus show promise for realizing energy-efficient implementations for real-world applications on resource-constrained edge-devices. However, advancement in this field is not well paced when compared to ANNs due to their limited learning ability.

2.5 Neuromorphic APIs and Libraries

Neuromorphic systems involving different spiking neuron models and architectures demand a paradigm shift in the standard softwares and libraries that are originally optimized for ANNs, such as Pytorch, Tensorflow, Caffe, and so on. This is due to the fact that the asynchronous event-driven processing required by SNN architectures can not be directly realized using the above software APIs on traditional **Graphical Processing Units (GPUs)**. This not only requires a complete rework at the hardware end to develop neuromorphic hardware such as Intel's Loihi [46], IBM's TrueNorth [6], SpiNNaker [171], and so on, as will be discussed in more detail in Section 4, but also requires developing specialized software that can handle these event-driven computations. Towards this direction, Intel with the introduction of its new and updated Loihi-2 also unveiled its LAVA software framework [45]. LAVA addresses the need for a common neuromorphic software framework allowing researchers and developers to utilize a common set of tools, methods, and libraries and run neural network models seamlessly on heterogeneous architectures across conventional and neuromorphic hardware. In addition, it enables development of applications without accessing specialized neurompprphic hardware. The kind of applications targeted by these neuromorphic systems are discussed next.

3 APPLICATIONS

SNNs are well suited to process both static as well as sequential data due to their inherent recurrence. Also, SNNs can naturally process discrete spatiotemporal data from event sensors. In this

Table 1. Performance of SNNs on Different Image-recognition Datasets

Paper	Neuron model	Input coding	Learning rule	Network architecture	Accuracy	Timesteps
MNIST						
[52]	LIF	Rate	STDP	2 FC	95%	700
[156]	IF	Temporal	Backprop	784FC-600FC-10FC	96.98%	167
[222]	LIF	Rate	Stochastic STDP	36C3-2P-128FC-10FC	66.23%%	100
[268]	LIF	Encoding layer	Backprop	15C5-P2-40C5-P2-300FC	99.53%	5
[267]	IF	Temporal	ANN-to-SNN	32C5-P2-64C5-P2-1024FC-10FC	99.41%	8
[198]	IF	Temporal	ANN-to-SNN	LeNet-5	98.53%	-
CIFAR10						
[76]	IF	Temporal	ANN-to-SNN	VGG16	93.63%	2,048
[209]	IF	Rate	ANN-to-SNN	VGG16	91.55%	1,000
[199]	IF	Rate	ANN-to-SNN	4 Conv, 2 FC	90.85%	400
[194]	LIF	Rate	Hybrid	VGG16	92.02%	200
[120]	LIF	Rate	Backprop	VGG9	90.45%	100
[222]	LIF	Rate	Stochastic STDP	256C3-2P-1024FC-10FC	98.54%	100
[255]	LIF	Encoding layer	Backprop	CIFARNet	90.53%	12
[268]	LIF	Encoding layer	Backprop	CIFARNet	91.41%	5
[193]	LIF	Encoding layer	Hybrid	VGG16	92.70%	5
ImageNet						
[76]	IF	Temporal	ANN-to-SNN	VGG16	73.46%	2,560
[77]	IF	Rate	ANN-SNN	VGG16	71.34%	768
[199]	IF	Rate	ANN-SNN	VGG16	49.61%	400
[209]	IF	Rate	ANN-SNN	VGG16	69.96%	300
[194]	LIF	Rate	Hybrid	VGG16	65.19%	250
[132]	IF	Encoding layer	ANN-SNN	VGG15	66.56%	64
[193]	LIF	Encoding layer	Hybrid	VGG16	69.00%	5
NMNIST						
[123]	LIF	Event sensor	Backprop	2 FC	98.74%	350 ms
[254]	LIF	Event sensor	Backprop	3 FC	98.78%	300 ms
[213]	SRM	Event sensor	Backprop	12C5-2P-64C5-2P-10FC	99.20%	300 ms
[96]	SRM	Event sensor	Backprop	2 FC	98.88%	0.6 ms
DVS CIFAR10						
[108]	IF	Event sensor	ANN-to-SNN	4 Conv, 2 FC	65.61%	60
[255]	LIF	Event sensor	Backprop	128C3-2P-128C3-256C3-2P-1024FC	60.5%	5
[253]	IF	Event sensor	Backprop	VGG7	62.5%	5

section, we discuss the application of SNNs in image classification, gesture recognition, sentiment analysis, biomedical applications, and motion estimation. Additionally, we review the relevance of SNNs as a defense mechanism against adversarial attacks.

3.1 Image Classification

Table 1 compares the performance of various SNN models on image classification tasks from frame-based image datasets (MNIST [117], CIFAR10 [106], ImageNet [50]) as well as neuromorphic datasets (N-MNIST [170], CIFAR10-DVS [127]). The pixel-based images are converted to spike-train based on the encoding methods discussed in Section 2.2. The learning algorithm is among the variants described in Section 2.3. In SNNs, the challenge is to achieve competitive accuracy with the minimum number of timesteps for better energy-efficiency. To that effect, networks employing pixel values directly as input and training with gradient-based backpropagation methods achieve the best overall performance. **Neuromorphic-MNIST (N-MNIST)** [170] and CIFAR10-DVS [127] are the spiking versions of the MNIST and CIFAR10 dataset, respectively, recorded with a dynamic vision sensor.

3.2 Gesture Recognition and Sentiment Analysis

Sequential classification tasks have inputs that have some temporal dependence between them. Thus, they require models that can keep this dependence into account when generating predictions. The ability of SNNs to inherently capture temporal information makes them directly compatible with such temporal inputs and suitable for sequence classification tasks. Works such as References [10, 213, 256] explore gesture recognition using deep SNNs on the IBM DVS gesture dataset [10]. Their results show that SNNs demonstrate comparable performance to corresponding state-of-the-art ANN implementations. Reference [213] also performs audio classification on the TIDIGITS dataset [125]. However, Reference [4] performs the task of sentiment analysis on movies reviews from the IMDB dataset. This implementation shows that the neuronal membrane potential tracks the positive/negative nature of the sentiment as the review is presented to the SNN as a sequence.

3.3 Bio-medical Applications

Real-world biological signals and patterns are generally time-varying signals, which are a natural fit for SNN-based processing. To that effect, researchers have proposed several solutions for analyzing and classifying these biological patterns. These include works analyzing and decoding signals such as **Electroencephalogram (EEG)** [54, 105, 109, 168, 229], **Electrocardiogram (ECG)** [259, 261], **Electromyography (EMG)** [55, 68], and so on. In addition, authors in References [30, 67, 275] show that **high frequency oscillations (HFO)** generated by the epileptogenic tissue in **Electrocorticography (ECoG)** recordings can also be efficiently processed and analyzed using SNNs.

3.4 Motion Estimation

The emergence of event-based sensors, as discussed previously (Section 2.2.4), have induced promising opportunities for SNNs owing to inherent input compatibility. Edge-devices such as small ground and flying robots incur huge benefits from event-based processing in terms of perception and planning tasks. Authors in Reference [44] demonstrate the importance of using event cameras to efficiently accomplish tasks such as obstacle detection and avoidance while moving at high speeds as well as awareness-based sensing of the environment. While there have been works that utilize ANNs along with event-cameras for such tasks [270], they are inefficient in retaining the rich temporal information held by the events. In contrast, SNNs show inherent compatibility by naturally capturing the temporal nature of event camera data. Authors in Reference [169] demonstrate visual motion estimation using SNNs by accounting for synaptic delays when generating motion-sensitive receptive fields. Reference [75] presented real-time model-based optical flow estimation on IBM's TrueNorth [6] for simple patterns of rotating spirals and pipes. In addition, authors in Reference [177] carried out optical flow estimation using convolutional SNNs trained using STDP-based learning. The main limitations of these works involve the usage of small-scale learning that does not scale well for dynamic and complex inputs. Training deep SNNs using end-to-end backpropagation brings forward new challenges. Deep SNNs suffer from the "spike vanishing" phenomenon, where the number of spikes propagating to the later layers reduce drastically, hindering learning and leading to poor performance. To that effect, a hybrid SNN-ANN architecture, where the SNN layers enable effortless handling of event data while the ANN layers enable end-to-end learning along with maintaining application performance, seems to be a promising alternative. Authors in Spike-FlowNet [118] utilize such a deep encoder-decoder architecture based on the U-Net [196] model for optical flow estimation on the **Multi-Vehicle Stereo Event Camera (MVSEC)** [271] dataset. This dataset consists of various sequences (*indoor_flying*1/2/3, *outdoor_day*1/2) recorded using a stereo pair of event cameras. The model is trained end-to-end

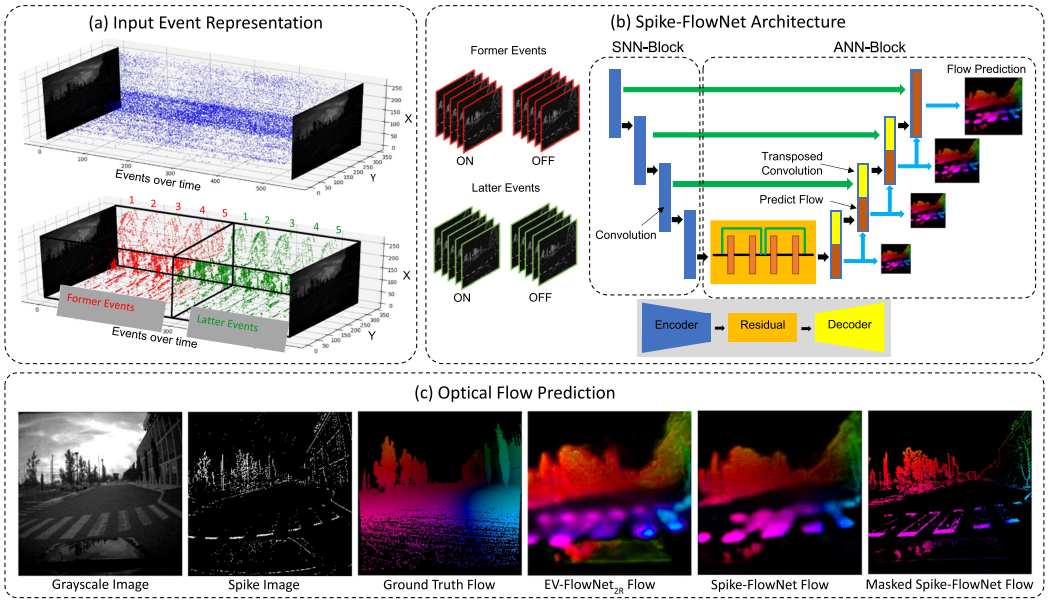

Fig. 11. (a) Input Event Representation. (Top) Continuous raw events and discrete grayscale images from a DAVIS camera. (Bottom) Accumulated event frames between two consecutive grayscale images to form the former and latter event groups. (b) Spike-FlowNet Architecture [118]. The 4-channeled input images, as groups of former and latter events, are sequentially passed through the hybrid network. The SNN-block contains the encoder layers of the network, while the ANN-block contains the residual and decoder layers. The loss is evaluated after forward-propagating all consecutive input event frames within the time window. (c) The predicted optical flow compared with the provided ground truth and EV-FlowNet [270].

using surrogate gradient-based **Backpropagation through-time (BPTT)**, as highlighted in Algorithm 2. Figure 11 highlights the input representation, hybrid SNN-ANN architecture and qualitative results, while Table 2 compares the **Average Endpoint Error (AEE)** with the state-of-the-art ANN implementation Ev-Flownet [270]. Usage of SNNs lead to lower energy consumption due to the **accumulate (AC)** operations in SNNs compared to **multiply-accumulate (MAC)** operations in ANNs. These works establish SNNs as prime candidates for real-world perception and planning tasks, making SNN and event-camera-based computer vision an active area of research.

However, there are traditional computer vision-based works that are again, highly efficient for small-scale inputs but do not offer similar performance or scale well to larger inputs. One such recent work is Reference [142], where the authors propose to estimate optical flow by modelling the temporal events from an event-sensor as a three-dimensional probability distribution parameterized by the pixel address and timestamp. The Fisher-Rao matrix for each parameter value is used to compute the optical flow given by the eigen vector corresponding to the least eigen value. This work demonstrates respectable performance compared to Ev-FlowNet [270] and Spike-Flownet [118], as shown in Table 2. These suggest that learning-based methods inspired by fundamental concepts from the field of traditional computer vision seem to hold a lot of potential towards realizing high-performance perception and planning while maintaining efficiency.

3.5 Adversarial Robustness

Several machine learning models, including neural networks, are vulnerable to adversarial attacks [110]. An adversarial example is an input sample perturbed with carefully crafted noise to

Table 2. Average Endpoint Error (AEE) Comparisons between EV-FlowNet [270], Spike-FlowNet [118], and Fisher-Rao Metric-based Method [142] on the MVSEC Dataset

	dt=1 frame				dt=4 frame			
	indoor1	indoor2	indoor3	outdoor1	indoor1	indoor2	indoor3	outdoor1
EV-FlowNet [270]	1.03	1.72	1.53	0.49	2.25	4.05	3.45	1.23
Spike-FlowNet [118]	**0.84**	**1.28**	**1.11**	**0.49**	**2.24**	**3.83**	**3.18**	**1.09**
Fisher-Rao Metric [142]	1.88	-	-	-	2.95	-	-	-

Lower is better.

cause the neural network to misclassify the input sample. The modified samples are subtle and undetectable by a human observer. Such systems can seriously undermine the security of neural networks for mission-critical applications. For example, a slightly modified image of the "Stop sign" is classified as a speed limit sign [60]. The defense mechanisms include activation pruning, input discretization, and non-linear transfer functions. SNNs inherently possess most of these features and are better suitable to handle adversarial attacks [14, 212]. The input to SNNs is binary, the dynamics of the LIF neuron are non-linear, and the activations are sparse. Also, SNNs can exploit the time and leak parameter to improve the resiliency of the network. The authors in Reference [212] showed that SNN trained with spike-based backpropagation, employing LIF neurons, and lower number of timesteps perform better under adversarial attack compared to ANN as well as ANN-to-SNN-converted networks that generally use IF neuron and require a larger number of timesteps. The authors in Reference [14] studied the robustness of SNNs under different input coding methods with random and adversarial perturbations. Networks trained with rate coding performed better compared to temporally coded network. The reason may be the particular form of temporal coding, first-to-spike, which allows only one spike per neuron, thereby reducing redundancy. To generate an adversarial sample, the gradient of the loss with respect to input is added to the input. In SNNs, the gradient is continuous, whereas the input is binary. The authors in Reference [128] proposed a method to convert continuous gradient to spike-compatible ternary gradients with probabilistic sampling. Adversarial robustness is an active area of research, and SNNs with their unique properties can potentially provide a low-power defense mechanism.

4 NEUROMORPHIC HARDWARE

We have described how neuromorphic computing presents a novel paradigm with diverse neuronal functionalities as well as synaptic learning algorithms. In this section, we delve into the design of neuromorphic hardware [41, 231] that can faithfully emulate the algorithmic functionalities and leverage the inherent computational efficiency offered by SNNs.

4.1 Motivation for Neuromorphic Hardware Design

Neuromorphic hardware design was initially inspired by building electronic equivalent of the human brain to mimic its computational capabilities [143]. This was demonstrated by Mahowald and Douglas in 1991 [139] with an analog integrated circuit for silicon neuron followed by a silicon retina for stereoscopic vision [137].

Over the recent years, growing interest in artificial intelligence and machine learning systems has led to domain-specific acceleration of such workloads in systems such as GPUs, TPUs, and so on, due to the data-intensive nature of the workloads. However, designing efficient neuromorphic hardware presents further challenges that need to be addressed. Neuromorphic computing workloads such as SNNs are inherently temporal in nature, i.e., it evaluates the neural network model over a number of timesteps. Furthermore, due to the time-dependent processing of the spiking neurons, there exists little to no temporal parallelism that can be exploited without breaking the pipeline of the processing. Thus, the data-level parallelism is confined within a single timestep of

processing, which of course can be leveraged through GPU or TPU acceleration. However, SNN workloads tend to show event-driven characteristics, i.e., the neurons process data asynchronously, leading to considerable sparsity in its activations at a given time. Moreover, spiking activity of the neurons can diminish in deeper layers in a deep SNN. Hardware systems such as GPUs and TPUs are designed to efficiently leverage data-parallelism, but they fail to exploit the high temporal sparsity as well as spatial sparsity of activations in SNNs. Additionally, SNNs have memory-intensive data-structures such as membrane potential that need to be processed across timesteps, an overhead that is not entirely mitigated by today's digital accelerators. There have been some proposals on using analog capacitors as memory elements in neurons [89] for "state-ful" processing, which could serve as a good design choice for mixed-signal SNN accelerators. Finally, the training algorithms of SNNs could involve both global as well as local weight updates across time, as described in Section 2, which can introduce further bottlenecks in efficient execution of such operations. Considering the limitations of hardware systems designed to accelerate machine learning workloads, researchers have explored several avenues [6, 11, 21, 31, 46, 89, 208, 231, 262] across the design stack from devices and circuits to architectural solutions to address the unique challenges presented by neuromorphic computing workloads such as SNNs. In the next subsection, we will discuss how intelligent design of basic compute primitives forms the platform for acceleration of SNN workloads.

4.2 Neuromorphic Architecture

The key facets of designing Neuromorphic architectures involve techniques that effectively utilize the temporal and spatial activation sparsity in SNNs, as well as optimize the basic computing architecture, such as building efficient functional primitives and coordinating the communication of various data-structures. The first challenge can be tackled through sparsity-driven optimizations, which include conditional activation of memory and processing units along with asynchronous communication [6, 46].

4.2.1 Sparsity-driven Solutions – Asynchronous Systems. We have discussed in Section 2 how SNNs have abundant temporal and spatial sparsity in neuron activity. One common way to leverage such sparsity is to use hierarchical mesh architecture and asynchronous communication as adopted by various researchers [6, 46, 154]. An example mesh operation, implemented in the *Loihi* chip, is shown in Figure 12(a). It is a fully asynchronous many-core chip where each core has its own sense of frequency and timing. Interaction between cores is also asynchronous, and not timed, and the operation of one neuron in the system is entirely independent of another. The operation of a core in the system commences at a local time t, and through iteration of the neuron compartments in each core, firing events are monitored. In the event of firing, the **network-on-chip (NOC)** broadcasts a spike message to only the cores that contain the synaptic fan-out of the firing core. Once the slowest core finishes processing, a synchronization mechanism between neighboring cores ensures that all spikes are safely delivered and received before proceeding to timestep $t+1$. A unique property of *Loihi* is that it deploys an entirely asynchronous system where different modes of operation within the core microarchitecture can operate at different frequencies, in addition to the event-driven communication system through the mesh network. The authors suggest that this local dataflow control suffices the necessity of varying workload-dependent timescales for spiking neuron processes as well as facilitates back-end timing closure.

An alternative way of realizing asynchronous systems is adopted by *TrueNorth*, where the computing cores are synchronous in nature, whereas the communication protocol between the cores through the NOC routers is asynchronous. This ensures that all cores operate in parallel and an asynchronous control circuitry enables only the cores in case of a need for synaptic integration and

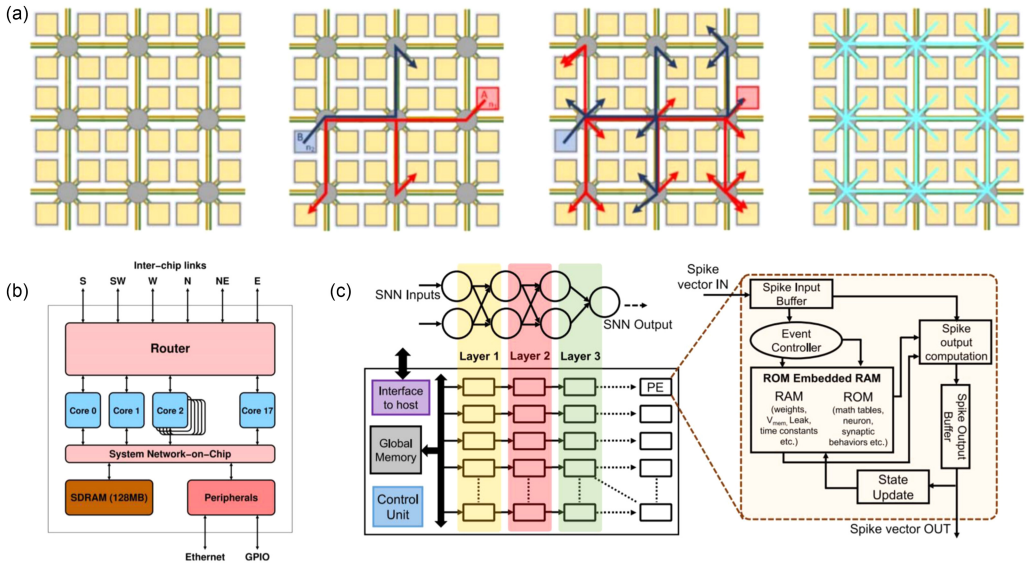

Fig. 12. (a) Mesh architecture adopted in *Loihi* for transmitting spikes across various neurons [46], (b) Multi-core implementation of neuromorphic processors connected to a common router [171] and (c) Near-memory processing architecture with a 2-D array of processing elements, with each PE consisting of internal storage for neuronal functionalities as well as computing elements for synaptic computations [5].

membrane potential update. The local synchronous digital circuit design facilitates low complexity design for complex neuron functionality.

Some extensions of mesh-based architectures involved hierarchical mixed-mode routing systems where various distinct levels of routers are deployed. For example, in Reference [154], three levels of routers are deployed: one responsible for local traffic, a second responsible for non-local events for nearby cores, whereas a third level router is responsible for long-distance communication. This chip, also knows as DYNAPs, demonstrates the efficacy of the novel digital communication scheme along with emulation of neuro-synaptic functionalities on CMOS analog circuits.

Event-driven systems can also be incorporated using 2-D processing arrays connected to a common packet router, as adopted by the SpiNNaker Project [171], shown in Figure 12(b). Events are communicated by maintaining routing tables for event sources. The processing elements simply emit a spiking event in form of a packet with the address of the spiking neuron and the router communicates the packets to any of the other processing cores where the selected routes are determined by the routing table.

In addition to the aforementioned approaches to building large-scale neuromorphic systems, researchers have explored scalable implementations that faithfully realize complex neuronal and synaptic functions using analog circuits. For example, the BrainScaleS system [203] performs wafer-scale integration of multiple instances of the HICANN ASIC, which implements the Adaptive Exponential Integrate-and-Fire Model using analog circuit along with synapses capable of performing STDP-based learning. Neurogrid [21] is a mixed-signal large-scale system to perform brain simulations and visualization. It used a combination of highly bio-plausible neural and synaptic sub-threshold analog circuits with detailed modeling of soma, dendritic trees, axons, and so on. Multiple cores communicate with each other using digital **Address Event Representation (AER)** protocol. The basic skeleton of neuromorphic systems with AER architecture containing an **Integrate-and-Fire Array Transceiver (IFAT)**, a **look-up table (LUT)** to store synaptic

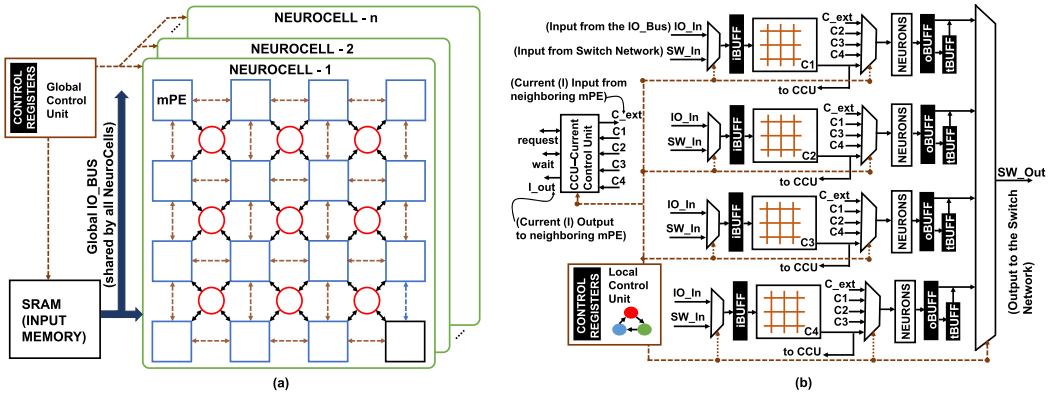

Fig. 13. (a) RESPARC as a pool of NeuroCells (b) Macro Processing Engine - The mPE receives input spikes over the bus and the switch network, which is processed by the crossbars to produce output currents: C1, C2, C3, C4. The crossbar currents get integrated into the neurons to produce output spikes that are then sent to the target neurons over the network.

connections, and AER protocol for communication between neurons is also used as a popular approach. Variants of IFAT systems have been explored using different kinds of neuronal circuits [239, 240]. A reconfigurable mixed-signal design called ROLLS, targeting both emulation of complex neuronal and synaptic learning functionalities, and image classification tasks, have been proposed [187]. Although this design derives motivation from various previous designs, it performs a holistic integration and performs complex tasks in a fully on-chip format.

Large-scale reconfigurable systems [244] have been explored to overcome the Liebig's law for neuromorphic hardware, where the performance can be limited by the component with shortest supply. Such reconfigurable systems consist of arrays of identical components, which can be configured as LIF neurons, a learning synapse with STDP-based rules, as well as an axon with trainable delay. Efficacy of such a system has been prototyped using a **field programmable gate array (FPGA)**-based design and later demonstrated on Silicon.

4.2.2 Efficient Computing Architectures. However, researchers have also explored tiled architectures to reduce data movements over the network. SPARE [5] and RESPARC [11] are examples of tiled architectures built with CMOS and post-CMOS primitives, respectively, that achieve significant improvements in efficiency for SNN execution. As shown in Figure 12(c) Each tile in SPARE is composed of ROM-Embedded RAM-based arrays, where RAM stores the weights for a small partition of the SNN, and ROM stores lookup tables for executing transcendental operations that can perform complex neuron functions. During an SNN execution, each tile performs multiply-and-accumulate and transcendental operations on its portion of weight data (weights remain stationary), thereby enabling near-memory computing. Typical SNN execution on SPARE is performed in a time-multiplexed manner, where each layer is mapped on the tiled architecture, one at a time. The currently mapped layer computes its outputs before the next layer is mapped.

RESPARC further extends the tiled architecture by leveraging the high storage density of hlmemristive crossbars to enable large number of on-chip tiles. However, expensive crossbar writes limit the applicability of a time-multiplexed architecture, where the crossbars are reused across layers by re-programming weight matrices and executing the corresponding dot-product operations. Consequently, a spatial architecture where the weight data of an entire SNN are pinned to crossbars located across multiple tiles is more efficient, as it leverages the benefits of high storage density while alleviating the costly writes. Figure 13 illustrates a tiled architecture built with post-CMOS

primitives (memristive crossbars). It is worth noting that while such a spatial architecture is effective for SNN inference, it may still suffer from costly writes in SNN training where weight data is updated frequently. Recent explorations on hybrid ANN-SNN architectures by partitioning the network into non-spiking and spiking counterparts have also shown promise to mitigate the algorithmic inference latency disadvantage of SNNs [216].

Alternatively, an SNN accelerator such as *Spinal – Flow* [163] has explored techniques to tackle the iterative memory access overhead over multiple timesteps for updating membrane potential and weight read. They propose a novel dataflow that leverages the temporal re-use patterns in SNN workloads. By creating a compact sorted list of spikes, *Spinal – Flow* sequentially walks through the spikes of all timesteps in a layer to yield significant speedups due to activation sparsity. The architecture uses an output stationary dataflow to map neurons to the processing elements across all timesteps. This allows the membrane potential to accumulate for an input across all the timesteps in a layer, before proceeding to next layer, thereby eliminating additional storage and data-movement costs for the membrane potential data-structure. Such a dataflow maximizes re-use of membrane potential, in contrast to dataflows followed in ANN accelerators, which aim to maximize re-use of data-structures such as inputs, weights, and outputs.

4.3 Asynchronous Communication

The communication between different neurons in a large-scale neuromorphic chip requires high speed, time-multiplexed asynchronous circuits. Typically, such circuits are serviced by a transceiver to read and write spikes in parallel and an on-chip router for transmission of neuronal events or spike. The most commonly used asynchronous communication protocol is **address-event representation (AER)** [27, 137]. The signals in the AER protocol carry analog information in form of "inter-spike intervals" using the digital medium, where addresses of the nodes responsible for events are represented in binary. In the most primitive case, a look-up table with source and destination pairs of addresses is maintained to determine connectivity between different nodes.

The primary obstacle of interconnected SNN hardware implementation is the high memory requirement for the look-up table that stores the network connectivity information. To accommodate arbitrary connectivity among n neurons, the routing table requires $O(n)$ entries, which can be mostly redundant. Instead, if the neurons, have preferential connectivity to local neighboring neurons, then routing protocols can be designed to trade off network configuration flexibility with routing memory. To that effect, multi-stage [153] and hierarchical [35] routing schemes have been explored to reduce memory requirements for implementing reconfigurable large-scale SNNs.

In addition to routing schemes, asynchronous communication in large-scale neuromorphic systems also require efficient on-chip routers. Typically, on-chip routers for neuromorphic systems need to have unique features such as low latency multi-cast routing to support high connectivity. Researchers have proposed tree-based routers [147] which uses recursive branching to broadcast packets to corresponding subtrees. By reducing the number of nodes to navigate, this approach reduces memory look-ups and also the size of the header of the data packets.

The delays in aforementioned asynchronous communication systems become critical since neuromorphic systems use accelerated timescales where circuits operate much faster than biological systems. The latency and performance of any AER-based neuromorphic system is hence fundamentally limited by communication latency as well as its memory requirements.

4.4 CMOS-based Neuromorphic Compute Primitives

The computing primitives for efficient neuromorphic systems have two design facets: (a) implementing complex neuronal [90] and synaptic learning functionalities [15] and (b) deploying low-power synaptic integration functions. We first describe how these design facets are realized in

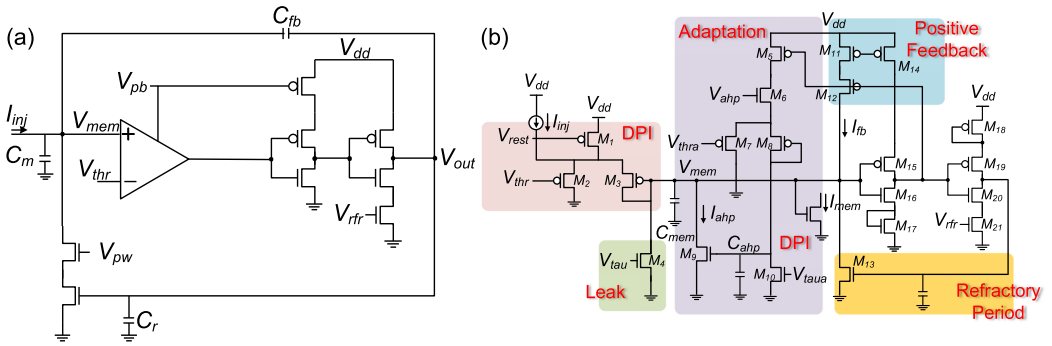

Fig. 14. Illustration of CMOS-based neuronal circuits. (a) Circuit shows basic comparison and firing behavior of an IF neuron [87] (b) Circuit shows more complex functionalities of an IF neuron, such as spike frequency adaptation, leak as well as refractory period [236].

CMOS technology, which has been a popular thrust of research in neuromorphic circuit design. Next, we will delineate how novel memory technologies can be effectively used to accelerate the aforementioned computation and realize the basic processing units in a compact fashion.

4.4.1 Neuronal Circuits. The classical case of representing neuronal circuits on silicon was based on equivalence between ion transportation in biological neurons and electron transport in transistors. Researchers have shown that the ionic channel transport in biological neurons can be modeled using very few transistors, operating in the sub-threshold domain [61]. Such sub-threshold transistors have been also leveraged to implement Hodgkin-Huxley-based neuron models [263] using programmable kinetics of the gating variables.

However, the more widely popular neuronal functionality is IF or LIF, which we have described in detail in Section 2. The abstract view of such a neuron primarily consists of a capacitive unit that holds and updates the membrane potential, a comparison unit, and a thresholding unit. The most primitive form of IF neuron was conceptualized back in the late '80s [143] with a simple feedback circuit that could produce fixed width, fixed height voltage pulses, where the rate of the pulses was proportional to the input injection current (arriving from synapses), and the temporal characteristics of the pulses represented the shape of the input current waveform. A subsequent design of IF neuron circuits in the analog domain involved incorporation of a comparator unit, as shown in Figure 14(a) [145, 236]. In this design, the injected current is integrated using the membrane capacitance, C_{mem}, and then fed to a comparator circuit to compare the resulting V_{mem} against the threshold voltage V_{thr}. The capacitive feedback, C_{fb}, in both circuits ensures that small fluctuations of V_{mem} around V_{thr} do not affect firing activity. The updated circuit also has a control for the refractory period of the neuron. Initially, after firing, V_{mem} decreases linearly, but once it falls below V_{thr}, the output of the first inverter sets high, leading to discharge of capacitor C_r at a controlled rate using V_{rfr}. This ensures V_{mem} does not start to increase, as long as voltage at C_r is above a certain value.

Analog IF neuron circuits have evolved significantly over time by incorporating more features, such as reset, leak, as well as spike frequency adaptation, and so on. Figure 14(b) shows a fairly complex and generalized IF neuron circuitry [89]. Such a neuron circuit consists of an input **differential pair integrator (DPI)**. The integration is performed by the membrane capacitor, C_{mem}, and the spike generation scheme is implemented using an inverting amplifier with positive feedback. The reset behavior is implemented using transistor M13, and together with transistor M21, refractory-period behavior is implemented. Transistors M5–M10 produce the current proportional

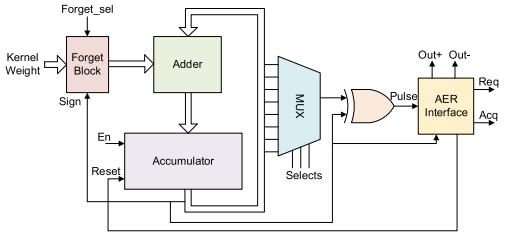

Fig. 15. Illustration of a fully digital IF neuron.

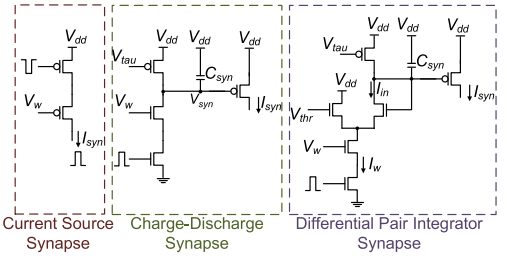

Fig. 16. Illustration of three different kinds of CMOS-based synaptic circuits.

to the neuron's firing rate, and hence lead to spike frequency adaptation mechanism. The modified equation of the implemented LIF neuron is:

$$C_{mem}\frac{d}{dt}V_{mem} = (I_{dpi} - I_{\tau}) - I_{ahp} + I_{fb}. \tag{10}$$

This generalized neuron realizes an adaptive, exponential IF neuron.

Other types of analog neuron implementations based on CMOS have also been explored, such as the log-domain LPF neuron [12], which implements a reconfigurable IF circuit. Yet another type of IF neuron circuit is called the "Tau-Cell neuron" [191, 237], which uses current-mode circuits, and the membrane potential is represented as a current. More compact IF neuron circuits have been proposed using above-threshold transistors, such as the quadratic IF neuron [249]. Such a neuron is loosely based on the Izhikevich neuron model [92] where two state variables are maintained across two separate capacitors instead of one. Leaky IF neurons have been also implemented using switched capacitor circuits where the switches are used to implement leak behavior between the membrane potential and resting potential [64]. The switch capacitance technique motivates more digitally inspired neuron designs. One such design involves weighted current mirror circuits activated by a binary coded digital weight [211]. After, the neuron generates a positive and negative spike based on the excitatory/inhibitory nature of the synapses.

Leading from digital inspiration of the previous neuron design, IF neurons have also been explored in fully digital mode [33]. A digital adder and accumulator along with comparator circuits can be used to implement integration and spike-generation behavior of IF neurons, shown in Figure 15. Leak in such a neuron is implemented by a fixed weight synapse driven by a global clock.

4.4.2 CMOS-based Synaptic Circuits. Synaptic circuits were first conceived by Carver Mead [145] as pulsed current-sources with transistors operating in the subthreshold domain, as shown in Figure 16. The output of the circuit is simply a synaptic current, I_{syn}, which is a pulse with width proportional to the width of the input spike. An extension to the aforementioned scheme involves exponential decay of the post-synaptic spike using the mechanism of charging and discharging of the node V_{syn} [12, 115]. When an input pulse is applied, the node V_{syn} decreases linearly, where the rate of decrease is governed by the difference in current through the transistors biased using V_{tau} and V_w, respectively. Consequently, the synaptic current I_{syn} exponentially increases. When there is no input spike, I_{syn} discharges exponentially at the rate governed by current through the transistor biased using V_{tau}. An alternative synaptic circuit can be implemented using a differential pair, more commonly known as a **differential pair integrator (DPI)** circuit [89]. During the charging phase, current through one branch of the differential pair represents the input current to the synapse. This DPI synapse has more independent control of the time-constant of charge and discharge of the synaptic voltage node. Implementation of plasticity, both

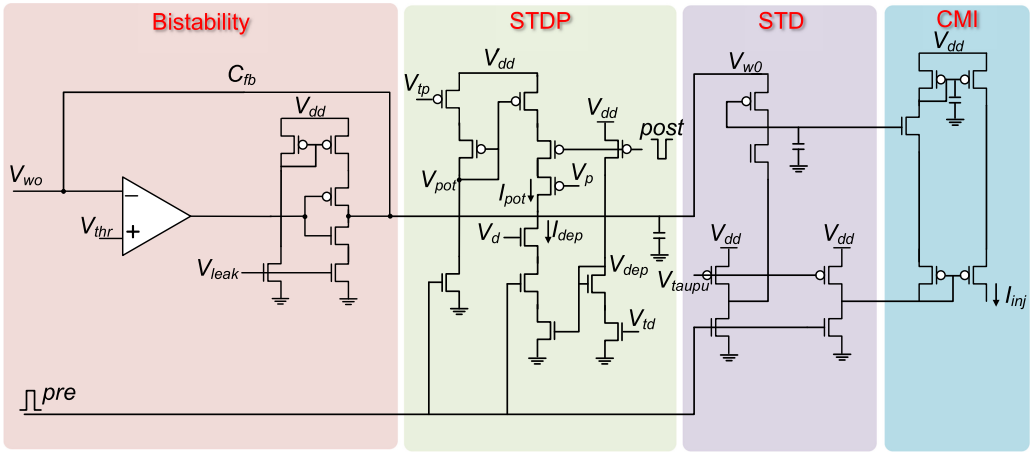

Fig. 17. Illustration of CMOS circuits consisting of subthreshold transistors exhibiting various complex synaptic functionalities such as STDP, short-term depression (STD), as well as integration [89].

short-term and long-term, in synapses requires additional circuitry. Unsupervised learning necessitates automatic update of weights based on internal signals rather than providing the update signals externally for individual synapses. Figure 17 shows an excitatory synaptic circuit [89] that implements both **short-term plasticity (STP)** as well as **long-term plasticity (STDP)** while using a **current mirror integrator circuit (CMI)** to enable the integrating behavior of a synapse. The CMI circuit operates similar to the integrating circuits we have discussed before. On arrival of a spike on the "pre" node, the CMI's integrating capacitor gets charged, while in absence of the spike, the charge decays through the diode connected transistor.

The STP in Figure 17 operates on the synaptic weight voltage V_{w0}. When spikes are applied to the "pre" node, the synaptic weight reduces at a rate controlled by the bias voltage. Thus, the synaptic weight goes through a short-term depression, i.e., it is maximum at onset of the spikes and reduces gradually on consecutive application of spikes at the "pre" node. The STDP mechanism, described in Section 2, is implemented using the STDP circuit shown in Figure 17. Specifically, the circuit modulates the analog voltage V_{w0} based on the relative difference between the pre- and post-spike times. Two waveforms, V_{pot} and V_{dep}, are generated based on the presynaptic and postsynaptic pulses, respectively. The pre- and post-spikes activate two transistors that control the current that cause increase and decrease in the V_{w0} node. The bias voltages V_p and V_d set a limit on the current injection and removal from the capacitance at the node V_{w0}. The transistors in the middle branch that carry currents I_{pot} and I_{dep} operate in the subthreshold region to enable an exponential relationship required for STDP. The bistability circuit shown in Figure 17 is required to hold the analog value of synaptic weight, since CMOS capacitors are prone to leakage of charge. In absence of spiking activity, the bistability circuit generates a constant leak current, which pulls the weight node V_{w0} towards one of the two stable states. The voltage V_{thr} that is fed to the comparator is set externally. If STDP circuits cause the synaptic weight to fall below the threshold, then the bistability circuits allow a negative current to flow to drive the weight towards an analog value representing the depressed state. When synaptic weight increases above the threshold, a positive current flows to drive the weight to a high state. Similar approach has been followed in other works with different variants of plastic synapses. For example, one chip called ROLLS [187] uses separate synaptic arrays for short-term plasticity and long-term plasticity along with custom neuronal circuits based on a variant of the adaptive exponential IF neuron circuit described earlier.

Researchers have also explored CMOS **floating-gate (FG)**-based devices to represent synapses [78, 190] due to their ability to hold on to the synaptic weights indefinitely. To implement plasticity, the synaptic weights are updated considering the injection and tunneling currents in FG transistors. The currents exponentially depend on the gate and tunneling voltages as shown below:

$$I_{inj} \propto exp(-\Delta V_g/V_0), \tag{11}$$

$$I_{tun} \propto exp(-\Delta V_{tun}/V_{ox})exp(-\Delta V_g/V_0). \tag{12}$$

The change in weight is directly proportional to the currents. For long-term potentiation, the change in weight exponentially decays with increasing spike timing difference if a linear dependence exists between the gate voltage, V_g with the time difference. The injection current governs the long-term potentiation and the tunneling current causes long-term depression. When both currents are combined, it results in STDP behavior. When a post-spike arrives, an injection pulse and then a tunnel input is applied. When the input spike is delayed, it occurs during the tunneling phase, which causes a negative change in weight. When input spike occurs before post-spike, the gate voltage goes down, and during the subsequent tunneling phase, it increases, resulting in exponential reduction of tunneling current, and hence a positive change in weight.

4.5 ROM-embedded RAM as Neuronal Function Storage

We have previously mentioned how efficient computing architectures can be composed of tiled **near memory processing (NMP)** systems to reduce data movement. A key factor of implementing such NMP systems is high on-chip storage density, which would alleviate the need for costly DRAM accesses. Improvements in storage densities can be achieved through recent proposals of placing **Read-Only Memory (ROM)** into **Static Random Access Memory (SRAM)** caches to obtain a **ROM-embedded cache (R-cache)** [122] architecture without incurring degradation in area or performance. Standard storage units in caches are **6-transistor (6T)** SRAM cells that usually consist of 2 access transistors and 2 cross-coupled inverters. The new memory cell involves adding an extra word-line to the standard 6T SRAM cell, as shown in Figure 18.

This standard cell design can be operated to achieve the functionality of both SRAM and a ROM. In the RAM mode, the wordlines WL1 and WL2 are connected together. They are turned ON and OFF together to operate the cell as a conventional 6T-SRAM for memory read and write. There is no performance degradation compared to a standard 6T-SRAM cell. The ROM functionality can be realized through a number of sequential steps. Step 1: "1"s are written to an entire row of bit-cells by activating both WL1 and WL2. Step 2: "0"s are written to the bit-cells connected to WL2 by turning on WL2 and keeping WL1 off. Note, only bit-cells connected to WL2 stores "0," while others store "1." Now, if consecutive bit-cells have different ROM data, then this step would end up performing a 5T write operation on the SRAM cell using the connnected access transistor. The ROM data is now accessible through conventional RAM read. The initial ROM retrieval process destroys RAM contents, so the RAM data should be backed up to use the design in the ROM mode. The ROM-embedded RAM structure thus bypasses the need to access external

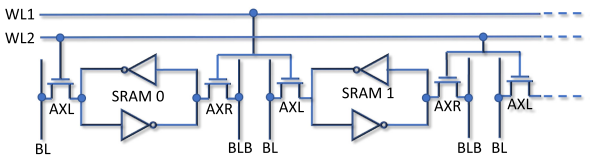

Fig. 18. Standard 6T-SRAM embedded with ROM. An additional word-line is used to retrieve the ROM data. This primitive can work both in RAM and ROM mode. In RAM mode, the WLs are shorted and the primitive operates as standard SRAM. In ROM mode, the WL connections enable writing of ROM data into the SRAM cells.

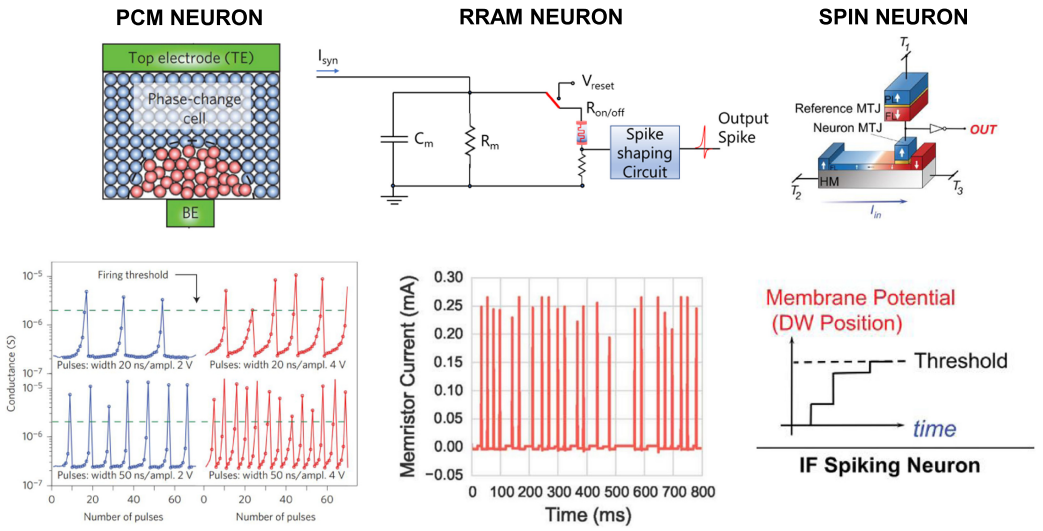

Fig. 19. Neuron devices and circuits based on various NVM technologies, such as PCM, RRAM, Spintronics, exhibiting IF characteristics.

memory to read necessary data such as look-up tables, and so on. These look-up tables can be used to store transcendental functions, as well as complex neuronal functions required for SNNs [5].

4.6 Non-volatile Memory-based Compute Primitives

In spite of significant advances in CMOS technology to emulate the logical adjacency of processing and storage elements in SNNs, there is a need to explore alternative technologies for two particular reasons. First, to improve efficiency of NMP systems, there is a need for higher on-chip storage density. Second, emulating complex neuronal and synaptic functionalities in CMOS technologies can lead to high area consumption. To that effect, **non-volatile memory (NVM)** technologies such as **Resistive RAMs (RRAM)**, **Phase Change Memories (PCM)**, and Spintronics offer significant promise [37]. They allow component-wise direct emulation of neuronal/synaptic functionality at a one-to-one level by the underlying device characteristics. Such a feature along with their high on-chip storage density and ability to perform massively parallel in-memory computations make NVM technologies particularly suitable for neuromorphic systems.

Originally conceptualized by L. Chua [42] in 1971, the fundamental component in NVM technologies, the memristor, was materialized by HP Labs in 2008 [225]. The main deviation in the basic functionality between memristors (memory+resistor) and conventional CMOS transistors (Boolean switches) arise from the fact that they are non-volatile programmable analog resistors. This enables us to mimic the computational units of neurons and synapses directly in the underlying device resistance state. Following naturally through principles of physics, these NVM devices can be arranged in array structures to realize highly compact and energy-efficient "In-Memory" dot-product computing kernels essential for neuromorphic computations. Each NVM device conductance state encodes the corresponding synaptic weight. Input spikes are applied as voltages along the rows of the crossbar array. The currents flowing through each device are weighted by the device conductance and get summed up along the column of each array to realize the dot-product operation. The crossbar array can be also interfaced with resistive neuronal devices to implement the neuronal processing operation. Important metrics for neural/synaptic devices are

the bit resolution available for programming, programming energy and speed, ratio of the maximum to minimum device conductance, reliability, and endurance [112, 201, 251, 273].

4.6.1 Neuronal Circuits. The rich device physics of NVM technologies can enable efficient emulation of neuronal and synaptic functionalities in single devices. Various material stacks are being actively investigated as synaptic and neuronal elements. For instance, **phase change materials (PCM)** are being currently investigated where a chalcogenide material sandwiched between two electrodes can be switched between amorphous and crystalline states due to the heating effect induced by current flowing through the electrodes [24, 250]. The variable current through the device in different states can be used to implement integrate and fire neurons, where the membrane potential is temporally integrated by successive crystallization pulses. The device changes its state to crystalline beyond a threshold, and it is reset to the amorphous state subsequently. The reset mechanism introduces stochasticity in the IF neuron, since each individual amorphous state is different. Researchers have explored applications such as temporal correlation between data streams using such stochastic IF neurons [233]. PCM-based neurons have been experimentally demonstrated in both electrical [234] and photonic domains [39].

Metal oxides such as SrTiO3 [248], as well as HfOx [72], TiOx [195], have been explored as an alternative material for constituting a class of device, known as RRAMs. Alike PCM, RRAM devices can also be used as LIF neurons by connecting it parallel to an external capacitance, as shown in Figure 19. The internal membrane potential is encoded in the conductance of the RRAM device. When the RRAM is in its ON conductance state, the current through the circuit suddenly increases, which leads to an analog spike. The voltage across the RRAM device represents the LIF characteristics. Other types of RRAM-based neurons involve controlling the migration of oxygen vacancies using post-synaptic pulses [114, 146].

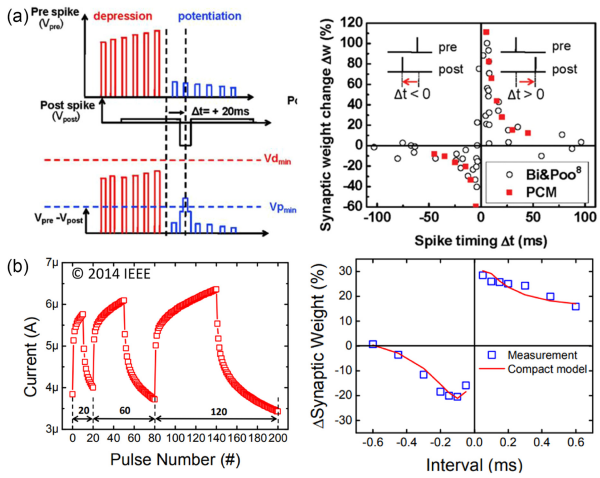

Fig. 20. (a) STDP characteristics can be emulated in RRAM synapses using repeated pulsing schemes [©2014 IEEE. Reprinted, with permission, from Wang IT et al. In 2014 IEEE International Electron Devices Meeting 2014 Dec. 15 (pp. 28.5.1–28.5.4). IEEE]. (b) STDP learning in PCM synapses [111] emulates neuroscientic experiments [186] (right) using series of pulses of increasing amplitude [Reprinted (adapted) with permission from *Nano Letters* 2012 May 9;12(5):2179–86. Copyright (2012) American Chemical Society].

RRAM and PCM devices are often characterized with high switching times and power over the years. Although, more recently, through extensive material research, the switching times of RRAM devices have been brought down a few *ns* timescale [1]. As an alternative technology, researchers have explored Spintronic devices such as Magnetic Tunnel Junctions [65] as LIF [93] as well as stochastic neurons [205]. MTJs are formed using two **ferromagnetic (FM)** nanomagnets with a spacer layer (MgO) sandwiched between them. MTJ can exist in two different resistance states based on the relative direction of magnetization of the two FM layers. The spin dynamics of an FM can be expressed effectively using the

stochastic-Landau-Lifshitz-Gilbert-Slonchewski (s-LLGS) equation:

$$\frac{\partial \hat{m}}{\partial t} = -|\gamma|(\hat{m} \times H_{EFF}) + \alpha \left(\hat{m} \times \frac{\partial \hat{m}}{\partial t} \right) + \frac{1}{qN_s}(\hat{m} \times I_s \times \hat{m}), \tag{13}$$

where $\hat{m}$ is the unit vector of free layer magnetization, H_{EFF} is the effective magnetic field including the shape anisotropy field, external field, and thermal field, γ is the gyromagnetic ratio for electron, α is Gilbert's damping ratio. The first two terms of the LLG equations is often used to represent the "leak" behavior, while the last term encodes the integrating behavior. This makes spin devices particularly amenable to be used as LIF neurons. Alternate spin dynamics such as the **magneto-electric (ME)** switching has also been explored as LIF neurons [93], where a ME oxide layer is used a capacitance to induce leaky behavior. IF neurons can also be implemented using larger magnets, more commonly known as **domain-wall magnets (DWM)** [207], as shown in Figure 19. The membrane potential is integrated through motion of the domain-wall, and it reaches a threshold value when the domain-wall reaches the extremity of the magnet.

Furthermore, the switching dynamics on Spintronic devices is a strong function of a thermal field, which leads to stochastic behavior. Such stochastic behavior can be leveraged to design stochastically switching binary neurons, described in Section 2.1.2. Since such neurons require frequent switching, they are often based on **Spin-Orbit-Torque (SOT)** MTJs for reduced power consumption [205]. SOT-MTJs consist of a heavy metal at the bottom of the MTJ stack and leverages the principle of **Spin Hall Effect (SHE)** to produce magnetization switching at lower currents.

Besides the RRAM, PCM and Spintronic technologies, recent interest in **Ferro-electric Field Effect Transistors (FEFETs)** have led to exploration of neuromorphic devices based on it. The abrupt switching of FEFET on application of repeated pulses can be leveraged to mimic functionality of IF neurons [159, 160]. The firing dynamics is a function of the amplitude and duration of the pulses. The leak behavior in a FEFET-based neuron has been experimentally demonstrated in an HZO thin film [56].

4.6.2 NVM devices for Synaptic Learning. The variable states in NVM devices can also be used to perform synaptic learning using both unsupervised and supervised weight update mechanisms, described earlier. Unsupervised learning schemes such as STDP have been experimentally demonstrated using PCM synapses [8, 111]. The conductance of synapses can be gradually increased or decreased through successive pulses to perform both **long term potentiation (LTP)** and **long term depression (LTD)**. The pre-spikes consist of pulses of gradually changing amplitude, and post-spike is a negative pulse, as shown in Figure 20(a). The overlap of pulses is a function of the spike time difference, and the change in conductance is proportional to the overlap. Other kinds of schemes involve usage of two PCM devices [26], one each for LTP and LTD. PCM devices based on photonics have also been demonstrated to perform on-chip STDP learning [40, 62]. In this technology, the change in optical response through refractive index modulation is used to change the state of the devices.

The state modulation in RRAM devices can also be used to perform STDP through differential pulse shaping of pre-synaptic and post-synaptic voltages [184, 189]. Like PCM device, gradual increase/decrease of conductance is also be achieved through a series of identical pulses [210, 243]. Since controlled pulses maybe difficult to generate, STDP has also been explored with an additional transistor, i.e., with a 2T/1R synapse [246] to enable more precise control.

We have discussed earlier how Spintronic devices can be programmed to have multi-level states using multi-domain magnets. Such multi-level states enable synaptic behavior whose weight can be encoded by the position of the domain wall in the device. Through the use of extra transistors [204], STDP learning is enabled by leveraging the exponential characteristics of the transistors

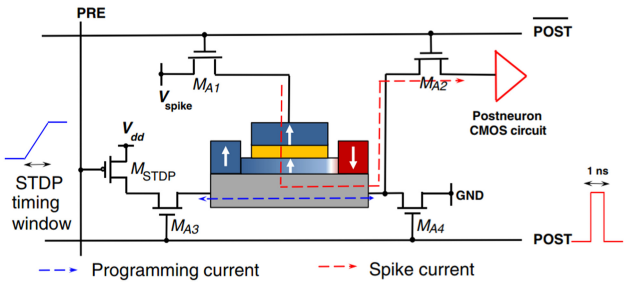

Fig. 21. STDP learning scheme in DWM-based spin synapse [204] using peripheral transistors. [Reproduced with permission *Physical Review Applied* 2016 Dec. 8;6(6):064003., Copyright 2017 American Physical Society].

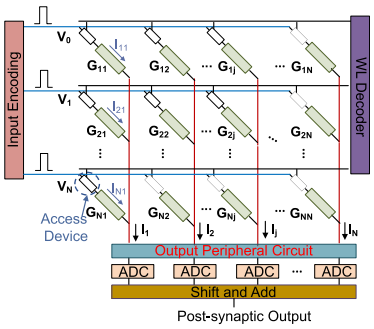

Fig. 22. Illustration of a typical NVM crossbar with necessary peripheral circuits.

in sub-threshold regime, as shown in Figure 21. The transistor M_{STDP} operates in the sub-threshold regime, and a linearly increasing gate voltage is applied at the time when pre-spike arrives. When the post-spike arrives, a programming current, exponentially dependent on the time difference of the spikes, flows through the heavy metal layer. The transistors MA2 and MA4 in Figure 21 can be removed when connecting multiple such synaptic devices in a crossbar fashion [204]. Due to the low resistance range of Spintronic devices, encoding multiple states can hurt functionality. To that effect, regular MTJs encoding binary information can also be used as synapses. A variant of STDP learning, namely, stochastic STDP, has been explored in binary synapses, which can lead to significant energy-efficiency [224, 238] due to low operating currents. To achieve multi-level stochastic STDP, researchers have explored multiple MTJs to represent a single synapse [266].

The switching behavior in FEFETs produces bi-stability that makes them suitable for synaptic operations. FEFET-based synapses can also achieve multiple levels and have been experimentally demonstrated [43, 94, 161]. STDP-based learning scheme has been achieved through conductance potentiation and depression using gradual threshold voltage turning.

4.6.3 NVM Crossbars. We have discussed briefly how NVM devices can be arranged in a matrix formation to form a crossbar to constitute ultra-dense memories as well as serve as primitives for highly parallel "In-Memory Computing." More specifically, such crossbars based on NVM devices can perform **matrix-vector multiplication (MVM)** operations efficiently. The conductance of the NVM devices stores the values of the matrix elements while the vector is provided as voltages to the word-lines of the crossbar, as shown in Figure 22. The multiplication occurs in the NVM device itself, and the product, which is the current through each device $I_{ij} = V_i * G_{ij}$, is summed up in the bit-line.

In addition to performing efficient MVM synaptic computations, learning mechanisms such as STDP have been demonstrated in NVM crossbars based on all the aforementioned technologies. For example, in PCM technology, small-scale arrays of size 10×10 consisting 1T-1R PCM cells have been used to perform on-chip STDP [58, 59]. With an additional transistor, the arrays could be further scaled [104]. PCM-based crossbars have also been explored [38] and even experimentally demonstrated [62] in the photonic domain. Crossbars based on RRAM devices have distinct similarity to PCM crossbars. As a result, *in situ* learning has been proposed in RRAM crossbar arrays for single layer neural networks [180, 245]. RRAM and PCM-based crossbars have been further scaled for supervised learning to demonstrate deeper networks [9, 32, 102, 126, 176, 247]. Although such

demonstrations have been limited to ANN workloads, it can be used to map SNNs trained using supervised learning algorithms. Spintronic/magnetic tunnel junction-based crossbars, however, have not been widely demonstrated due to very low ON/OFF resistance ratios due to fabrication challenges. A lower ON/OFF ratio can pose limitations on application accuracy. Simulation studies have explored STDP-based learning [204] at an array level based on higher predictive ON/OFF ratios [80], which has been closely matched through experimental demonstration [86].

4.7 Hardware-software Co-design Approaches

In this section, we have described different methodologies for designing custom SNN hardware at various levels of the hardware stack such as architectures, communications, and finally compute primitives. Researchers have also explored effective hardware-software co-optimization techniques to efficiently map SNNs to the hardware systems. One of the approaches include Bayesian hyper-parameter optimization [178], where the authors have performed a grid search on hardware-specific hyperparameters. The authors of Reference [163] explored an architectural solution that we have described earlier, which required each layer of the SNN model to be computed through all the timesteps before progressing to the next layer—a prime example of hardware-aware optimization of SNN models for achieving compute efficiency. Transforming SNN models to achieve better amenability to neuromorphic hardware constraints has also been explored [95] to achieve high resource utilization. Efficient mapping of SNN models on the non-volatile memory-based SNN accelerators has been explored in Reference [11]. Hardware-Software co-optimization is at a fairly nascent stage in the neuromorphic domain. Researchers can draw motivation from corresponding deep learning hardware-software co-optimization techniques such as utilizing pruning, quantization as well as exploring models that can enable parameter re-use. In addition, as we scale SNN models, we should continue addressing unique challenges faced by neuromorphic systems as well such as additional overhead of the membrane potential storage and how re-use and quantization can help in that regard.

5 DISCUSSION

Artificial intelligence has become ubiquitous in diverse fields and is transforming the world around us. The current powerful machine learning models are mostly deployed on large cloud-computing systems. To enable large-scale adoption of edge intelligence or TinyML on IoT devices, there is a need to rethink the prevailing solutions. To that effect, techniques such as model compression, pruning, and quantization in ANNs have shown significant promise, but to achieve brain-like efficiency that may not be enough. Brain-inspired neuromorphic computing, specifically SNNs, can assist in bridging the energy gap. SNNs operate as dynamical systems with temporally evolving quantities that define the dynamics of the neurons and synapses. Although recent gradient-based methods consider the spike times for parameter updates, the backpropagation through time is memory-intensive. Therefore, further research is needed to develop learning algorithms that can efficiently employ the rich timing information to achieve faster learning with fewer resources.

As discussed in Section 2.4 and Figure 9, SNNs can behave as RNNs due to their inherent recurrence. This provides a unique opportunity to employ SNNs for both static and sequential tasks. Research, so far, is mainly focused on using SNNs for static image classification. Other tasks such as language modeling, speech recognition, machine translation, and so on, where RNNs have performed well, may be explored for SNNs. In Section 2.1, we discussed the IF/LIF neuron model whose simplicity is its strength, but further research in neuron modeling, structural plasticity, and the role of dendrites in efficient learning is required to mimic the complex dynamics of the brain. Also, the individual advancement in neuron modeling and learning algorithms may not result in a compatible solution. The focus should be on the co-design of neuron models and learning

algorithms that can achieve an optimal tradeoff between complexity and trainability. Similarly, the backpropagation-based algorithms may not be very suitable for deep SNNs, and the various hardware-friendly local learning methods may be more apt for performing computations on edge. Unsupervised STDP-based methods work well on shallow networks for simple tasks but fail to optimize deep networks. Other variants of STDP learning in combination with homeostasis and local gradient-based techniques may be explored to discover better learning mechanisms. Batch Normalization [91] has proven to be a successful technique in training deep ANNs, but its application in SNNs is limited and has not resulted in significant improvements. Moreover, edge applications require real-time online learning (batch size = 1), which raises the question of whether batch normalization can be done. SNNs are highly successful in handling spatiotemporal data from event sensors. They perform relatively better than ANNs at tasks such as motion estimation and classifying images from neuromorphic datasets. Therefore, further research is needed to identify more such tasks or efficiently convert other tasks in a discrete form that is more suitable for SNNs.

In Section 4, we have described the fundamental requirements as well as recent explorations toward building neuromorphic emulators and accelerators. The advancements in the field of neuromorphic hardware has closed the gap that exists between the algorithm space and their amenability in today's general-purpose as well as domain-specific accelerators. Further, the research in neuromorphic hardware leverages the unique features of SNN workloads that can potentially lead to low computational complexity as well as energy-efficiency. Despite the development in the space of neuromorphic hardware, significant evolution is required to truly realize the potential of energy-efficiency offered by SNN workloads. Neuromorphic algorithms are constantly evolving, as new features are being incorporated in the workloads, necessitating neuromorphic hardware to evolve as well. Over the past two decades, we have overseen massive leaps in building analog neuromorphic circuits in CMOS as well mapping such functionalities directly in single NVM devices. The primary challenge toward building efficient neuromorphic systems is scalability. We have delved into large-scale CMOS neuromorhic systems, which include wafer-scale integration as well as multi-chip modules. However, scaling NVM-based neuromorphic systems remains a challenge. First, the device variability and reliability poses a big challenge in realizing scalable systems using NVM-based neural primitives. Second, NVM-based compute primitives perform the synaptic computations in the analog domain. Analog computing in NVM-based primitives [37] is erroneous in nature and requires modeling and sufficient mitigation [210] for reasonably accurate operation. Various algorithmic strategies have been explored to mitigate and potentially leverage device mismatch and variability in a beneficial sense. For example, stochasticity in synapses [222] and neurons [197] have been shown to be beneficial towards generalization in SNNs.

In spite of significant efforts on building large-scale neuromorphic systems in CMOS, one critical aspect of such systems is implementing neuro-synaptic functions using analog circuits. Analog circuits are prone to threshold voltage variations across transistors. Although various electrical engineering techniques can be used to minimize device variations, analog design of silicon neurons and synapses requires additional circuitry for incorporating various kinds adaptation and feedback mechanisms. There have been explorations in devising techniques to counteract such variations in analog circuits [164]. Researchers also argue that a certain degree of imprecision is often beneficial to neural computing, drawing analogies with imprecise and diverse computing patterns in the biological brain [136]. Further, device mismatch has also been demonstrated [149] to enable stable receptive fields and balanced network activity in recurrent SNNs. Although there is significant merit to such arguments, it is also necessary to devise engineering solutions to circumvent the effect of device variations in large-scale neuromorphic systems, especially as CMOS technology has scaled down to below 10 nm.

Last, architecting neuromorphic hardware has to consider workload-related overheads. While there have been significant developments in designing event-driven hardware, SNN workloads are associated with additional data movements and storage due to an extra data-structure, which is the membrane potential, unlike ANN workloads. Architectures exploring non-volatile memory-based neurons can also allow *in situ* membrane potential storage and update [216], thereby reducing movement of membrane potential. Further optimizations can be considered to minimize the cost of fetching and updating membrane potential at every timestep. One such technique can involve quantization of membrane potential to lower than full-precision. This would reduce the storage requirements as well as data movement costs. Alternatively, one could explore stationarity options between inputs, weights, partial sums, and membrane potential and design a dataflow that reduces the movement of the most critical data structure. Overall, reducing membrane potential overhead remains a key design challenge in future neuromorphic hardware.

Neuromorphic computing has evolved significantly since its inception in the '90s, aided by developments in the field of neuroscience as well as semiconductor technology. With our current understanding of the functionalities of the biological brain, we have harnessed some key features such as event-based computation and communication, localized learning, as well as co-location of memory and processing capabilities. In addition, we have drawn inspiration from the development of AI algorithms to scale up the networks and achieve comparable performance in state-of-the-art recognition tasks compared to conventional ANNs while potentially reducing the power consumption. Along with the momentous progress, neuromorphic computing presents us with numerous challenges mentioned above that will shape the way for future research. Such efforts across the stack of sensors, hardware, and algorithms can truly help achieve efficient intelligent systems based on neuromorphic computing.

REFERENCES

[1] 2016. International roadmap for devices and systems (IRDS). *IEEE*. Retrieved from https://irds.ieee.org.

[2] L. Abbott, Brian DePasquale, and Raoul-Martin Memmesheimer. 2016. Building functional networks of spiking model neurons. *Nature Neurosci.* 19 (02 2016), 350–355. DOI: https://doi.org/10.1038/nn.4241

[3] Larry F. Abbott and Sacha B. Nelson. 2000. Synaptic plasticity: Taming the beast. *Nature Neurosci.* 3, 11 (2000), 1178–1183.

[4] Amogh Agrawal, Mustafa Ali, Minsuk Koo, Nitin Rathi, Akhilesh Jaiswal, and Kaushik Roy. 2021. IMPULSE: A 65-nm digital compute-in-memory macro with fused weights and membrane potential for spike-based sequential learning tasks. *IEEE Solid-state Circ. Lett.* 4 (2021), 137–140.

[5] Amogh Agrawal, Aayush Ankit, and Kaushik Roy. 2018. SPARE: Spiking neural network acceleration using rom-embedded RAMs as in-memory-computation primitives. *IEEE Trans. Comput.* 68, 8 (2018), 1190–1200.

[6] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam et al. 2015. TrueNorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 34, 10 (2015), 1537–1557.

[7] Alireza Alemi, Christian K. Machens, Sophie Denève, and Jean-Jacques E. Slotine. 2018. Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. In *32nd AAAI Conference on Artificial Intelligence*. AAAI Press, 588–595. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17438.

[8] Stefano Ambrogio, Nicola Ciocchini, Mario Laudato, Valerio Milo, Agostino Pirovano, Paolo Fantini, and Daniele Ielmini. 2016. Unsupervised learning by spike timing dependent plasticity in phase change memory (PCM) synapses. *Front. Neurosci.* 10 (2016), 56.

[9] Stefano Ambrogio, Pritish Narayanan, Hsinyu Tsai, Robert M. Shelby, Irem Boybat, Carmelo Nolfo, Severin Sidler, Massimo Giordano, Martina Bodini, Nathan C. P. Farinha et al. 2018. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* 558, 7708 (2018), 60.

[10] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. 2017. A low power, fully event-based gesture recognition system. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7388–7397. DOI: https://doi.org/10.1109/CVPR.2017.781

[11] Aayush Ankit, Abhronil Sengupta, Priyadarshini Panda, and Kaushik Roy. 2017. RESPARC: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks. In *54th Annual Design Automation Conference*. ACM, 27.

[12] John V. Arthur and Kwabena Boahen. 2004. Recurrently connected silicon neurons with active dendrites for one-shot learning. In *IEEE International Joint Conference on Neural Networks*. IEEE, 1699–1704.

[13] Richard C. Atkinson and Richard M. Shiffrin. 1968. Human memory: A proposed system and its control processes. In *Psychology of Learning and Motivation*, Vol. 2. Elsevier, 89–195.

[14] Alireza Bagheri, Osvaldo Simeone, and Bipin Rajendran. 2018. Adversarial training for probabilistic spiking neural networks. In *IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 1–5.

[15] Chiara Bartolozzi and Giacomo Indiveri. 2007. Synaptic dynamics in analog VLSI. *Neural Computat.* 19, 10 (2007), 2581–2603.

[16] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. In *International Conference on Advances in Neural Information Processing Systems*. 787–797.

[17] Guillaume Bellec, Franz Scherr, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. 2019. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049* (2019).

[18] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. 2020. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Commun.* 11, 1 (July 2020), 3625. DOI: https://doi.org/10.1038/s41467-020-17236-y

[19] Marc Benayoun, Jack D. Cowan, Wim van Drongelen, and Edward Wallace. 2010. Avalanches in a stochastic model of spiking neurons. *PLoS Computat. Biol.* 6, 7 (2010), e1000846.

[20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 5, 2 (1994), 157–166.

[21] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V. Arthur, Paul A. Merolla, and Kwabena Boahen. 2014. NeuroGrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 5 (2014), 699–716.

[22] R. Benosman, C. Clercq, X. Lagorce, S. Ieng, and C. Bartolozzi. 2014. Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 2 (2014), 407–417. DOI: https://doi.org/10.1109/TNNLS.2013.2273537

[23] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. 2012. Asynchronous frameless event-based optical flow. *Neural Netw.* 27 (2012), 32–37.

[24] Roberto Bez. 2009. Chalcogenide PCM: A memory technology for next decade. In *IEEE International Electron Devices Meeting*. IEEE, 1–4.

[25] Guo-qiang Bi and Mu-ming Poo. 2001. Synaptic modification by correlated activity: Hebb's postulate revisited. *Ann. Rev. Neurosci.* 24, 1 (2001), 139–166.

[26] Olivier Bichler, Manan Suri, Damien Querlioz, Dominique Vuillaume, Barbara DeSalvo, and Christian Gamrat. 2012. Visual pattern extraction using energy-efficient "2-PCM synapse" neuromorphic architecture. *IEEE Trans. Electron Dev.* 59, 8 (2012), 2206–2214.

[27] Kwabena A. Boahen. 2000. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans. Circ. Syst. II: Analog Digit. Sig. Process.* 47, 5 (2000), 416–434.

[28] Irem Boybat, Manuel Le Gallo, S. R. Nandakumar, Timoleon Moraitis, Thomas Parnell, Tomas Tuma, Bipin Rajendran, Yusuf Leblebici, Abu Sebastian, and Evangelos Eleftheriou. 2018. Neuromorphic computing with multi-memristive synapses. *Nature Commun.* 9, 1 (2018), 2514.

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

[30] Karla Burelo, Mohammadali Sharifshazileh, Niklaus Krayenbühl, Georgia Ramantani, Giacomo Indiveri, and Johannes Sarnthein. 2021. A spiking neural network (SNN) for detecting high frequency oscillations (HFOs) in the intraoperative ECoG. *Sci. Rep.* 11, 1 (2021), 1–10.

[31] Geoffrey W. Burr, Robert M. Shelby, Abu Sebastian, Sangbum Kim, Seyoung Kim, Severin Sidler, Kumar Virwani, Masatoshi Ishii, Pritish Narayanan, Alessandro Fumarola et al. 2017. Neuromorphic computing using non-volatile memory. *Adv. Phys. X* 2, 1 (2017), 89–124.

[32] Fuxi Cai et al. 2019. A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. *Nature Electron.* 2, 7 (2019), 290–299.

[33] Luis Camunas-Mesa, Antonio Acosta-Jiménez, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. 2008. Fully digital AER convolution chip for vision processing. In *IEEE International Symposium on Circuits and Systems*. IEEE, 652–655.

[34] Yongqiang Cao, Yang Chen, and Deepak Khosla. 2015. Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* 113, 1 (2015), 54–66.
[35] Snaider Carrillo, Jim Harkin, L. J. McDaid, Sandeep Pande, Seamus Cawley, Brian McGinley, and Fearghal Morgan. 2012. Hierarchical network-on-chip and traffic compression for spiking neural network implementations. In *IEEE/ACM 6th International Symposium on Networks-on-Chip*. IEEE, 83–90.
[36] Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In *International Workshop on Spoken Language Translation*.
[37] I. Chakraborty, A. Jaiswal, A. K. Saha, S. K. Gupta, and K. Roy. 2020. Pathways to efficient neuromorphic computing with non-volatile memory technologies. *Appl. Phys. Rev.* 7, 2 (2020), 021308.
[38] Indranil Chakraborty, Gobinda Saha, and Kaushik Roy. 2019. Photonic in-memory computing primitive for spiking neural networks using phase-change materials. *Phys. Rev. Appl.* 11, 1 (2019), 014063.
[39] Indranil Chakraborty, Gobinda Saha, Abhronil Sengupta, and Kaushik Roy. 2018. Toward fast neural computing using all-photonic phase change spiking neurons. *Sci. Rep.* 8, 1 (2018), 12980.
[40] Zengguang Cheng, Carlos Ríos, Wolfram H. P. Pernice, C. David Wright, and Harish Bhaskaran. 2017. On-chip photonic synapse. *Sci. Adv.* 3, 9 (2017), e1700160.
[41] Elisabetta Chicca, Fabio Stefanini, Chiara Bartolozzi, and Giacomo Indiveri. 2014. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proc. IEEE* 102, 9 (2014), 1367–1388.
[42] Leon Chua. 1971. Memristor-the missing circuit element. *IEEE Trans. Circ. Theor.* 18, 5 (1971), 507–519.
[43] Wonil Chung, Mengwei Si, and D. Ye Peide. 2018. First demonstration of Ge ferroelectric nanowire FET as synaptic device for online learning in neural network with high number of conductance state and G max/G min. In *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 15–2.
[44] Gregory Cohen, Saeed Afshar, Brittany Morreale, Travis Bessell, Andrew Wabnitz, Mark Rutten, and André van Schaik. 2019. Event-based sensing for space situational awareness. *J. Astronaut. Sci.* 66 (01 2019). DOI: https://doi.org/10.1007/s40295-018-00140-5
[45] Mike Davies et al. 2021. Taking neuromorphic computing to the next level with Loihi 2. In *Intel Newsroom Technology brief*. Retrieved from https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf.
[46] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 1 (2018), 82–99.
[47] Peter Dayan and Laurence F. Abbott. 2005. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. MIT Press.
[48] Arnaud Delorme et al. 2001. Networks of integrate-and-fire neurons using Rank Order Coding B: Spike timing dependent plasticity and emergence of orientation selectivity. *Neurocomputing* 38 (2001), 539–545.
[49] Yiğit Demirağ, Filippo Moro, Thomas Dalgaty, Gabriele Navarro, Charlotte Frenkel, Giacomo Indiveri, Elisa Vianello, and Melika Payvand. 2021. PCM-trace: Scalable synaptic eligibility traces with resistivity drift of phase-change materials. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[50] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 248–255.
[51] Brian DePasquale, Mark M. Churchland, and L. F. Abbott. 2016. Using firing-rate dynamics to train recurrent networks of spiking model neurons. *arXiv preprint arXiv:1601.07620* (2016).
[52] Peter U. Diehl and Matthew Cook. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Computat. Neurosci.* 9 (2015), 99.
[53] Peter U. Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*. IEEE, 1–8.
[54] Peter U. Diehl, Bruno U. Pedroni, Andrew Cassidy, Paul Merolla, Emre Neftci, and Guido Zarrella. 2016. TrueHappiness: Neuromorphic emotion recognition on TrueNorth. In *International Joint Conference on Neural Networks*. IEEE, 4278–4285.
[55] Elisa Donati, Melika Payvand, Nicoletta Risi, Renate Krause, and Giacomo Indiveri. 2019. Discrimination of EMG signals using a neuromorphic implementation of a spiking neural network. *IEEE Trans. Biomed. Circ. Syst.* 13, 5 (2019), 795–803.
[56] Sourav Dutta, Atanu K. Saha, Priyadarshini Panda, W. Chakraborty, J. Gomez, Abhishek Khanna, Sumeet Gupta, Kaushik Roy, and Suman Datta. 2019. Biologically plausible energy-efficient ferroelectric quasi-leaky integrate and fire neuron. In *Symposium on VLSI Technology*.
[57] Salah El Hihi and Yoshua Bengio. 1996. Hierarchical recurrent neural networks for long-term dependencies. In *International Conference on Advances in Neural Information Processing Systems*. 493–499.

[58] Sukru B. Eryilmaz, Duygu Kuzum, Rakesh Jeyasingh, SangBum Kim, Matthew BrightSky, Chung Lam, and H.-S. Philip Wong. 2014. Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Front. Neurosci.* 8 (2014), 205.

[59] S. Burc Eryilmaz, Duygu Kuzum, Rakesh G. D. Jeyasingh, SangBum Kim, Matthew BrightSky, Chung Lam, and H.-S. Philip Wong. 2013. Experimental demonstration of array-level learning with phase change synaptic devices. In *IEEE International Electron Devices Meeting*. IEEE, 25–5.

[60] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1625–1634.

[61] Ethan Farquhar and Paul Hasler. 2005. A bio-physically inspired silicon neuron. *IEEE Trans. Circ. Syst. I: Reg. Papers* 52, 3 (2005), 477–488.

[62] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice. 2019. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature* 569, 7755 (2019), 208.

[63] Paul Ferré, Franck Mamalet, and Simon J. Thorpe. 2018. Unsupervised feature learning with winner-takes-all based STDP. *Front. Computat. Neurosci.* 12 (2018), 24.

[64] Fopefolu Folowosele, Ralph Etienne-Cummings, and Tara Julia Hamilton. 2009. A CMOS switched capacitor implementation of the Mihalas-Niebur neuron. In *IEEE Biomedical Circuits and Systems Conference*. IEEE, 105–108.

[65] Xuanyao Fong, Yusung Kim, Karthik Yogendra, Deliang Fan, Abhronil Sengupta, Anand Raghunathan, and Kaushik Roy. 2016. Spin-transfer torque devices for logic and memory: Prospects and perspectives. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 35, 1 (2016), 1–22.

[66] Nicolas Frémaux and Wulfram Gerstner. 2016. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circ.* 9 (2016), 85.

[67] Max Garagnani, Guglielmo Lucchese, Rosario Tomasello, Thomas Wennekers, and Friedemann Pulvermüller. 2017. A spiking neurocomputational model of high-frequency oscillatory brain responses to words and pseudowords. *Front. Computat. Neurosci.* 10 (2017), 145.

[68] Nikhil Garg, Ismael Balafrej, Yann Beilliard, Dominique Drouin, Fabien Alibart, and Jean Rouat. 2021. Signals to spikes for neuromorphic regulated reservoir computing and EMG hand gesture recognition. In *International Conference on Neuromorphic Systems*. 1–8.

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

[70] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. 2014. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press.

[71] Aditya Gilra and Wulfram Gerstner. 2017. Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *eLife* 6 (Nov. 2017), e28295. DOI: https://doi.org/10.7554/eLife.28295

[72] Ludovic Goux, Piotr Czarnecki, Yang Yin Chen, Luigi Pantisano, XinPeng Wang, Robin Degraeve, Bogdan Govoreanu, Malgorzata Jurczak, D. J. Wouters, and Laith Altimime. 2010. Evidences of oxygen-mediated resistive-switching mechanism in TiN\HfO 2\Pt cells. *Appl. Phys. Lett.* 97, 24 (2010), 243509.

[73] Michael Graupner and Nicolas Brunel. 2012. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proc. Nat. Acad. Sci.* 109, 10 (2012), 3991–3996.

[74] Ankur Gupta and Lyle N. Long. 2007. Character recognition using spiking neural networks. In *International Joint Conference on Neural Networks*. IEEE, 53–58.

[75] G. Haessig, A. Cassidy, R. Alvarez, R. Benosman, and G. Orchard. 2018. Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system. *IEEE Trans. Biomed. Circ. Syst.* 12, 4 (2018), 860–870. DOI: https://doi.org/10.1109/TBCAS.2018.2834558

[76] Bing Han and Kaushik Roy. 2020. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*.

[77] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. RMP-SNNs: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[78] Paul E. Hasler, Chris Diorio, Bradley A. Minch, and Carver Mead. 1995. Single transistor learning synapses. In *International Conference on Advances in Neural Information Processing Systems*. 817–824.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[80] Atsufumi Hirohata, Hiroaki Sukegawa, Hideto Yanagihara, Igor Žutić, Takeshi Seki, Shigemi Mizukami, and Raja Swaminathan. 2015. Roadmap for emerging materials for Spintronic device applications. *IEEE Trans. Magnet.* 51, 10 (2015), 1–11.

[81] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber et al. 2001. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press.
[82] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computat.* 9, 8 (1997), 1735–1780.
[83] Alan L. Hodgkin and Andrew F. Huxley. 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* 117, 4 (1952), 500–544.
[84] Miao Hu, Yiran Chen, J. Joshua Yang, Yu Wang, and Hai Helen Li. 2016. A compact memristor-based dynamic synapse for spiking neural networks. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 36, 8 (2016), 1353–1366.
[85] Eric Hunsberger and Chris Eliasmith. 2015. Spiking deep networks with LIF neurons. *arXiv preprint arXiv:1510.08829* (2015).
[86] S. Ikeda, J. Hayakawa, Y. Ashizawa, Y. M. Lee, K. Miura, H. Hasegawa, M. Tsunoda, F. Matsukura, and H. Ohno. 2008. Tunnel magnetoresistance of 604% at 300 K by suppression of Ta diffusion in Co Fe B/ Mg O/ Co Fe B pseudo-spin-valves annealed at high temperature. *Appl. Phys. Lett.* 93, 8 (2008), 082508.
[87] Giacomo Indiveri. 2000. Modeling selective attention using a neuromorphic analog VLSI device. *Neural Computat.* 12, 12 (2000), 2857–2880.
[88] Giacomo Indiveri. 2021. Introducing "neuromorphic computing and engineering." *Neuromorph. Comput. Eng.* 1, 1 (2021), 010401.
[89] Giacomo Indiveri, Elisabetta Chicca, and Rodney J. Douglas. 2006. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 1 (2006).
[90] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André Van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud et al. 2011. Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5 (2011), 73.
[91] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).
[92] Eugene M. Izhikevich. 2003. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 6 (2003), 1569–1572.
[93] Akhilesh Jaiswal, Sourjya Roy, Gopalakrishnan Srinivasan, and Kaushik Roy. 2017. Proposal for a leaky-integrate-fire spiking neuron based on magnetoelectric switching of ferromagnets. *IEEE Trans. Electron Dev.* 64, 4 (2017), 1818–1824.
[94] Matthew Jerry, Pai-Yu Chen, Jianchi Zhang, Pankaj Sharma, Kai Ni, Shimeng Yu, and Suman Datta. 2017. Ferroelectric FET analog synapse for acceleration of deep neural network training. In *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 6–2.
[95] Yu Ji, YouHui Zhang, ShuangChen Li, Ping Chi, CiHang Jiang, Peng Qu, Yuan Xie, and WenGuang Chen. 2016. NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints. In *49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–13. DOI: https://doi.org/10.1109/MICRO.2016.7783724
[96] Yingyezhe Jin, Wenrui Zhang, and Peng Li. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 7005–7015.
[97] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B. Bhadviya, Pinaki Mazumder, and Wei Lu. 2010. Nanoscale memristor device as synapse in neuromorphic systems. *Nano Lett.* 10, 4 (2010), 1297–1301.
[98] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 1–12.
[99] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. 2020. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci.* 14 (2020), 424.
[100] Georg B. Keller and Thomas D. Mrsic-Flogel. 2018. Predictive processing: A canonical cortical computation. *Neuron* 100, 2 (2018), 424–435. DOI: https://doi.org/10.1016/j.neuron.2018.10.003
[101] Richard Kempter, Wulfram Gerstner, and J. Leo Van Hemmen. 1999. Hebbian learning and spiking neurons. *Phys. Rev. E* 59, 4 (1999), 4498.
[102] Riduan Khaddam-Aljameh, Milos Stanisavljevic, Jordi Fornt Mas, Geethan Karunaratne, Matthias Brändli, Feng Liu, Abhairaj Singh, Silvia M. Müller, Urs Egger, Anastasios Petropoulos et al. 2022. HERMES-Core–A 1.59-TOPS/mm 2 PCM on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs. *IEEE J. Solid-state Circ.* 57, 4 (2022), 1027–1038.
[103] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. 2018. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Netw.* 99 (2018), 56–67.
[104] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. W. Burr, N. Sosa, A. Ray et al. 2015. NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. In *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 17–1.
[105] Isabell Kiral-Kornek, Dulini Mendis, Ewan S. Nurse, Benjamin S. Mashford, Dean R. Freestone, David B. Grayden, and Stefan Harrer. 2017. TrueNorth-enabled real-time classification of EEG data for brain-computer interfacing. In *39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 1648–1651.

[106] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. Citeseer.
[107] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *International Conference on Advances in Neural Information Processing Systems*. 1097–1105.
[108] Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. 2020. Efficient processing of spatio-temporal data streams with spiking neural networks. *Front. Neurosci.* 14 (2020), 439.
[109] Kaushalya Kumarasinghe, Nikola Kasabov, and Denise Taylor. 2021. Brain-inspired spiking neural networks for decoding and understanding muscle activity and kinematics from electroencephalography signals during hand movements. *Sci. Rep.* 11, 1 (2021), 1–15.
[110] Alexey Kurakin, Ian Goodfellow, Samy Bengio et al. 2018. Adversarial examples in the physical world. *Artificial Intelligence Safety and Security*. Chapman and Hall/CRC, 99–112.
[111] Duygu Kuzum, Rakesh G. D. Jeyasingh, Byoungil Lee, and H.-S. Philip Wong. 2011. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* 12, 5 (2011), 2179–2186.
[112] Duygu Kuzum, Shimeng Yu, and H. S. Philip Wong. 2013. Synaptic electronics: Materials, devices and applications. *Nanotechnology* 24, 38 (2013), 382001.
[113] Raphael Lamprecht and Joseph LeDoux. 2004. Structural plasticity and memory. *Nature Rev. Neurosci.* 5, 1 (2004), 45.
[114] S. Lashkare, S. Chouhan, T. Chavan, A. Bhat, P. Kumbhare, and U. Ganguly. 2018. PCMO RRAM for integrate-and-fire neuron in spiking neural networks. *IEEE Electron Dev. Lett.* 39, 4 (2018), 484–487.
[115] John Lazzaro and John Wawrzynek. 1994. Low-power silicon neurons, axons and synapses. In *Silicon Implementation of Pulse Coded Neural Networks*. Springer, 153–164.
[116] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.
[117] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[118] Chankyu Lee, Adarsh Kumar Kosta, Alex Zihao Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. 2020. Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks. In *Computer Vision – ECCV 2020*. Springer International Publishing, Cham, 366–382.
[119] Chankyu Lee, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. 2018. Training deep spiking convolutional neural networks with STDP-based unsupervised pre-training followed by supervised fine-tuning. *Front. Neurosci.* 12 (2018), 435.
[120] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. 2020. Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci.* 14 (2020).
[121] C. Lee, G. Srinivasan, P. Panda, and K. Roy. 2018. Deep spiking convolutional neural network trained with unsupervised spike timing dependent plasticity. *IEEE Trans. Cog. Devel. Syst.* (2018), 1–1.
[122] Dongsoo Lee and Kaushik Roy. 2013. Area efficient ROM-embedded SRAM cache. *IEEE Transactions on Cognitive and Developmental Systems* 11, 3 (2018), 384–394.
[123] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. 2016. Training deep spiking neural networks using backpropagation. *Front. Neurosci.* 10 (2016), 508.
[124] A. S. Lele, Y. Fang, J. Ting, and A. Raychowdhury. 2020. Learning to walk: Spike based reinforcement learning for hexapod robot central pattern generation. In *2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. 208–212. DOI: https://doi.org/10.1109/AICAS48895.2020.9073987
[125] R. Gary Leonard and George Doddington. 1993. Tidigits speech corpus. In *Texas Instruments, Inc.* Linguistic Data Consortium.
[126] Can Li, Miao Hu, Yunning Li, Hao Jiang, Ning Ge, Eric Montgomery, Jiaming Zhang, Wenhao Song, Noraica Dávila, Catherine E. Graves et al. 2018. Analogue signal and image processing with large memristor crossbars. *Nature Electron.* 1, 1 (2018), 52.
[127] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. 2017. CIFAR10-DVS: An event-stream dataset for object classification. *Front. Neurosci.* 11 (2017), 309.
[128] Ling Liang, Xing Hu, Lei Deng, Yujie Wu, Guoqi Li, Yufei Ding, Peng Li, and Yuan Xie. 2020. Exploring adversarial attack in spiking neural networks with spike-compatible gradient. *arXiv preprint arXiv:2001.01587* (2020).
[129] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. 2008. A 128× 128 120 dB 15μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-state Circ.* 43, 2 (2008), 566–576.
[130] S. Liu, B. Rueckauer, E. Ceolini, A. Huber, and T. Delbruck. 2019. Event-driven sensing for efficient perception: Vision and audition algorithms. *IEEE Sig. Process. Mag.* 36, 6 (2019), 29–37. DOI: https://doi.org/10.1109/MSP.2019.2928127
[131] Ali Lotfi Rezaabad and Sriram Vishwanath. 2020. Long short-term memory spiking networks and their applications. In *International Conference on Neuromorphic Systems 2020*. 1–9.
[132] Sen Lu and Abhronil Sengupta. 2020. Exploring the connection between binary and spiking neural networks. *arXiv preprint arXiv:2002.10064* (2020).

[133] Wolfgang Maass. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* 10, 9 (1997), 1659–1671.
[134] Wolfgang Maass. 2011. Liquid state machines: Motivation, theory, and applications. In *Computability in Context: Computation and Logic in the Real World*. World Scientific, 275–296.
[135] Wolfgang Maass and Henry Markram. 2004. On the computational power of circuits of spiking neurons. *J. Comput. Syst. Sci.* 69, 4 (2004), 593–616.
[136] Wolfgang Maass, Thomas Natschläger, and Henry Markram. 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computat.* 14, 11 (2002), 2531–2560.
[137] Misha Mahowald. 1994. *An Analog VLSI System for Stereoscopic Vision*, Vol. 265. Springer Science & Business Media.
[138] Misha Mahowald. 1994. The silicon retina. In *An Analog VLSI System for Stereoscopic Vision*. Springer, 4–65.
[139] Misha Mahowald and Rodney Douglas. 1991. A silicon neuron. *Nature* 354, 6354 (1991), 515.
[140] Stephen J. Martin, Paul D. Grimwood, and Richard G. M. Morris. 2000. Synaptic plasticity and memory: An evaluation of the hypothesis. *Ann. Rev. Neurosci.* 23, 1 (2000), 649–711.
[141] Timothée Masquelier and Simon J. Thorpe. 2007. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computat. Biol.* 3, 2 (2007), e31.
[142] Stephen J. Maybank, Sio-Hoi Ieng, Davide Migliore, and Ryad Benosman. 2021. Optical flow estimation using the Fisher–Rao metric. *Neuromorph. Comput. Eng.* 1, 2 (2021), 024004.
[143] Carver Mead. 1990. Neuromorphic electronic systems. *Proc. IEEE* 78, 10 (1990), 1629–1636.
[144] Carver Mead. 2020. How we created neuromorphic engineering. *Nature Electron.* 3, 7 (2020), 434–435.
[145] Carver Mead and Mohammed Ismail. 2012. *Analog VLSI Implementation of Neural Systems*, Vol. 80. Springer Science & Business Media.
[146] Adnan Mehonic and Anthony J. Kenyon. 2016. Emulating the electrical activity of the neuron using a silicon oxide RRAM cell. *Front. Neurosci.* 10 (2016), 57.
[147] Paul Merolla, John Arthur, Rodrigo Alvarez, Jean-Marie Bussat, and Kwabena Boahen. 2014. A multicast tree router for multichip neuromorphic systems. *IEEE Trans. Circ. Syst. I: Reg. Papers* 61, 3 (2014), 820–833.
[148] Alexander Meulemans, Matilde Tristany Farinha, Javier Garcia Ordonez, Pau Vilimelis Aceituno, João Sacramento, and Benjamin F. Grewe. 2021. Credit assignment in neural networks through deep feedback control. *Adv. Neural Inf. Process. Syst.* 34 (2021).
[149] Moritz Benjamin Milde. 2019. *Spike-based Computational Primitives for Vision-based Scene Understanding*. Ph.D. Dissertation. University of Zurich.
[150] Moritz B. Milde, Olivier J. N. Bertrand, Harshawardhan Ramachandran, Martin Egelhaaf, and Elisabetta Chicca. 2018. Spiking elementary motion detector in neuromorphic systems. *Neural Computat* 30, 9 (2018), 2384–2417.
[151] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos. 2018. Event-based moving object detection and tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1–9. DOI: https://doi.org/10.1109/IROS.2018.8593805
[152] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *33rd International Conference on International Conference on Machine Learning (ICML'16)*. JMLR.org, 1928–1937.
[153] Saber Moradi, Nabil Imam, Rajit Manohar, and Giacomo Indiveri. 2013. A memory-efficient routing method for large-scale spiking neural networks. In *European Conference on Circuit Theory and Design (ECCTD)*. IEEE, 1–4.
[154] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. 2017. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Trans. Biomed. Circ. Syst.* 12, 1 (2017), 106–122.
[155] Hesham Mostafa. 2017. Supervised learning based on temporal coding in spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* (2017). *IEEE Transactions on Neural Networks and Learning Systems* 29, 7 (2017), 3227–3235.
[156] Hesham Mostafa, Bruno U. Pedroni, Sadique Sheik, and Gert Cauwenberghs. 2017. Fast classification using sparsely active spiking networks. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–4.
[157] Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. 2018. Deep supervised learning using local errors. *Front. Neurosci.* 12 (2018), 608.
[158] E. Mueggler, B. Huber, and D. Scaramuzza. 2014. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2761–2768. DOI: https://doi.org/10.1109/IROS.2014.6942940
[159] Halid Mulaosmanovic, Elisabetta Chicca, Martin Bertele, Thomas Mikolajick, and Stefan Slesazeck. 2018. Mimicking biological neurons with a nanoscale ferroelectric transistor. *Nanoscale* 10, 46 (2018), 21755–21763.
[160] Halid Mulaosmanovic, Thomas Mikolajick, and Stefan Slesazeck. 2018. Accumulative polarization reversal in nanoscale ferroelectric transistors. *ACS Appl. Mater. Interf.* 10, 28 (2018), 23997–24002.

[161] H. Mulaosmanovic, J. Ocker, S. Müller, M. Noack, J. Müller, P. Polakowski, T. Mikolajick, and S. Slesazeck. 2017. Novel ferroelectric FET based synapse for neuromorphic systems. In *Symposium on VLSI Technology*. IEEE, T176–T177.

[162] Nishant Mysore, Gopabandhu Hota, Stephen R. Deiss, Bruno Umbria Pedroni, and Gert Cauwenberghs. 2021. Hierarchical network connectivity and partitioning for reconfigurable large-scale neuromorphic systems. *Frontiers in Neuroscience* 1891, 15 (2021), 797654. DOI: 10.3389/fnins.2021.797654

[163] Surya Narayanan, Karl Taht, Rajeev Balasubramonian, Edouard Giacomin, and Pierre-Emmanuel Gaillardon. 2020. SpinalFlow: An architecture and dataflow tailored for spiking neural networks. In *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 349–362.

[164] Emre Neftci and Giacomo Indiveri. 2010. A device mismatch compensation method for VLSI neural networks. In *Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 262–265.

[165] Emre O. Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. 2017. Event-driven random backpropagation: Enabling neuromorphic deep learning machines. *Front. Neurosci.* 11 (2017), 324.

[166] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. 2019. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Sig. Process. Mag.* 36, 6 (2019), 51–63.

[167] Wilten Nicola and Claudia Clopath. 2017. Supervised learning in spiking neural networks with FORCE training. *Nature Commun.* 8, 1 (Dec. 2017), 2208. DOI: https://doi.org/10.1038/s41467-017-01827-3

[168] Ewan Nurse, Benjamin S. Mashford, Antonio Jimeno Yepes, Isabell Kiral-Kornek, Stefan Harrer, and Dean R. Freestone. 2016. Decoding EEG and LFP signals using deep learning: Heading TrueNorth. In *ACM International Conference on Computing Frontiers*. ACM, 259–266.

[169] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor. 2013. A spiking neural network architecture for visual motion estimation. In *IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 298–301. DOI: https://doi.org/10.1109/BioCAS.2013.6679698

[170] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. 2015. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* 9 (2015), 437.

[171] Eustace Painkras, Luis A. Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R. Lester, Andrew D. Brown, and Steve B. Furber. 2013. SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid-state Circ.* 48, 8 (2013), 1943–1953.

[172] Giorgio Palma, Manan Suri, Damien Querlioz, Elisa Vianello, and Barbara De Salvo. 2013. Stochastic neuron design using conductive bridge RAM. In *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 95–100.

[173] Priyadarshini Panda, Jason M. Allred, Shriram Ramanathan, and Kaushik Roy. 2018. ASP: Learning to forget with adaptive synaptic plasticity in spiking neural networks. *IEEE J. Emerg. Select. Topics. Circ. Syst.* 8, 1 (2018), 51–64.

[174] Priyadarshini Panda and Kaushik Roy. 2016. Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. In *International Joint Conference on Neural Networks*. IEEE, 299–306.

[175] Priyadarshini Panda and Narayan Srinivasa. 2017. Learning to recognize actions from limited training examples using a recurrent spiking neural model. *CoRR* abs/1710.07354 (2017). arXiv:1710.07354

[176] Angeliki Pantazi, Stanisław Woźniak, Tomas Tuma, and Evangelos Eleftheriou. 2016. All-memristive neuromorphic computing with level-tuned neurons. *Nanotechnology* 27, 35 (2016), 355205.

[177] F. Paredes-Vallés, Kirk Y. W. Scheper, and G. de Croon. 2020. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE Trans. Pattern Anal. Mach. Intell.* 42 (2020), 2051–2064.

[178] Maryam Parsa, J. Parker Mitchell, Catherine D. Schuman, Robert M. Patton, Thomas E. Potok, and Kaushik Roy. 2019. Bayesian-based hyperparameter optimization for spiking neuromorphic systems. In *IEEE International Conference on Big Data (Big Data)*. 4472–4478. DOI: https://doi.org/10.1109/BigData47090.2019.9006383

[179] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. 1310–1318.

[180] G. Pedretti, V. Milo, S. Ambrogio, R. Carboni, S. Bianchi, A. Calderoni, N. Ramaswamy, A. S. Spinelli, and D. Ielmini. 2017. Memristive neural network for on-line learning and tracking with brain-inspired spike timing dependent plasticity. *Sci. Rep.* 7, 1 (2017), 5288.

[181] Michael Pfeiffer and Thomas Pfeil. 2018. Deep learning with spiking neurons: Opportunities and challenges. *Front. Neurosci.* 12 (2018), 774.

[182] Wachirawit Ponghiran and Kaushik Roy. 2021. Hybrid analog-spiking long short-term memory for energy efficient computing on edge devices. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.

[183] Wachirawit Ponghiran, Gopalakrishnan Srinivasan, and Kaushik Roy. 2019. Reinforcement learning with low-complexity liquid state machines. *Front. Neurosci.* 13 (2019), 883. DOI: https://doi.org/10.3389/fnins.2019.00883

[184] Mirko Prezioso, F. Merrikh Bayat, Brian Hoskins, K. Likharev, and D. Strukov. 2016. Self-adaptive spike-time-dependent plasticity of metal-oxide memristors. *Sci. Rep.* 6 (2016), 21331.

[185] M. Prezioso, M. R. Mahmoodi, F. Merrikh Bayat, H. Nili, H. Kim, A. Vincent, and D. B. Strukov. 2018. Spike-timing-dependent plasticity learning of coincidence detection with passively integrated memristive circuits. *Nature Commun.* 9, 1 (2018), 1–8.

[186] Guo qiang Bi and Mu ming Poo. 1998. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* 18, 24 (Dec. 1998), 10464–10472. DOI: https://doi.org/10.1523/jneurosci.18-24-10464.1998

[187] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. 2015. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Front. Neurosci.* 9 (2015), 141.

[188] Damien Querlioz, Olivier Bichler, Adrien Francis Vincent, and Christian Gamrat. 2015. Bioinspired programming of memory devices for implementing an inference engine. *Proc. IEEE* 103, 8 (2015), 1398–1416.

[189] Bipin Rajendran, Yong Liu, Jae-sun Seo, Kailash Gopalakrishnan, Leland Chang, Daniel J. Friedman, and Mark B. Ritter. 2012. Specifications of nanoscale devices and circuits for neuromorphic computational systems. *IEEE Trans. Electron Dev.* 60, 1 (2012), 246–253.

[190] Shubha Ramakrishnan, Paul E. Hasler, and Christal Gordon. 2011. Floating gate synapses with spike-time-dependent plasticity. *IEEE Trans. Biomed. Circ. Syst.* 5, 3 (2011), 244–252.

[191] Venkat Rangan, Abhishek Ghosh, Vladimir Aparin, and Gert Cauwenberghs. 2010. A subthreshold aVLSI implementation of the Izhikevich simple neuron model. In *Annual International Conference of the IEEE Engineering in Medicine and Biology*. IEEE, 4164–4167.

[192] Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. 2018. STDP-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition. *IEEE Trans. Comput.-aid. Des. Integ. Circ. Syst.* 38, 4 (2018), 668–677.

[193] Nitin Rathi and Kaushik Roy. 2020. Diet-SNN: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658* (2020).

[194] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. 2020. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=B1xSperKvH.

[195] Christina Rohde, Byung Joon Choi, Doo Seok Jeong, Seol Choi, Jin-Shi Zhao, and Cheol Seong Hwang. 2005. Identification of a determining parameter for resistive switching of Ti O 2 thin films. *Appl. Phys. Lett.* 86, 26 (2005), 262907.

[196] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, Cham, 234–241.

[197] Deboleena Roy, Indranil Chakraborty, and Kaushik Roy. 2019. Scaling deep spiking neural networks with binary stochastic activations. In *IEEE International Conference on Cognitive Computing (ICCC)*. IEEE, 50–58.

[198] Bodo Rueckauer and Shih-Chii Liu. 2018. Conversion of analog to spiking neural networks using sparse temporal coding. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[199] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11 (2017), 682.

[200] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein et al. 2015. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.* 115, 3 (2015), 211–252.

[201] Sylvain Saïghi, Christian G. Mayr, Teresa Serrano-Gotarredona, Heidemarie Schmidt, Gwendal Lecerf, Jean Tomas, Julie Grollier, Sören Boyn, Adrien F. Vincent, Damien Querlioz et al. 2015. Plasticity in memristive devices for spiking neural networks. *Front. Neurosci.* 9 (2015), 51.

[202] Arash Samadi, Timothy P. Lillicrap, and Douglas B. Tweed. 2017. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural Computat.* 29, 3 (2017), 578–602.

[203] Johannes Schemmel, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. 2010. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1947–1950.

[204] Abhronil Sengupta, Aparajita Banerjee, and Kaushik Roy. 2016. Hybrid Spintronic-CMOS spiking neural network with on-chip learning: Devices, circuits, and systems. *Phys. Rev. Appl.* 6, 6 (2016), 064003.

[205] Abhronil Sengupta, Priyadarshini Panda, Parami Wijesinghe, Yusung Kim, and Kaushik Roy. 2016. Magnetic tunnel junction mimics stochastic cortical spiking neurons. *Sci. Rep.* 6 (2016), 30039.

[206] Abhronil Sengupta, Maryam Parsa, Bing Han, and Kaushik Roy. 2016. Probabilistic deep spiking neural systems enabled by magnetic tunnel junction. *IEEE Trans. Electron Dev.* 63, 7 (2016), 2963–2970.

[207] Abhronil Sengupta and Kaushik Roy. 2016. A vision for all-spin neural networks: A device to system perspective. *IEEE Trans. Circ. Syst. I: Reg. Papers* 63, 12 (2016), 2267–2277.

[208] Abhronil Sengupta and Kaushik Roy. 2017. Encoding neural and synaptic functionalities in electron spin: A pathway to efficient neuromorphic computing. *Appl. Phys. Rev.* 4, 4 (2017), 041105.

[209] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. 2019. Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* 13 (2019).

[210] Kyungah Seo, Insung Kim, Seungjae Jung, Minseok Jo, Sangsu Park, Jubong Park, Jungho Shin, Kuyyadi P. Biju, Jaemin Kong, Kwanghee Lee et al. 2011. Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device. *Nanotechnology* 22, 25 (2011), 254023.

[211] Rafael Serrano-Gotarredona, Teresa Serrano-Gotarredona, Antonio Acosta-Jimenez, and Bernab Linares-Barranco. 2006. A neuromorphic cortical-layer microchip for spike-based event processing vision systems. *IEEE Trans. Circ. Syst. I: Reg. Papers* 53, 12 (2006), 2548–2566.

[212] Saima Sharmin, Nitin Rathi, Priyadarshini Panda, and Kaushik Roy. 2020. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *European Conference on Computer Vision*. Springer, 399–414.

[213] Sumit Bam Shrestha and Garrick Orchard. 2018. SLAYER: Spike layer error reassignment in time. In *International Conference on Advances in Neural Information Processing Systems*. 1412–1421.

[214] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.

[215] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[216] S. Singh, A. Sharma, A. Pattnaik N. Jao, S. Lu, K. Yang, A. Sengupta, V. Narayanan, and C. Das. 2020. NEBULA: A neuromorphic spin-based ultra-low power architecture for SNNs and ANNs. In *IEEE/ACM International Symposium on Computer Architecture (ISCA'20)*. IEEE.

[217] B. Son, Y. Suh, S. Kim, H. Jung, J. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu. 2017. 4.1 A 640×480 dynamic vision sensor with a 9μm pixel and 300Meps address-event representation. In *IEEE International Solid-State Circuits Conference (ISSCC)*. 66–67. DOI: https://doi.org/10.1109/ISSCC.2017.7870263

[218] Sen Song, Kenneth D. Miller, and Larry F. Abbott. 2000. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neurosci.* 3, 9 (2000), 919.

[219] Nicholas Soures and Dhireesha Kudithipudi. 2019. Deep liquid state machines with neural plasticity for video activity recognition. *Front. Neurosci.* 13 (2019), 686. DOI: https://doi.org/10.3389/fnins.2019.00686

[220] Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. 2018. SpiLinC: Spiking liquid-ensemble computing for unsupervised speech and image recognition. *Front. Neurosci.* 12 (2018), 524. DOI: https://doi.org/10.3389/fnins.2018.00524

[221] Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. 2018. STDP-based unsupervised feature learning using convolution-over-time in spiking neural networks for energy-efficient neuromorphic computing. *ACM J. Emerg. Technol. Comput. Syst.* 14, 4 (2018), 1–12.

[222] Gopalakrishnan Srinivasan and Kaushik Roy. 2019. ReStoCNet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing. *Front. Neurosci.* 13 (2019), 189.

[223] Gopalakrishnan Srinivasan and Kaushik Roy. 2021. BlocTrain: Block-wise conditional training and inference for efficient spike-based deep learning. *Frontiers in Neuroscience* 1891, 15 (2021), 603433. DOI: 10.3389/fnins.2021.603433

[224] Gopalakrishnan Srinivasan, Abhronil Sengupta, and Kaushik Roy. 2016. Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip STDP learning. *Sci. Rep.* 6 (2016), 29545.

[225] Dmitri B. Strukov, Gregory S. Snider, Duncan R. Stewart, and R. Stanley Williams. 2008. The missing memristor found. *Nature* 453, 7191 (2008), 80.

[226] Manan Suri, Damien Querlioz, Olivier Bichler, Giorgio Palma, Elisa Vianello, Dominique Vuillaume, Christian Gamrat, and Barbara DeSalvo. 2013. Bio-inspired stochastic computing using binary CBRAM synapses. *IEEE Trans. Electron Dev.* 60, 7 (2013), 2402–2409.

[227] Ilya Sutskever. 2013. *Training Recurrent Neural Networks*. University of Toronto, Toronto, Ontario, Canada.

[228] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2818–2826.

[229] Aykut Tahtirvanci, Akif Durdu, and Burak Yilmaz. 2018. Classification of EEG signals using spiking neural networks. In *26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 1–4.

[230] Amirhossein Tavanaei, Timothée Masquelier, and Anthony S. Maida. 2016. Acquisition of visual features through probabilistic spike-timing-dependent plasticity. In *International Joint Conference on Neural Networks*. IEEE, 307–314.
[231] Chetan Singh Thakur, Jamal Lottier Molin, Gert Cauwenberghs, Giacomo Indiveri, Kundan Kumar, Ning Qiao, Johannes Schemmel, Runchun Wang, Elisabetta Chicca, Jennifer Olson Hasler et al. 2018. Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Front. Neurosci.* 12 (2018), 891.
[232] Johannes C. Thiele, Olivier Bichler, and Antoine Dupret. 2018. Event-based, timescale invariant unsupervised online deep learning with STDP. *Front. Computat. Neurosci.* 12 (2018), 46.
[233] Tomas Tuma, Manuel Le Gallo, Abu Sebastian, and Evangelos Eleftheriou. 2016. Detecting correlations using phase-change neurons and synapses. *IEEE Electron Dev. Lett.* 37, 9 (2016), 1238–1241.
[234] Tomas Tuma, Angeliki Pantazi, Manuel Le Gallo, Abu Sebastian, and Evangelos Eleftheriou. 2016. Stochastic phase-change neurons. *Nature Nanotechnol.* 11, 8 (2016), 693.
[235] Robert Urbanczik and Walter Senn. 2014. Learning by the dendritic prediction of somatic spiking. *Neuron* 81, 3 (2014), 521–528.
[236] André Van Schaik. 2001. Building blocks for electronic spiking neural networks. *Neural Netw.* 14, 6-7 (2001), 617–628.
[237] André van Schaik, Craig Jin, Alistair McEwan, and Tara Julia Hamilton. 2010. A log-domain implementation of the Izhikevich neuron model. In *IEEE International Symposium on Circuits and Systems*. IEEE, 4253–4256.
[238] Adrien F. Vincent, Jérôme Larroque, Nicolas Locatelli, Nesrine Ben Romdhane, Olivier Bichler, Christian Gamrat, Wei Sheng Zhao, Jacques-Olivier Klein, Sylvie Galdin-Retailleau, and Damien Querlioz. 2015. Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. *IEEE Trans. Biomed. Circ. Syst.* 9, 2 (2015), 166–174.
[239] R. Jacob Vogelstein, Udayan Mallik, Eugenio Culurciello, Gert Cauwenberghs, and Ralph Etienne-Cummings. 2007. A multichip neuromorphic system for spike-based visual information processing. *Neural Computat.* 19, 9 (2007), 2281–2300.
[240] R. Jacob Vogelstein, Udayan Mallik, Joshua T. Vogelstein, and Gert Cauwenberghs. 2007. Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw.* 18, 1 (2007), 253–265.
[241] John Von Neumann. 2012. *The Computer and the Brain*. Yale University Press.
[242] Edward Wallace, Marc Benayoun, Wim Van Drongelen, and Jack D. Cowan. 2011. Emergent oscillations in networks of stochastic spiking neurons. *PLoS One* 6, 5 (2011), e14804.
[243] I-Ting Wang, Yen-Chuan Lin, Yu-Fen Wang, Chung-Wei Hsu, and Tuo-Hung Hou. 2014. 3D synaptic architecture with ultralow sub-10 fJ energy per spike for neuromorphic computation. In *IEEE International Electron Devices Meeting*. IEEE, 28–5.
[244] Runchun Wang and André van Schaik. 2018. Breaking Liebig's law: An advanced multipurpose neuromorphic engine. *Front. Neurosci.* 12 (2018), 593.
[245] Yu Wang, Tianqi Tang, Lixue Xia, Boxun Li, Peng Gu, Huazhong Yang, Hai Li, and Yuan Xie. 2015. Energy efficient RRAM spiking neural network for real time classification. In *25th Great Lakes Symposium on VLSI*. ACM, 189–194.
[246] Zhongqiang Wang, Stefano Ambrogio, Simone Balatti, and Daniele Ielmini. 2015. A 2-transistor/1-resistor artificial synapse capable of communication and stochastic learning in neuromorphic systems. *Front. Neurosci.* 8 (2015), 438.
[247] Zhongrui Wang, Saumil Joshi, Sergey Savel'ev, Wenhao Song, Rivu Midya, Yunning Li, Mingyi Rao, Peng Yan, Shiva Asapu, Ye Zhuo et al. 2018. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electron.* 1, 2 (2018), 137.
[248] Yukio Watanabe, J. Ge Bednorz, A. Bietsch, Ch. Gerber, D. Widmer, A. Beck, and S. J. Wind. 2001. Current-driven insulator–conductor transition and nonvolatile memory in chromium-doped SrTiO 3 single crystals. *Appl. Phys. Lett.* 78, 23 (2001), 3738–3740.
[249] Jayawan H. B. Wijekoon and Piotr Dudek. 2008. Compact silicon neuron circuit with spiking and bursting behaviour. *Neural Netw.* 21, 2-3 (2008), 524–534.
[250] H.-S. Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P. Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E. Goodson. 2010. Phase change memory. *Proc. IEEE* 98, 12 (2010), 2201–2227.
[251] H.-S. Philip Wong and Sayeef Salahuddin. 2015. Memory leads the way to better computing. *Nature Nanotechnol.* 10, 3 (2015), 191.
[252] Stanisław Woźniak, Angeliki Pantazi, Thomas Bohnstingl, and Evangelos Eleftheriou. 2020. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Mach. Intell.* 2, 6 (2020), 325–336.
[253] Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. 2021. Training spiking neural networks with accumulated spiking flow. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 12 (2021), 10320–10328.
[254] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12 (2018).

[255] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. 2019. Direct training for spiking neural networks: Faster, larger, better. In *AAAI Conference on Artificial Intelligence*. 1311–1318.
[256] Yannan Xing, Gaetano Di Caterina, and John Soraghan. 2020. A new spiking convolutional recurrent neural network (SCRNN) with applications to event-based hand gesture recognition. *Front. Neurosci.* 14 (2020), 1143.
[257] Xiaowei Xu, Yukun Ding, Sharon Xiaobo Hu, Michael Niemier, Jason Cong, Yu Hu, and Yiyu Shi. 2018. Scaling for edge inference of deep neural networks. *Nature Electron.* 1, 4 (2018), 216.
[258] F. Xue, Hang Guan, and X. Li. 2016. Improving liquid state machine with hybrid plasticity. In *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. 1955–1959. DOI: https://doi.org/10.1109/IMCEC.2016.7867559
[259] Zhanglu Yan, Jun Zhou, and Weng-Fai Wong. 2021. Energy efficient ECG classification with spiking neural network. *Biomed. Sig. Process. Contr.* 63 (2021), 102170.
[260] Chengxi Ye, Anton Mitrokhin, Cornelia Fermüller, James A. Yorke, and Yiannis Aloimonos. 2019. Unsupervised Learning of Dense Optical Flow, Depth and Egomotion from Sparse Event Data. (2019). arXiv:cs.CV/1809.08625
[261] Bojian Yin, Federico Corradi, and Sander M. Bohté. 2021. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Mach. Intell.* 3, 10 (2021), 905–913.
[262] Shimeng Yu. 2018. Neuro-inspired computing with emerging nonvolatile memorys. *Proc. IEEE* 106, 2 (2018), 260–285.
[263] Theodore Yu and Gert Cauwenberghs. 2010. Analog VLSI biophysical neurons and synapses with programmable membrane channel kinetics. *IEEE Trans. Biomed. Circ. Syst.* 4, 3 (2010), 139–148.
[264] Friedemann Zenke and Surya Ganguli. 2018. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Computat.* 30, 6 (2018), 1514–1541.
[265] Friedemann Zenke and Emre O Neftci. 2021. Brain-inspired learning on neuromorphic substrates. *Proc. IEEE* 109, 5 (2021), 935–950.
[266] Deming Zhang, Lang Zeng, Youguang Zhang, Weisheng Zhao, and Jacques Olivier Klein. 2016. Stochastic Spintronic device based synapses and spiking neurons for neuromorphic computation. In *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, 173–178.
[267] Ming Zhang, Zonghua Gu, Nenggan Zheng, De Ma, and Gang Pan. 2020. Efficient spiking neural networks with logarithmic temporal coding. *IEEE Access* 8 (2020), 98156–98167.
[268] Wenrui Zhang and Peng Li. 2020. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *arXiv preprint arXiv:2002.10085* (2020).
[269] Yang Zhang, Zhongrui Wang, Jiadi Zhu, Yuchao Yang, Mingyi Rao, Wenhao Song, Ye Zhuo, Xumeng Zhang, Menglin Cui, Linlin Shen et al. 2020. Brain-inspired computing with memristors: Challenges in devices, circuits, and systems. *Appl. Phys. Rev.* 7, 1 (2020), 011308.
[270] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. 2018. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. *Robot.: Sci. Syst. XIV* (June 2018). DOI: https://doi.org/10.15607/rss.2018.xiv.062
[271] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. 2018. The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robot. Autom. Lett.* 3, 3 (2018), 2032–2039.
[272] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. 2019. Unsupervised event-based learning of optical flow, depth, and egomotion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 989–997. DOI: https://doi.org/10.1109/CVPR.2019.00108
[273] Mohammed A. Zidan, John Paul Strachan, and Wei D Lu. 2018. The future of electronics based on memristive systems. *Nature Electron.* 1, 1 (2018), 22.
[274] Robert S. Zucker and Wade G. Regehr. 2002. Short-term synaptic plasticity. *Ann. Rev. Physiol.* 64, 1 (2002), 355–405.
[275] Rui Zuo, Jing Wei, Xiaonan Li, Chunlin Li, Cui Zhao, Zhaohui Ren, Ying Liang, Xinling Geng, Chenxi Jiang, Xiaofeng Yang et al. 2019. Automated detection of high-frequency oscillations in epilepsy based on a convolutional neural network. *Front. Computat. Neurosci.* 13 (2019), 6.

Received 30 October 2021; revised 29 August 2022; accepted 25 October 2022