REVIEW

Review of spike-based neuromorphic computing for brain-inspired vision: biology, algorithms, and hardware

Hagar Hendy and Cory Merkel*
Rochester Institute of Technology, Brain Lab, Rochester, New York, United States

Abstract. Neuromorphic computing is becoming a popular approach for implementations of brain-inspired machine learning tasks. As a paradigm for both hardware and algorithm design, neuromorphic computing aims to emulate several aspects related to the structure and function of the biological nervous system to achieve artificial intelligence with efficiencies that are orders of magnitude better than those exhibited by general-purpose computing hardware. We provide a holistic treatment of spike-based neuromorphic computing (i.e., based on spiking neural networks), detailing biological motivation, key aspects of neuromorphic algorithms, and a survey of state-of-the-art neuromorphic hardware. In particular, we focus on these aspects within the context of brain-inspired vision applications. Our aim is to serve as a complement to several of the existing reviews on neuromorphic computing while also providing a unique perspective. © *2022 SPIE and IS&T* [DOI: 10.1117/1.JEI.31.1.010901]

Keywords: neuromorphic; brain-inspired computing; spiking neural networks; vision.

Paper 210593V received Sep. 3, 2021; accepted for publication Jan. 13, 2022; published online Jan. 31, 2022.

1 Introduction

Nature has long served as a source of inspiration and creativity in humans' technological advancement. Indeed, evolution has created ingenious solutions to a number of engineering challenges from the flight of birds, to the search strategies used by ant colonies, to the photosynthesis processes that fuel plants, and countless others. But, perhaps the most intriguing product of nature is evolution's solution to what we call intelligence: the human brain. Our brains are the gold standard against which we compare the capabilities of artificial intelligence (AI) systems, especially those based on machine learning (ML). Therefore, it should be no surprise that some of the most advanced ML algorithms, neural networks, are inspired by the brain's complex network of ∼86 billion neurons. Neural networks have made significant achievements (e.g., Krizhevsky et al.'s landmark paper[1] on ImageNet classification with convolutional neural networks) in the last decade due to a convergence of (1) an increased amount of data available for training large neural network models, (2) increased compute power [i.e., better/faster graphics processing units (GPUs), memory, etc.], and (3) key developments in the theory and design of neural network structures and training algorithms [e.g., the dropout regularization scheme,[2] use of rectified linear unit (ReLU) activations[3]]. Together, these three factors have led to steady improvement in the accuracy of neural network models for tasks such as object classification,[4] voice recognition,[5] machine translation,[6] and more. However, these improvements come at the cost of the models' size and complexity. This is shown in Fig. 1, which shows the size of different neural network models versus the accuracy that they achieve on the ImageNet object classification benchmark (data collected from Ref. 7). The blue, labeled datapoints represent the Pareto frontier and indicate that small gains in accuracy come at the cost of exponentially larger models. Moreover, implementing these state-of-the-art models on conventional CPU/GPU hardware incurs large overheads that are incompatible with application domains that have strict size, weight, and power constraints. These include, for example, AI on mobile devices, satellites, sensors, and, in general, AI-at-the-edge. Enabling AI in these domains is being pursued through

*Address all correspondence to Cory Merkel, cemeec@rit.edu

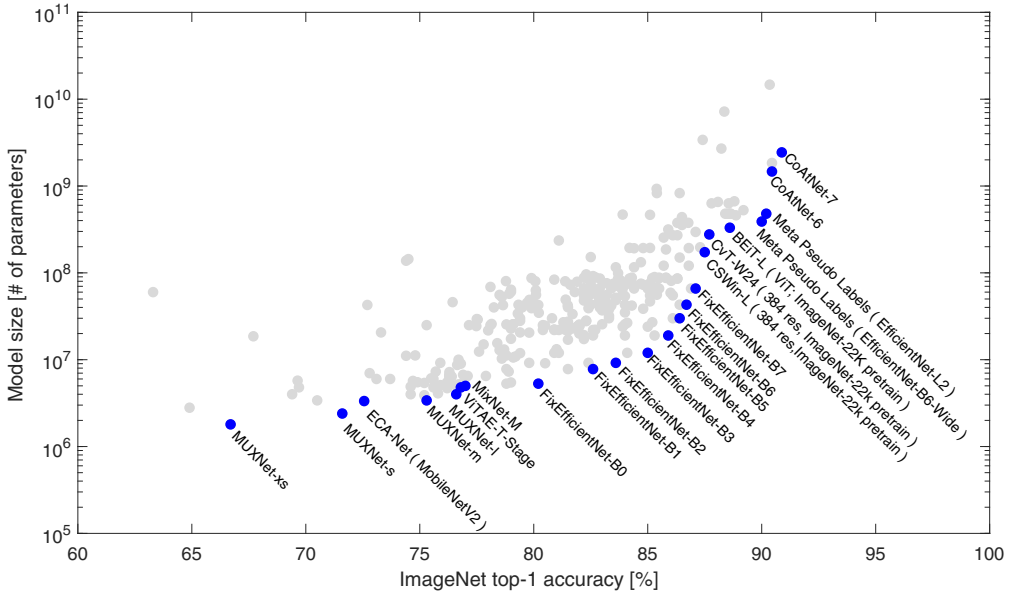


Fig. 1 Neural network model size versus accuracy on the ImageNet dataset. Each datapoint (gray and blue dots) represents a neural network model that has been published and tested on the ImageNet dataset, showing the number of model parameters and the Top-1 ImageNet accuracy that it achieved. Blue dots represent the Pareto frontier and indicate that linear gains in accuracy correspond to exponential growth in model size. Model names are indicated for points on the Pareto frontier. Data used to produce the plot are available in Ref. 7.

two fundamentally different approaches. In the first approach, modern deep neural networks (DNNs) are being compressed through techniques such as quantization/pruning and sharing techniques, low-rank factorization methods, transferred/compact weight methods, and knowledge distillation.[8] The second approach, which is the topic of this review, is to codesign hardware and algorithms that more closely emulate the anatomical and physiological characteristics that underlie the incredible efficiency of biological intelligence. This approach is called neuromorphic computing.

Pioneered in the late 1980s by Carver Mead, neuromorphic computing originated as an analog/mixed signal design paradigm whereby neurophysiological functions could be efficiently mimicked in integrated circuits (ICs) by leveraging the rich physics of transistors and other electronic devices. Today, the term neuromorphic computing is used more broadly to describe a set of brain-inspired hardware and algorithms for neural networks with varying degrees of biological plausibility. Dimensions of biological plausibility include characteristics such as information representation, digital versus analog hardware, stochasticity, coupling of processing, and memory. For example, today's von Neumann computers represent information in 32 or 64 bit binary format using digital, deterministic hardware, and have physically separated memory and processing units. In contrast, our brains represent information in the patterns of neuron action potentials, or "spikes," using noisy mixed signal hardware, and have closely coupled memory and processing units. The space of neuromorphic computing is expansive, and the term has been used to describe designs lying anywhere between these two extremes for applications spanning object/gesture recognition,[9] image reconstruction,[10] star tracking,[11] simultaneous localization and mapping,[12] surveillance, and monitoring.[13] In this paper, we review neuromorphic computing as it applies to brain-inspired vision. Specifically, we aim to inform readers about the computational primitives that are supported by neuromorphic hardware and how vision-related tasks such as classification can be implemented on the hardware using neuromorphic algorithms. It is impossible to exhaustively cover all of the excellent works on neuromorphic computing. Therefore, to keep the review concise, we focus on spike-based neuromorphic computing for vision applications. Other reviews on neuromorphic computing, such as Refs. 14 to 19, differ from ours in several ways, having either broader application focus, limited discussion on biological underpinnings of neuromorphic computing, or less attention to only spike-based designs.

We believe that this paper provides a good complement to the existing literature. Note also that we have opted to only give a limited historical perspective on neuromorphic computing, focusing instead on state-of-the-art designs. Interested readers may wish to refer to reviews such as Ref. 20 for a more detailed account of earlier work related to neuromorphic computing, especially in the early to mid twentieth century.

The rest of this review is organized as follows: Sec. 2 gives basic background on concepts from biological vision which are critical for understanding and designing neuromorphic vision systems. Section 3 discusses spiking neural networks (SNNs), which are the principal abstraction of the nervous system that neuromorphic computing systems employ to emulate brain function. In Sec. 4, we discuss some aspects of modeling the retina in neuromorphic hardware for efficient encoding of visual information. Then, Sec. 5 provides an overview of neuromorphic processors implemented in standard complementary metal oxide semiconductor (CMOS) technology. In Sec. 6, we discuss advanced memory technologies for scalable and efficient neuromorphic computing hardware. Section 7 concludes this review with an outlook and directions for future research.

2 Biological Basis for Neuromorphic Computing

Neuromorphic is a portmanteau combining neuro (relating to the nervous system) and morphic (having a specified form or shape). The goal of neuromorphic computing is to emulate the behavior of the biological nervous system, often focusing on the brain, through a combination of hardware and algorithm design. In the purest form of neuromorphic computing, the idea is that the hardware should completely define the algorithm, eliminating the need for software. However, hardware/software codesign generally adds more flexibility through different levels of design abstraction. In any case, a critical first step in designing a neuromorphic computing system is understanding the underlying neuroanatomical and neurophysiological aspects that are to be emulated. In this section, we provide a minimal overview to give context to the algorithms and hardware described in the rest of the review.

2.1 *Human Vision*

The evolution of biological vision is a long and fascinating journey from the development of light-calibrated circadian clocks and phototaxis in early bacteria, to the origin of the eye and advanced visual processing capabilities of humans and other animals.[21] Here, we will review some of the key steps in human vision as they relate neuromorphic computing. For more detailed discussions of human vision, refer to Refs. 22 and 23.

Figure 2 shows a high-level overview of some of the key steps that take place in human vision. Light enters the eyes, stimulating photoreceptors (rods and cones) on the retina (see

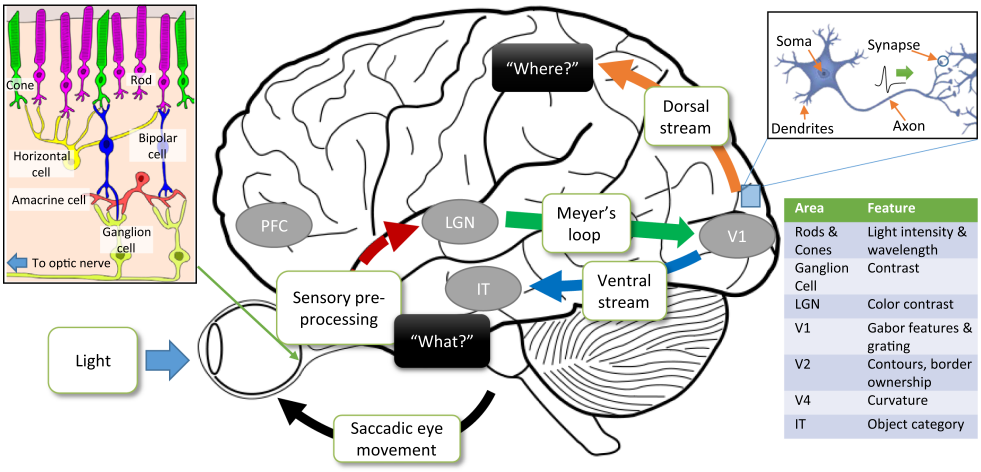


Fig. 2 High-level overview of key steps in human vision.

top-left inset of Fig. 2). The photoreceptors encode light intensity as graded (continuous) membrane polarization levels. Between rods and the different types of cones, our eyes can sense light in the range of ∼400 to 700 nm (the visual light spectrum). The response of the photoreceptors propagates through three additional types of retinal cells before reaching the retinal ganglion cells. Note that because of the small size of the retina (a fraction of a millimeter thick), most of the neurons communicate using graded potentials. The one exception is retinal ganglion cells, which produce action potentials or "spikes," [see Fig. 3(a)] which need to travel a long distance down the optic nerve to the thalamus. It is likely that spiking neurons evolved because encoding information as all-or-nothing spike patterns enables much better noise tolerance, especially when communicating over long distances. The output of retinal ganglion cells leaves the eye via the optic nerve. By this point, the retina has already performed a fair amount of preprocessing on the raw visual data, encoding it to capture information about spatial and temporal contrast, improve noise tolerance, and increasing invariance to overall light level. In essence, this is accomplished by encoding relative changes in light (e.g., spatial and temporal contrast) rather than only communicating information about absolute light intensities. In fact, if we were able to keep our eyes perfectly still, then a static scene would quickly fade from our perception. Luckily, constant eye movements called saccades and fixational eye movements maintain movement of a scene on our retinas.

After information leaves the eyes, it follows the optic nerves and eventually the optic tract to the lateral geniculate nucleus of the thalamus. From there, information radiates through several pathways in the temporal and parietal lobes collectively called Meyer's loop, eventually reaching the V1 and other visual cortices in the occipital lobe. It is here that several low-level features are extracted, such as edges and corners (see table in Fig. 2). The pathway from the retina to the V1 cortex is known as the retino-geniculo-striate pathway. From the occipital lobe, visual information takes two distinct pathways to the parietal (dorsal stream) and temporal (ventral stream) lobes. Processing in these locations allows us to locate objects in space/time and identify/classify objects. The features that are extracted along the ventral stream become increasingly complex, and neurons in the higher-level areas such as the inferotemporal (IT) cortex respond to specific classes of objects. It takes ∼100 ms for light to travel from our retinas to the IT cortex.[24] Other cortical areas such as the prefrontal cortex help us associate images with meaning.

2.2 *Spiking Neurons*

The generation of action potentials in neurons is a relatively complicated process, involving several steps. Here, we give a high level overview. As shown in the top-right inset of Fig. 2, a prototypical spiking neuron (Here, we will describe a typical spiking neuron. However, note that there are several tens to hundreds of types of neurons or even more, depending on the granularity of classification, each with different morphologies and behaviors.) such as a retinal ganglion cell or a cortical pyramidal cell has four main parts: dendrites, soma, axon, and synapses. Dendrites, together with the soma act as a leaky integrator of ionic currents that flow into or out of the neuron. These currents result from the opening and closing of channels in the cell membrane and through ion diffusion or active ion pumps. For example, when a presynaptic neuron creates an action potential, neurotransmitters are released and cause the opening of ligand-gated ion channels at the postsynaptic neuron's dendrites. This causes ions to flow resulting in an increase (in the case of an excitatory presynaptic neuron) or a decrease (in the case of an inhibitory presynaptic neuron) in the potential of the postsynaptic neuron's cell membrane. Interestingly, a number of complex computations such as boolean logic functions can take place in the dendritic arbor before information reaches the cell body.[25] If the membrane potential rises (also called depolarization) from its resting value of around −70 mV to a threshold value (typically around −55 mV), then an action potential about 1 ms wide and +40 mV at its peak [see Fig. 3(a)] is generated, which travels down the neuron's axon, ending at the synaptic terminals where neurotransmitter is released to the next set of neurons. After a neuron spikes, there is typically a refractory period of several milliseconds during which it cannot produce another spike. During this time, the neuron's membrane is hyperpolarized below its resting potential. The efficacy of neurotransmitters' modification of postsynaptic neurons' ion conductances is what is abstractly captured as a "weight" in neural networks. Much of our ability to learn has

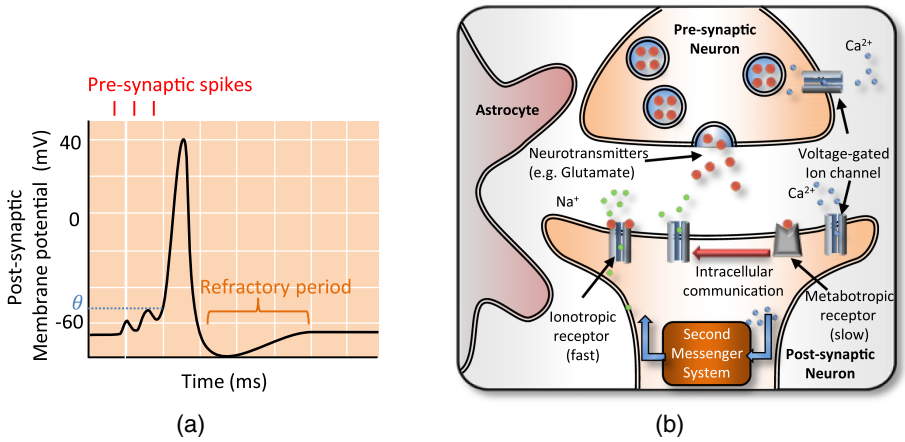


Fig. 3 (a) Transient characteristics of neuron action potential. (b) Overview of a chemical synapse.

been attributed to mechanisms that facilitate the strengthening and weakening of these efficacies, as well as the creation or removal of synaptic connections.

An ongoing debate and one of the biggest challenges in the area of neuromorphic computing is determining how much detail of the physiological processes needs to be modeled to faithfully capture the underlying computational principles. In the case of neurons, there is a wide range of models with different levels of complexity. On one end of the spectrum, multicompartment models such as the multicompartment Hodgkin–Huxley model,[26] FitzHugh–Nagumo model,[27,28] or Morris–Lecar model,[29] capture detailed behavior of ion conductances and can reproduce several complex behaviors of spiking neurons. However, these types of models are computationally intensive and usually reserved for research in neuroscience. On the other end of the spectrum, simple threshold models such as the McCulloch–Pitts model[30] capture only simple neuron behavior, such as whether the neuron is spiking above or below a particular rate. Other simple models such as sigmoid and ReLUs convey information about the spike rate of a neuron only and do not relay detailed temporal information. Due to their simplicity, they are very computationally efficient, and they are employed in most modern DNNs. Finally, a third set of spiking neuron models lies in the middle of the spectrum, with enough complexity to capture important temporal information of spike statistics, but enough abstraction to be computationally efficient. These models also lend themselves to relatively simple hardware implementations. More details on these models are given in Sec. 3.

2.3 *Synapses*

Communication between neurons takes place at synapses. There are two types of synapses in the nervous system: electrical and chemical synapses. Here, we focus on chemical synapses, which play a major role in both communication and learning in the central nervous system. Figure 3(b) shows some of the key components of a chemical synapse. When an action potential is generated in a presynaptic neuron, there is an influx of calcium ions that initiates a process in which synaptic vesicles, filled with neurotransmitters, dock at the cell membrane and undergo exocytosis: the release of the transmitter into the synaptic cleft. There are several types of neurotransmitters that have different effects on postsynpaptic neurons and can be found in different parts of the nervous system. Two particularly important neurotransmitters are glutamate and γ-aminobutyric acid (GABA), which have an excitatory and inhibitory effect on the postsynaptic membrane potential, respectively. After diffusing across the synaptic cleft to the postsynaptic neuron, neurotransmitters bind with two main types of receptors: ionotropic and metabotropic receptors, each of which allows the flow of ions such as sodium, potassium, or chloride across the cell membrane, changing its potential. Another class of cells called neuroglial cells also play important roles in a number of brain functions such as synaptic transmission. In particular, astrocytes help regulate the release and reuptake of neurotransmitters. Clearly, the efficacy with which a presynaptic action potential causes a change in the postsynaptic membrane potential depends on

several factors. Moreover, this efficacy is modified through numerous means based on factors such as the relative pre- and postsynaptic neuron activity, which can result in long-term strengthening (long-term potentiation or LTP) or weakening (long-term depression or LTD) of synaptic connections.

3 Neuromorphic Algorithms: Spiking Neural Networks

As we have seen in the last section, the computations underlying biological intelligence depend on networks of neurons communicating through spikes. State-of-the-art DNNs significantly abstract the behavior of biological neuronal networks, with several simplifications such as reducing a neuron's behavior to a simple spike rate. However, in spite of the recent rapid development of DNNs in the field of ML, specifically for computer vision tasks, their performance is not efficient enough compared with the biological brain in terms of power and speed.[18,31] While information in the biological brain is processed asynchronously and in parallel, information in DNNs is computed in a sequential form (i.e., computations at each layer of the network have to be completed on the input image before forwarding it to the following stage). Consequently, a significant delay appears in the network.[31] In 1996, Thorpe et al.[32] showed that the biological brain is able to recognize visual images with one spike propagating through all layers of the visual cortex, and Rolls and Tovee[33] measured the same visual processing speed in the macaque monkey. Theses works highlight the incredible efficiency of the way our brains encode information in spikes.[31]

The efficiency of spike-based computation motivated ML and neuroscience researchers to begin exploring SNNs, the third generation of neural networks. Computation in SNNs is event-driven as in the biological brain, so each neuron in the network generates its outputs only when enough spikes indicating the existence of a specific feature or pattern have been detected.[31] This feature gives SNNs the capability to solve complex spatiotemporal tasks and to make use of efficient event-driven sensors, such as event-based cameras.

Table 1 shows a comparison of the main aspects between some of the key properties of DNNs and SNNs for vision. As stated in the previous section, synchronous computation in each layer of DNNs can be time consuming. On the other hand, in SNNs, the computation is processed asynchronously in spike form, allowing information to propagate to the next layer before all computation in the current layer is complete. However, this asynchrony combined with the nondifferentiable nature of spikes complicates the credit assignment problem and limits the use of many popular training algorithms employed in DNNs. On the upside, though, the inherent temporal dynamics of SNNs allows them to perform more complex tasks than DNNs.[34] For example, SNNs have neuron-level temporal memory, enabled by the leaky integration of information at the neuron's input. This means that even purely feedforward SNNs have an inherent short-term memory. Contrast this with DNNs, which can have short-term memory enabled by their network topology (e.g., with recurrent connections), but there is usually no built-in temporal memory at the level of individual neurons. This extralayer of short-term memory in SNNs makes them a good fit for temporal processing of data such as audio and video.

Table 1 Comparison between DNNs and SNNs for vision.

Aspects	DNNs	SNNs
Data processing	Frame-based	Event-based
Latency	High	Low
Differentiable	Yes	No
Activation	Sigmoid, ReLU, etc.	Spike
Model complexity	Low	High
Short-term memory	Network level	Neuron and network levels

Below, we discuss three critical design aspects of SNNs: spiking neuron models, spike encoding, and learning in SNNs.

3.1 *Spiking Neuron Models*

The spiking neuron models used in neuromorphic computing need to have enough complexity to capture key dynamic properties of biological neurons without having significant computational or hardware overhead. Figure 4 shows a comparison of different models presented in the literature versus their computational cost.[35] The vertical axis shows the number of biological features that the model exhibits, with positive error bars indicating the maximum number of features that a model can exhibit if its parameters are carefully tuned. The horizontal axis indicates the computational complexity of the model. In general, we see that adding more biological features leads to exponential growth in the computational cost. Also, note that the simple rate-based neurons (e.g., ReLU, sigmoid, etc.) that are used in the DNNs of Fig. 1 will have computational cost and biological realism that is on the order of, or less than, the integrate-and-fire (IF) neuron model. However, even with a reduced computational cost, measured in floating point operations per second, the hardware overhead of implementing rate-based neurons (number of transistors, power consumption, etc.) can be much larger than that of spiking neurons.

IF or leaky IF (LIF) is one of the most popular spiking neuron models used in neuromorphic computing:

$$\tau_m \frac{\mathrm{d}s}{\mathrm{d}t} = s_{\text{rest}} - s(t) + \sum_j w_j \sum_k \alpha(t - t_j) + R_m I^{\text{ext}}(t), \tag{1}$$

where s is the membrane potential, s_{rest} is the resting membrane potential, τ_m is the membrane time constant, R_m is the membrane resistance, w_j is the synaptic weight connecting the neuron to the j'th presynaptic neuron, t_j is the last spike time of pre-synaptic neuron j, I^{ext} is an external current that is driving the neuron, and α describes the transient response of postsynaptic membrane current as a result of a presynaptic spike. While this equation describes the membrane dynamics, it also needs to be paired with a rule that says when the membrane voltage reaches a threshold, the voltage will be reset, and the neuron will produce a spike. This model has low computational cost and can easily be created in hardware with a relatively compact circuit. However, it is not able to capture some of the more complex biological features of real spiking neurons such as phasic spiking, tonic and phasic bursting, spike frequency adaptation, and accommodation, to name a few. An excellent review of these features and the models shown in Fig. 4 can be found in Ref. 35. Although, as noted in Sec. 2.2, and indicated in Fig. 4, more detailed neuron models such as the Hodgkin–Huxley model,[26] which requires numerical solution of four nonlinear differential equations, have significant computational overhead, making it

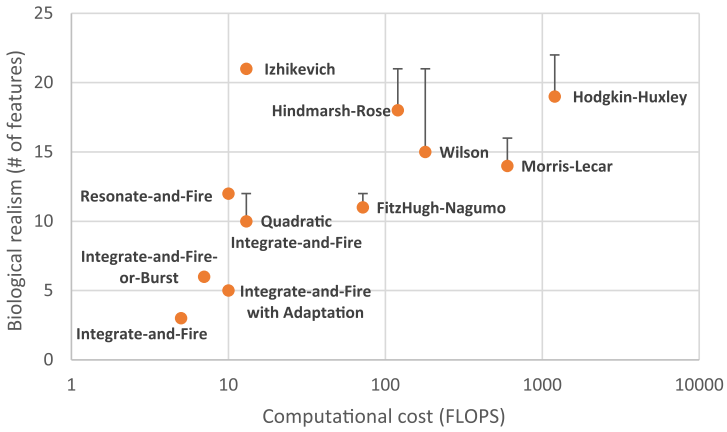


Fig. 4 Comparison of different spiking neuron models, indicating the number of biological features they are able to reproduce versus their computational cost.[35]

difficult to employ them in low-power neuromorphic systems. Moreover, it is still unclear which biological features are necessary for designing neuromorphic systems with a particular set of desired behaviors. A number of other models have been proposed in the literature that have medium complexity and are a good tradeoff between feature richness and computational cost. These include models such as the adaptive IF model,[36] the spike response model,[37] and the Izhikivich model,[38] which are able to capture more complex dynamics than the LIF model such as bursting and chattering. Still, other models capture additional behaviors that are important for neural computation, such as stochasticity of spiking and synaptic transmission[39] or energy-dependence of neural activity.[40,41]

3.2 *Spike Encoding*

The dynamics and learning performance for SNNs depend on the type of spike encoding scheme employed. There is evidence of multiple encoding schemes at play in the brain: rate coding, temporal coding, and population coding.[42] Each of these encoding schemes has pros and cons. Rate codes represent information by the average number of spikes that a neuron produces in a given time window. Rate encoding is robust to noise that may be superimposed with the input spikes. However, achieving high precision with rate encoding requires counting spikes over long time windows, which leads to high power consumption and latency when implementing this scheme in hardware. In contrast, temporal encoding schemes use the exact timing of spikes to represent information. This is usually more efficient in terms of energy and latency because it relies on fewer spikes. For example, interspike interval codes represent information using the time elapsed between a neuron's spikes, and spike latency codes encode information as the latency between a neuron's spike and the onset of a stimulus.[43] These codes are less noise-tolerant because of their dependence on exact timing. Another encoding technique is to use groups of neurons, rather than a single neuron, to represent information. This strategy, called population coding, is an equitable approach compared with the other coding schemes, but more neurons are required to represent a feature of the data.

3.3 *Learning in SNNs*

Our brains are remarkably adept at taking in new information, combining it with stored knowledge, generalizing it to create new knowledge, and applying it to solve both simple and complex problems. Although we do not fully understand all of the details of how this learning process works, it is generally accepted that modification of synaptic efficacy plays a major role.[42] However, we also note that other forms of plasticity are critical in learning such as the formation of new cells and new connections in the brain (e.g., neurogenesis and synaptogenesis).[44] Here, we focus primarily on modeling learning mechanisms that occur through modification of synaptic efficacy. This type of learning has led to several powerful (nonspiking) DNN models that are trained by incrementally modifying the network weights according to a loss function such as classification accuracy. While it is tempting to apply the learning algorithms employed in DNNs to SNNs, there are some challenges related to SNNs' differentiability and temporal credit assignment. Below, we review some of the popular methods for training SNNs. These can broadly be categorized as supervised (where labeled targets are provided), unsupervised learning (where no targets are provided), and reinforcement learning (where learning is based on rewards). Here, we focus on supervised and unsupervised approaches.

3.3.1 *Supervised learning*

State-of-the-art supervised learning in SNNs can be categorized as either (1) DNN-to-SNN conversion, (2) spike timing-dependent backpropagation (STDB), or (3) combined training. Here, we discuss some of the main approaches within each category, but we point the interested reader to Ref. 45 for a dedicated review on supervised training of SNNs.

Conversion from DNN to SNN. A straightforward way to train an SNN is to first train a DNN with standard backpropagation training and then convert it to an SNN by either modifying

the network topology or approximating the SNN's discontinuous behavior using common DNN functions. Among the first to explore this idea were Pérez-Carrasco et al.[46] In that work, the authors mapped continuous neural networks to SNNs to enable compatibility with event-based image sensors, which are discussed in Sec. 4. Other works, such as Ref. 47, model the spike rate of SNN neurons using ReLU functions and show good performance on simple datasets but have limited scalability to large networks. In Ref. 48, a weight normalization-based method is used to regulate firing rates of SNN neurons to more closely match the behavior of DNNs with ReLU activations. Other techniques such as Refs. 4 and 49 aim to tackle the conversion of DNN to SNN by reducing how much information is lost during the membrane potential reset after a spike occurs. In Ref. 50, a threshold balancing technique is proposed to directly convert DNNs to SNNs without modifying their weights. The choice of the SNN neurons' firing threshold allows one to adjust between accuracy and latency for rate coded designs. In Ref. 51, the authors proposed a conversion technique based on time to first spike. This method succeeded to reduce power at the expense of the accuracy. Although conversion from DNN to SNN has some limitations in implementing operations, such as average pooling, bias, and batch operations, other works have shown that some of these limitations can be overcome.[4]

Spike timing-dependent backpropagation. The discrete and nondifferentiable nature of SNNs is inconsistent with the gradient descent-based algorithms that are commonly used for DNNs, such as backpropagation. Consequently, several papers presented solutions to overcome this challenge. One of the first proposed algorithms for applying gradient descent training to SNNs is SpikeProp,[52] which models each synapse as a set of parallel connections between a single pre- and postsynaptic neuron pair, each with its own weight and delay value. Then, a backpropagation-like algorithm is derived to minimize the error between the spike times of a target output spike train and the actual SNN output spike times. One of the key ideas employed is to approximate a linear relationship between the change in a neuron's membrane potential and its spike time. The authors show that their approach can be used to solve the classic XOR problem, but a limitation of their algorithm is that each neuron is only allowed to spike once per input. Since then, several additional algorithms have been proposed, such as, Refs. 53 to 56. The main idea of the proposed solution methods is to approximate the functionality of the spiking neurons as a continuous differentiable function or a surrogate gradient (SG) estimate of the SNN's real gradients. Although, a drawback of these methods is that they often require a large number of time steps per input when rate coding is used. This is because the precision of the rate code increases as a function of the time window over which it is calculated. Another interesting technique, proposed in Ref. 57 is to use a random network of weights to propagate errors backwards from the network's output. In the regular implementation of backpropagation, errors need to be propagated through the same weights in the backwards pass as in the forward pass. However, by breaking the symmetry and using random weights in the backward pass, the authors are able to achieve good performance with the potential to reduce the complexity of the algorithm's hardware implementation. Recently, various works presented the demonstration of STDB learning rules for hardware implementations of SNNs. In Ref. 58, a simplified backpropagation learning rule for SNNs is proposed that is mapped to a memristor-based neuromorphic processor. The learning rule requires only two additional bits per neuron to store the neuron's activity (spike or no spike) in the previous time step, as well as the derivative of its activation value. In Refs. 59 and 60, the authors used a supervised learning algorithm based on temporal coding as an approach for an energy efficient neuromorphic processor design. As discussed in Sec. 3.2, temporal codes are generally more efficient than rate encoding. Furthermore, rate codes are quantized based on the size of the window used to calculate them, making it more difficult to perform gradient descent on them (since quantized functions are discontinuous). On the other hand, temporal codes such as spike latency encoding have continuous values, enabling easier implementation of algorithms such as backpropagation.

Combined learning. Another training approach, such as in Ref. 61 combines SNN-DNN conversion followed by gradient descent training. By combining these two techniques, fewer number of epochs and fewer time steps are required. A similar method was used in Ref. 56 where the authors proposed a quantization-aware spike timing-dependent backpropagation

(Q-STDB) algorithm that maps DNNs to SNNs with quantized weights using the two-step approach. Other hybrid methods such as Ref. 62 combine STDB learning rules with evolutionary or other optimization algorithms.

3.3.2 *Unsupervised learning*

Unsupervised learning in SNNs is largely based on the postulate put forward by Donald Hebb, which is often paraphrased as: "neurons that fire together wire together."[63] For rate-based neurons, this learning rule can be expressed as a weight update that is proportional to the product of the presynaptic and postsynaptic firing rates. In this case, it can be shown that the vector of weights impinging on a postsynaptic neuron will eventually align with the dominant eigenvector of the covariance matrix of inputs[64] or the principal component of the presynaptic neurons. A number of variations of Hebb's rule have been proposed over the years. For example, Oja's rule[65] modifies the basic Hebbian learning rule with a weight decay term that ensures weight vectors will not grow without bound. Other modifications, such as the Bienenstock, Cooper, and Munro rule,[66] force competition between synaptic weights, which can also help with normalization. Hebbian-like learning in SNNs is often realized through spike timing-dependent plasticity (STDP).[67] STDP strengthens synapses between neurons where one causes the firing of another and weakens those where there is no evidence of the causation. The causal relationship or lack thereof is inferred from the relative pre- and postsynaptic firing times. If the presyaptic neuron fires shortly before the postsynaptic neuron, then the synapse is strengthened (LTP). If the presynaptic neuron fires shortly after the postsynaptic neuron, then the synapse is weakened (LTD). By reinforcing the causal relationships, the network learns correlations between different features of the inputs. Combining STDP learning with other constructs such as lateral inhibition (firing of one neuron causes reduced activity of its neighbors) can enable unsupervised extraction of discriminative input features. However, as demonstrated in works such as Ref. 68, more research needs to be done to get the performance of STDP feature learning to the level of more conventional methods such as autoencoders. Recently, various works presented the demonstration of SNN hardware focused on the STDP learning rule.[69–72] In addition, approaches that combine unsupervised learning with supervised or reinforcement learning have also been explored for SNNs.[73]

4 Neuromorphic Vision Sensors

Efficient implementation of the SNN models discussed in the last section for vision tasks starts with a good strategy for encoding raw visual data. The human eye has the capability to perceive a dynamic range of 160 dB overall,[74] distinguishing details both in intense sunlight and dim starlight. It has been reported that the intrascene dynamic range of the human retina exceeds 120 dB even in relatively dim light environments. At the same time, the human eye can detect flashes as short as a few milliseconds.[74] Designing a camera that meets these specifications is a great challenge.[74] Broadly speaking, there are two types of cameras: conventional frame-based cameras and event-based cameras. Frame-based cameras capture complete light intensity information blindly at a fixed rate,[75] even if the scene is static. The aim of event-based cameras is to emulate the behavior of the biological eye by primarily capturing dynamic information such as movement over time or spatial contrast.[75] This significantly helps to alleviate downstream processing latency, blurring during fast motion, data redundancy, and issues related to difficult lighting conditions.[76,77] The outputs of the event-based cameras are characterized by an address-event representation with four different components: (x, y) interpreted as the two-dimensional coordinates of the event, a timestamp (t), and a polarity (p) for the change.[75] Because of the event-driven nature of the camera's output, it easily forms a front end for downstream processing by SNNs. The difference between the two camera types is shown in Fig. 5. On the left, a frame-based camera captures a poster of rocks. Even if both the scene and the camera are static, the frame-based camera will continue to capture all of the pixel intensity information. In the center, an event-based camera reports areas of high temporal contrast (red and blue) while ignoring areas that have small changes in pixel intensities over time (gray). Here, the event camera is moving

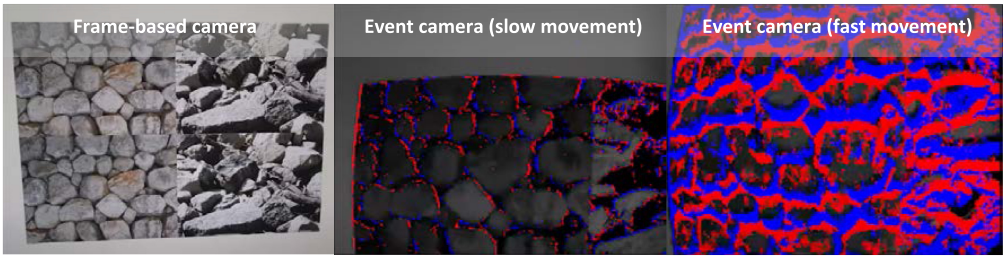


Fig. 5 Differences between standard (frame-based) and event-based cameras. Event cameras only capture dynamic information that comes from spatial and temporal changes in pixel intensities.[78]

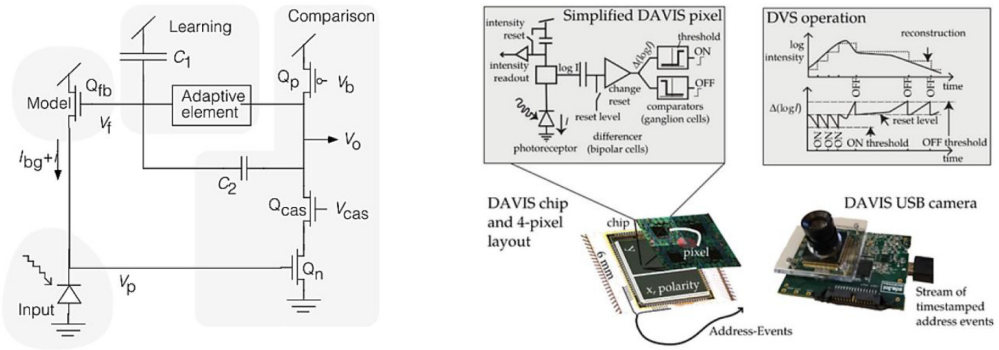
slowly across the poster. On the right, the event camera moves more quickly, which is reflected by a higher level of activity in the camera's output.

The first practical event-based sensor, the 64×64 pixel dynamic vision sensor (DVS), was developed by iniVation with the collaboration of the Institute for Neuro-Informatics (INI) at the Swiss Federal Institute of Technology in Zurich (ETH-Zurich).[79] In 2006, they developed a 128×128 pixel array.[80] Soon after, the performance metrics, such as minimum attainable event threshold, threshold mismatch, pixel bandwidth, dynamic range, and latency, were proposed as critical metrics for characterizing the performance of event-based cameras.[81] Today, a number of event-based cameras are commercially available,[82] and below we review a few of the key processing steps in their operation.

4.1 *Principle of Operation*

The history of neuromorphic event-based cameras dates back to the work of Mead's logarithmic pixel sensor[83,84] used in the early Mahowald and Mead silicon retinas photoreceptor[85] and in their See Hear chip.[86] Figure 6(a) shows a transistor-level design of a photoreceptor circuit.[87] The principle idea is that a model of the background illumination is learned over time and compared with the light intensity given as the output of a photodiode. The output of the circuit conveys information about relative changes between the model and the intensity indicated by the photodiode, allowing only temporal changes in illumination to be sent for downstream processing.

The photodiode's current is sourced from an NMOS transistor Q_{fb}. The NMOS transistor's gate is connected to the output of an inverting amplifier (Q_p, Q_{cas}, Q_n) whose input is connected to the photodiode. This structure, known as transimpedance configuration, converts the logarithmic photocurrent to a voltage, which also hold the photodiode clamped at a virtual ground.[89] The model of intensity over large time scales is stored as a charge on capacitor C_1. The transistor Q_{fb} works in subthreshold mode and its feedback supplies the model photocurrent I_{bg}.[87] The gate



(a) (b)

Fig. 6 (a) A photoreceptor in transistor form labeled with the elements of the conceptual model.[87] (b) The basic structure of simple DVS pixel.[88]

voltage V_f and the photocurrent $I_{bg} + i$ control the value of voltage V_p. As the transistor Q_{fb} works in subthreshold mode, the current is in exponential form, and the receptor (i.e., photodiode) becomes a logarithmic detector. The output voltage V_p is very small, thus an inverting amplifier with high gain is needed. The memory behavior is achieved by feeding the output V_o to V_f through an adaptive element that could be represented by a resistor and two capacitors C_1 and C_2. For more details see Refs. 87 and 89.

In Fig. 6(b), the logarithmic voltage V_o is converted from analog to digital by a differencing amplifier that amplifies the change in log intensity from the memorized value that was stored on the capacitor.[75] This voltage V_d is compared with the reference signal through two comparators to detect the increasing and decreasing of the brightness. Consequently, an ON or OFF event is generated if the input exceeds a specific threshold. The location, time, and polarity are communicated at the camera's output with each generated event. In addition, whenever the event or the spike is triggered, the reset switch drains the capacitor to reset the change level to zero.[88]

5 Neuromorphic Processors

Neuromorphic vision sensors such as the ones described in Sec. 4 perform encoding and early processing of visual data, similar to the function of the retina and early stages of vision in the brain. Efficiently emulating further processing stages in the brain, such as those along the ventral stream, requires additional hardware for SNNs that can learn visual tasks such as object recognition. While conventional computer hardware such as CPUs and GPUs offer flexibility for a number of tasks, their architectures are not well-suited for efficient implementations of SNNs. In particular, SNNs often run asynchronously, have massive parallelism, sparse communication, and computation in memory, which stands in stark contrast to the sequential operations and von Neumann separation of memory from computation on CPUs and GPUs.[31] Consequently, a large body of research has been directed toward filling this gap, developing neuromorphic processors which are specifically customized for efficient SNN acceleration, moving from von Neumann architectures to distributed and compute-in-memory designs.[18]

Neuromorphic processors can have digital, analog, or mixed signal implementations. The analog and mixed signal implementations replicate the biological brain better than digital designs in terms of power and speed.[90] In fact, the concept of neuromorphic computing was originally proposed as a paradigm in which analog ICs mimicked the behavior of neural tissue by leveraging the inherent properties of devices such as MOSFETs, resistors, capacitors, etc. For example, Mead's axon-hillock spiking neuron uses an RC circuit to emulate the behavior of a leaky cell membrane and an amplifier with positive feedback to produce spikes once a voltage threshold is reached.[91] Since then, several analog designs with varying levels of complexity have been proposed to efficiently mimick neuronal and synaptic behavior.[92] However, scaling up analog designs presents a significant challenge as they suffer from effects of noise and process, voltage, and temperature variations, which can limit the precision/reliability of SNN computations compared to digital designs. In contrast, digital designs offer better flexibility, which helps designers to quickly prototype different types of algorithms with low power consumption compared to GPUs. Mixed signal chips combine the advantages of the analog chips (e.g., low power consumption) and the digital ones (e.g., high precision). Below we highlight some of the key design features and application spaces of state-of-the-art neuromorphic processors, with a summary comparison of large-scale designs given in Table 2.

5.1 *Large Scale Analog/Mixed-Signal Neuromorphic Processors*

Neurogrid[93] and BrainScaleS[94] are two of the most well-known neuromorphic processors that are based on analog/mixed-signal designs. Neurogrid is a subthreshold mixed analog–digital system designed by the Brains in Silicon group at Stanford university and fabricated in 180-nm CMOS technology. This chip is designed to emulate biophysics of cortical models[93] with million of neurons and billion of synapses. The chip does not support on-chip learning. Recently, their group implemented a Braindrop prototype chip that is fabricated on 28-nm FDSOI technology

Table 2 Comparison of large-scale neuromorphic processors.

Platform	Brain	SpiNNaker	Loihi	TrueNorth	Neurogrid	BrainScaleS
Technology	Biology	Digital	Digital	Digital	Analog	Analog
Number of transistors	—	100 M	2.07 B	5.4 B	23 M	15 M
Chip size	2 L	1 cm^2	60 mm^2	4.3 cm^2	1.7 cm^2	0.5 cm^2
Neurons/chip	10^{11}	16 k	131 k	1 M	65 k	512
Synapses/chip	10^{15}	16 M	126 M	256 M	100 M	100 k
Chips/board	—	48	—	16	16	352
Neurons/board	—	768 k	—	16 M	1 M	200 k
Synapses/board	—	768 M	—	4 B	4 B	40 M
Power/efficiency metrics	20 W	—	23.6 pJ/ synaptic Op.	26 pJ/ synaptic Op.	—	—
On chip learning	Yes	Yes	Yes	No	No	Yes

process and has 4096 neurons integrated into a single core. Their goal is to build multiple cores of Braindrop to construct a large system called Brainstorm.

BrainScaleS is an above threshold mixed analog/digital system at wafer scale.[95] This work is based on the fast analog computing with emergent transient states project and is a collaborative effort between the University of Heidelberg and Dresden University of Technology. Fabricated at the 180-nm node, BrainScaleS contains around 180 k neurons and 40 M synapses. This chip runs faster than the biological brain with acceleration factor ranging from 10^3 to 10^5 to understand the biological functions of the brain with emphasis on long term learning. At the 2018 Neuro-Inspired Computational Elements (NICE) Workshop, the second generation of BrainScaleS processor was released with 64 neurons and 2 k synapses. It includes a programmable plasticity processor as well as a multicompartment neuron model that enables a range of behaviors such as dendritic computations and structural plasticity. For example, in Ref. 96, it is demonstrated that a combination of structural plasticity, based on synaptic pruning and reassignment, and Hebbian learning leads to high classification accuracy and optimized resource utilization on BrainScales-2.

5.2 *Large-Scale Digital Neuromorphic Processors*

Three of the most prominent large-scale digital neuromorphic processors are SpiNNaker from the University of Manchester, TrueNorth from IBM, and Loihi from Intel. SpiNNaker is a 130-nm CMOS design with a distributed von Neumann architecture,[97] employing 18 ARM processor cores per chip. As part of the European Human Brain Project, it is used in studies directed at understanding detailed brain behavior.[15] SpiNNaker has been paired with event camera applications, such as stereo depth estimation,[98] optic flow computation,[99] object tracking,[100] and object recognition.[101] A second generation of SpiNNaker is in progress. Their aim is to simulate a large number of neurons per chip compared with the first generation and their prototype chip is implemented in 22-nm FDSOI technology.

TrueNorth is a digital asynchronous chip with fixed hardware built by IBM under the DARPA SYNAPSE program. This chip follows the full custom digital hardware implementation, which is characterized by high density. Each chip contains around 5.4 million transistors fabricated in 28-nm CMOS technology. It achieves a very low average power consumption of around 70 mW. This energy efficiency partially stems from close coupling of memory used to store synaptic weights with processing logic for computing neuron activations. It also uses very low precision computation (e.g., ternary synaptic weights), which saves power and area.

TrueNorth has been used in a number of vision applications, such as gesture recognition,[9] stereo reconstruction,[102] and optical flow estimation.[99] Its energy advantage has also been demonstrated in works such as Ref. 103, where object detection using the you only look once algorithm was shown to be 280× more energy efficient on TrueNorth versus a GPU. One big drawback of TrueNorth is that it does not perform on-chip learning, which limits its use in applications where real-time adaptation is needed.

In 2018, Intel released its Loihi SNN chip with a pure digital asynchronous design, fabricated on 14-nm FinFET process technology.[104] Loihi offers flexibility for tuning neuron and synaptic dynamics, allowing it to achieve a larger range of biologically plausible behaviors and learning dynamics compared to TrueNorth. A number of vision applications have been explored using the Loihi chip, such as image retrieval and segmentation, object classification, and gesture recognition.[105] Intel's Neuromorphic Research Community (INRC) encourages researchers to use Loihi to find creative solutions for the current challenges that face neuromrophic computing architectures for real-time applications.

5.3 *Other Low Power SNN Accelerators*

In addition to the large-scale design presented above, a number of other chips have been proposed for low power acceleration of SNN algorithms. Two chips, ODIM and MORPHIC, were released by Frenkel et al. in 2018[106] and 2019,[107] respectively. They have a digital full custom design and have been used for vision applications such as image classification. ODIN was fabricated in 28-nm FDSOI CMOS technology and supports online learning through spike-driven synaptic plasticity. It also supports both LIF and Izhikevich models for flexible neuron dynamics. MORPHIC is a quad core digital processor, fabricated with 65 nm technology. It uses binary weights as an approach to reduce power and increase efficiency of the system along with a stochastic spike-driven synaptic plasticity learning rule. Another chip, the reconfigurable online learning spiking neuromorphic processor (ROLLS),[108] employs a subthreshold analog mixed signal design. ROLLS has 256 neurons, 128 k synapses, and support for online learning. Recently, it has been scaled up to the dynamic neuromorphic asynchronous processor (DYNAPs) with 1 k neurons and 64 k synapses.[109]

Often, it is critical to codesign both the hardware and software for neuromorphic processors. For example, Ref. 110 uses hardware–software co-design to leverage the sparsity of spike trains for a low-power event-triggered neuromorphic chip. It can be trained online with weight-dependent STDP algorithms and efficiently decreases hardware complexity through an innovative modification of the STDP learning rule. We believe that close coupling of hardware/algorithm/software design is key for next-generation neuromorphic systems.

6 Memory Technologies for Spike-Based Neuromorphic Computing

Our brains contain on the order of 100 trillion synaptic connections which are responsible for learning and maintaining information that is related to our entire lifetime of experiences in a 20-W power budget. To emulate the brain's massive storage and learning capabilities, it is critical for neuromorphic computing hardware to incorporate efficient memory technologies. For energy efficiency, it is desirable to have synaptic weights and other parameters stored in nonvolatile memory (NVM) devices, which do not require frequent refresh operations to maintain their state. This is especially true if the SNNs implemented on the neuromorphic hardware have very sparse synaptic operations. There are a number of emerging NVM technologies that are being explored for neuromorphic computing applications, including two-terminal devices, which can generally be classified as "memristors." Memristors exhibit behavioral similarity to chemical synapses, combining storage, adaptation, and physical connectivity in one device.[18] Table 3 shows a comparison of different two-terminal NVM technologies across several key metrics. Each device exploits different physics to achieve nonvolatile storage of information as a resistance state. Then, Ohm's law can be exploited to read the stored information by applying a voltage and reading the resulting current. In phase change memory (PCRAM), a phase change material is transformed between amorphous and crystalline phases to increase or decrease its resistance

Table 3 Comparison of NVM technologies for neuromorphic computing.[111,112]

	PCRAM	ReRAM	STTRAM
Cell size	4 to $20F^2$	4 to $6F^2$	$12F^2$
Write energy	6 nJ	2 nJ	<1 nJ
Write latency	100 ns	10 ns	2 to 25 ns
Read latency	10 ns	1 to 10 ns	2 to 25 ns
MLC	4 bits	4 to 6 bits	2 bits
Endurance	10^6 to 10^9	10^6 to 10^9	10^{14}
Retention	10^4 s	10^4 s	—
ON/OFF ratio	10^3	10^3	10

value, respectively. One challenge associated with PCRAM is the high current that is required to melt the phase change material, leading to large write energies. Another drawback is the relatively slow write latency which stems from the time required for the crystallization process.[113] On the other hand, resistive RAM (ReRAM) and spin transfer torque RAM (STTRAM) both have low write energies and latencies, making them better suited for neuromorphic accelerators with on-chip learning. STTRAM is composed of a magnetic tunneling junction, where electron tunneling probability is modulated by modifying the relative magnetic orientation of two ferromagnetic films. Two drawbacks of STTRAM are its small number of achievable resistance states (2 bits) and low ON/OFF conductance ratio. In contrast, ReRAM, which operates by modulating the distribution of defects such as oxygen vacancies in an insulating transition metal oxide film, offers a high ON/OFF ratio and offers 4 to 6 bits of memory, which enables quantized neural network inference with limited accuracy degradation.[114]

Three-terminal NVMs have also been explored for implementing synaptic functionality in neuromorphic computing. For example, neuromorphic architectures based on flash memory have been studied for the past several years.[115] In Ref. 116, a three-layer neural network with 784 inputs, fabricated in 180 nm technology, uses flash memory cells to store synaptic weights. The chip was programmed using a transfer learning approach with 6-bit analog precision and the experimental results shows an accuracy of 94.7% for classification of MNIST handwritten digits. Other three-terminal devices such as ferroelectric field effect transistors (FeFET) have also been explored. In Ref. 117, the authors present a recent experiment for a three-layer SNN with 784 inputs to classify handwritten digits. This experiment was demonstrated on 28-nm high-K metal gate FeFET technology. They use an SG learning algorithm to perform supervised learning on the MNIST dataset. The main advantage of using FeFETs for implementing electronic synapses is the reduction of the variability compared with the other emerging technologies such as two-terminal memristor devices.

Both two-terminal and three-terminal NVMs can be integrated into high-density crossbar circuits,[118–122] as shown in Fig. 7, which offers a $4F^2$ cell size, where F is the wire half-pitch. The NVM conductances implement the weight matrix between two densely connected layers of neurons. Each spike produced in a presynaptic layer of neurons produces a current that flows through each NVM in the corresponding row that is proportional to its stored conductance. The currents from each row get summed on the columns and are the inputs to the postsynaptic neurons. Therefore, loosely speaking, each row can be though of as a presyanaptic neuron's axon, while each column is the postsynaptic neuron's set of dendrites. Crossbar sizes up to 128×128 have been demonstrated,[123] and multiple crossbars can be combined to implement larger layers. However, a key challenge, especially in large crossbars, is sneak path current, which is current that flows in one of several unintended parallel current pathways, and is affected by several factors, such as the NVM OFF conductance, the potential at the inputs of postsynaptic neurons,

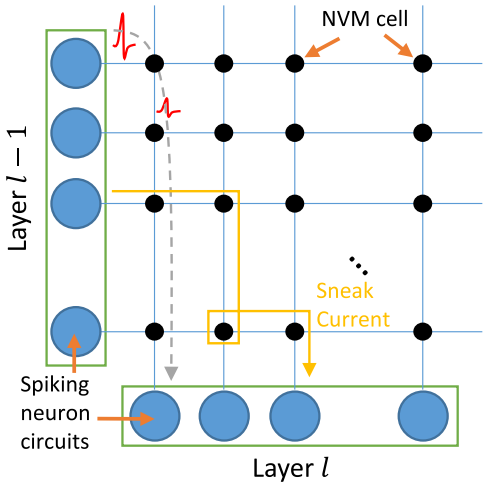


Fig. 7 NVM crossbar circuit for implementing weight matrix between two fully connected neuron layers in a neuromorphic DNN accelerator.

parasitic wire resistance, and more.[124] Sneak currents not only cause memory read and write errors but also increase the crossbar power consumption. Several mitigation strategies exist, such as row/column grounding,[125] algorithmic approaches (e.g., Ref. 126), and use of selector devices.[127] The use of MOSFET selectors is the most common approach to mitigating sneak paths and allows for easy and controllable isolation of NVM cells to avoid interference between cells on adjacent bit lines. Two-terminal NVMs such as ReRAM integrated with MOSFET selectors are called 1T1R cells. Other NVMs, such as FeFETs[128] and other gated synaptic devices,[12] employ a three-terminal structure, so their selector is "built-in."

Neuromorphic chips based on NVM crossbars have been demonstrated for vision tasks, such as classification[129–131] and several others.[132] Training these chips can be performed on or off-chip. For off-chip training, synaptic weights are found using software such as Tensorflow and then transferred onto the chip by programming the NVM conductances.[133–136] A number of works have also proposed circuits for both supervised[137–140] and unsupervised[141,142] on-chip training. NVM crossbar-based implementations often come with huge energy advantages over conventional hardware. For example, in Ref. 143, the authors show that an NVM crossbar-based neuromorphic chip can achieve over 100,000× improvement in energy efficiency for classification tasks compared with a reduced instruction set computer processor. However, despite these successes, there are some key remaining challenges in mapping neural networks to NVM crossbar architectures that need to be addressed. One challenge is related to the variability of NVM devices, which can cause device-to-device differences in behavior (e.g., change in resistance with a given write voltage) or differences in a single device each time it is read/written. See, e.g., Refs. 144 and 145, for a review on NVM devices for neuromorphic computing. These discrepancies between the expected and actual fabricated NVM device behaviors can lead to significant degradation of a neural network's performance when it is mapped to the hardware.[146–148] To mitigate the effect of device variations, it is important to either map the location of faulty devices before or during the training process[149,150] or perform training *in-situ* (on-chip).[146–148] On-chip training is especially attractive because it allows the device variations/faults to be incorporated into the training process without the need to explicitly map them. However, support for complete on-chip training requires additional circuitry to support the state-of-the-art learning methods, which may have a large overhead. Hybrid training represents a good compromise, where a portion of the training is performed off-chip and some additional on-chip training, with reduced overhead, is used to mitigate the effects of device variations.[147,148] But, even in the ideal case of variation-free circuits, a key challenge remains in mapping neural networks to NVM crossbar architectures. The difficulty stems from the fact that the number of weights between two layers of neurons may be larger or smaller than the size of the crossbars available on the hardware. Moreover, memristor crossbars can be utilized most efficiently for strictly

layered neural networks, where each layer is fully connected to the next layer, and there are no recurrent connections or feedforward connections that skip layers. Otherwise, neural network topologies that contain sparse connections, skip connections, feedback, weight matrices that do not match the size of the crossbars will lead to either (i) the need for multiple crossbars to represent the weight matrix between a set of pre- and postsynaptic neurons, (ii) a single crossbar representing portions of multiple weight matrices, and/or (iii) underutilization of the crossbars. In these cases, it is important to optimize the mapping of weights to the different crossbars in the neuromorphic system to optimize multiple objectives, such as speed, energy, and number of required crossbars. Several mapping algorithms have been proposed to satisfy these objectives. For example, in Ref. 151, the authors propose an algorithm to optimize mapping of neural networks to neuromorphic hardware under different constraints such as limited neuron fan-in and fan-out, which, in the context of NVM crossbar architectures leads to the optimum number of crossbars needed to implement a particular topology. In Ref. 152, a different mapping algorithm is proposed that instead focuses on improving the speed and energy consumption of neuromorphic SNN implementations based on NVM crossbars by optimizing neuron and synapse placement using graph partitioning algorithms. In a follow-up paper, the same authors focus on optimizing their algorithm to increase its speed for fast run-time mapping of SNNs to neuromorphic hardware.[153]

7 Future Directions and Conclusions

In this paper, we have provided a perspective on spike-based neuromorphic computing for brain-inspired vision. Research in neuromorphic computing continues to grow rapidly, with over 5000 articles related to or at least mentioning the topic in 2021 alone and over 53,000 total. (These numbers come from the number of search results for "neuromorphic" on Google Scholar.) The growing complexity of DNN models and associated energy requirements makes the alternative approach of neuromorphic computing particularly attractive for achieving AI on resource-constrained edge devices, such as mobile phones, sensors, satellites, etc. Neuromorphic computing's sustained growth is also supported by several conferences and workshops devoted to the topic such as the Telluride Neuromorphic Cognition Engineering Workshop, the Neuro-Inspired Computational Elements (NICE) conference, the International Conference on Neuromorphic Systems (ICONS), and the CapoCaccia Workshops, to name a few. The amount of activity in the neuromorphic community is not only encouraging but also highlights that neuromorphic computing, despite being several decades old, still offers a lot to be explored. We conclude this review with a discussion of some exciting future directions.

7.1 *Binarized Spike-Based Neuromorphic Hardware*

Recently, the concept of binary neural networks (BNNs) have been proposed, in which the weights and activations of the network take on binary values,[154] enabling extremely efficient hardware implementation. So far, however, this idea has limited evaluation in spike-based neuromorphic systems, with a few exceptions, such as Ref. 155, where the authors use binary SNNs based on temporal coding as an approach to increase speed and reduce computations. The potential of BNNs for spike-based hardware is that they significantly reduce the design complexity of neuron and synapse circuits. In particular, synapse circuits based on NVM technologies would be able to leverage NVM devices that have been optimized for memory designs with 1-bit resolution cells. Furthermore, expensive neuron circuits could be replaced with simplified threshold activations or even single-spike codes. However, training neural networks (spiking or nonspiking) with binary weights or activations becomes challenging on-chip because it requires a high-precision copy of neural network weights to be stored along with their binarized versions. This means that any efficiency gained by reducing the weight precision is lost after considering the memory requirements for training. While some stochastic techniques such as Ref. 156 eliminate the need for storing high-precision data during training, they have degraded performance. Therefore, the design of efficient on-chip training methods for BNNs and their extension to spiking BNNs is an open problem.

7.2 *Incorporation of Non-Neuronal Cells*

Neuroglial (glial) cells are just as numerous as neurons in the brain, yet they are typically not modeled in neuromorphic systems. Historically, glial cells such as astrocytes were thought to only perform support and maintenance functions without explicitly influencing the brain's computations.[23] Although, it has been shown that astrocytes play key roles in neural computation such as influence of synaptic activity and plasticity, as well as modulation of neuron activity.[157] However, the ways in which astrocyte behavior manifests to enable specific computational primitives is largely unknown. Even so, it is easy to imagine that the modulatory function of astrocytes could play a role in abilities such as continual learning without catastrophic forgetting, which is a key challenge in implementing lifelong learning systems.[158] For humans, lifelong learning is effortless. For example, we can easily learn new categories of objects without forgetting those that we learned previously. Replicating this behavior in neuromorphic hardware would have huge implications for the design of autonomous systems. Some limited work on integrating astrocyte models with neurons on neuromorphic hardware have been proposed. For example, in Ref. 159, it is shown that neuron–astrocyte interaction enhances the information processing capabilities of a neuromorphic system and in Ref. 160, the authors are able to model astrocytes' calcium ion dynamics in a VLSI circuit. However, to the authors' knowledge, there has not yet been any demonstration of a fundamentally new capability in neuromorphic hardware enabled by astrocytes. One potential avenue is to explore the astrocytes' regulation of energy in the brain as a source of inspiration for energy-efficient neuromorphic chip design.[40]

7.3 *Adversarial Robustness of Spike-Based Neuromorphic Systems*

Since 2013,[161] a growing group of researchers has been studying the brittleness of DNNs when applied to tasks such as visual object classification. Backpropagation learning in DNNs creates classification decision regions in the input space of the network which tend to have boundaries that lie close to the training and test data. As a result, small perturbations of an image (often imperceptible to a human observer) can cause the DNNs to misclassify the input with high confidence. These so-called adversarial examples represent a significant safety and security vulnerability for DNNs. Several theoretical analyses and defenses have been proposed in the literature,[162] but these are almost entirely from the algorithm perspective, without consideration of the underlying hardware. As neuromorphic computing becomes more mainstream, we will entrust more of our safety and security-critical applications to neuromorphic implementations of AI algorithms. As such, it is imperative that we understand the implications of hardware-specific behaviors on the robustness of these systems to adversarial examples. For example, device variability due to process variations or transient faults that occur in synapse and neuron circuits will have an effect on adversarial robustness. Some limited work in has been done in this area, such as,[163–167] but it is still an open question if and how spike-based neuromorphic hardware will exacerbate the already-brittle nature of neural network algorithms for vision.

In closing, we strongly believe that a key factor for the advancement of neuromorphic technologies (in general and for the future directions listed above) is the collaboration among researchers from a diverse set of disciplines, such as device physics, electrical and computer engineering, computer science, neuroscience, and psychology. New discoveries and innovations in all of these fields are required for us to improve our understanding of how the brain works, efficiently emulate its behaviors, and leverage that behavior to make transformational advances in AI and neuromorphic systems for the betterment of our society.

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv. Neural Inf. Process. Syst.* **25**, 1097–1105 (2012).
2. N. Srivastava et al., "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014).
3. V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Int. Conf. Mach. Learn.* (2010).

4. B. Rueckauer et al., "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Front. Neurosci.* **11**, 682 (2017).
5. G. E. Dahl et al., "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio Speech Lang. Process.* **20**(1), 30–42 (2012).
6. F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Twelfth Annu. Conf. Int. Speech Commun. Assoc.* (2011).
7. "Papers with code," https://paperswithcode.com/.
8. Y. Cheng et al., "Model compression and acceleration for deep neural networks: the principles, progress, and challenges," *IEEE Signal Process. Mag.* **35**(1), 126–136 (2018).
9. A. Amir et al., "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 7243–7252 (2017).
10. H. Rebecq et al., "High speed and high dynamic range video with an event camera," *IEEE Trans. Pattern Anal. Mach. Intell.* **43**, 1964–1980 (2019).
11. T.-J. Chin et al., "Star tracking using an event camera," in *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit. Workshops* (2019).
12. A. Jones et al., "A neuromorphic SLAM architecture using gated-memristive synapses," *Neurocomputing* **381**, 89–104 (2020).
13. M. Litzenberger et al., "Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor," in *IEEE Intell. Transp. Syst. Conf.*, IEEE, pp. 653–658 (2006).
14. C. Sung, H. Hwang, and I. K. Yoo, "Perspective: a review on memristive hardware for neuromorphic computation," *J. Appl. Phys.* **124**(15), 151903 (2018).
15. G. Gallego et al., "Event-based vision: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **44**(1), 154–180 (2022)..
16. J. A. Leñero-Bardallo, R. Carmona-Galán, and A. Rodrguez-Vázquez, "Applications of event-based image sensors: review and analysis," *Int. J. Circuit Theory Appl.* **46**(9), 1620–1630 (2018).
17. O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: a review," *IEEE Trans. Neural Networks Learn. Syst.* **31**(1), 4–23 (2020).
18. C. Frenkel, D. Bol, and G. Indiveri, "Bottom-up and top-down neural processing systems design: neuromorphic intelligence as the convergence of natural and artificial intelligence," arXiv:2106.01288 (2021).
19. K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature* **575**(7784), 607–617 (2019).
20. C. D. James et al., "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," *Biol. Inspired Cognit. Archit.* **19**, 49–64 (2017).
21. W. J. Gehring, "The evolution of vision," *Wiley Interdiscip. Rev. Dev. Biol.* **3**(1), 1–40 (2014).
22. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Henry Holt and Co., Inc., New York (1982).
23. E. R. Kandel et al., *Principles of Neural Science*, 6th ed., Vol. **4** (2021).
24. J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How does the brain solve visual object recognition?" *Neuron* **73**(3), 415–434 (2012).
25. L. Squire et al., *Fundamental Neuroscience*, Academic Press (2012).
26. A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.* **117**(4), 500–544 (1952).
27. R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophys. J.* **1**(6), 445–466 (1961).
28. J. Nagumo, S. Arimoto, and S. Yoshizawa, "An active pulse transmission line simulating nerve axon," *Proc. IRE* **50**(10), 2061–2070 (1962).
29. C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophys. J.* **35**(1), 193–213 (1981).
30. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.* **5**(4), 115–133 (1943).

31. L. A. Camuñas-Mesa, B. Linares-Barranco, and T. Serrano-Gotarredona, "Neuromorphic spiking neural networks and their memristor-CMOS hardware implementations," *Materials* **12**(17), 2745 (2019).
32. S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *Nature* **381**(6582), 520–522 (1996).
33. E. T. Rolls and M. J. Tovee, "Processing speed in the cerebral cortex and the neurophysiology of visual masking," *Proc. R. Soc. London Ser. B* **257**(1348), 9–15 (1994).
34. W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Netw.* **10**(9), 1659–1671 (1997).
35. E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.* **15**(5), 1063–1070 (2004).
36. R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *J. Neurophysiol.* **94**(5), 3637–3642 (2005).
37. W. Gerstner, R. Ritz, and J. L. Van Hemmen, "Why spikes? Hebbian learning and retrieval of time-resolved excitation patterns," *Biol. Cybern.* **69**(5), 503–515 (1993).
38. E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.* **14**(6), 1569–1572 (2003).
39. W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press (2002).
40. A. Fountain and C. Merkel, "Energy constraints improve liquid state machine performance," in *Int. Conf. Neuromorphic Syst.*, pp. 1–8 (2020).
41. J. Burroni et al., "Energetic constraints produce self-sustained oscillatory dynamics in neuronal networks," *Front. Neurosci.* **11**, 80 (2017).
42. P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, Computational Neuroscience Series (2001).
43. S. Panzeri et al., "Sensory neural codes using multiplexed temporal scales," *Trends Neurosci.* **33**(3), 111–120 (2010).
44. Y. Sagi et al., "Learning in the fast lane: new insights into neuroplasticity," *Neuron* **73**(6), 1195–1203 (2012).
45. X. Wang, X. Lin, and X. Dang, "Supervised learning in spiking neural networks: a review of algorithms and evaluations," *Neural Netw.* **125**, 258–280 (2020).
46. J. A. Pérez-Carrasco et al., "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets," *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(11), 2706–2719 (2013).
47. Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.* **113**(1), 54–66 (2015).
48. P. U. Diehl et al., "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Int. Joint Conf. Neural Networks*, IEEE, pp. 1–8 (2015).
49. B. Han, G. Srinivasan, and K. Roy, "RMP-SNN: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proc. IEEE/CVF Conf. Comput. Vision and Pattern Recognit.*, pp. 13558–13567 (2020).
50. A. Sengupta et al., "Going deeper in spiking neural networks: VGG and residual architectures," *Front. Neurosci.* **13**, 95 (2019).
51. B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 1–5 (2018).
52. S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing* **48**(1–4), 17–37 (2002).
53. J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Front. Neurosci.* **10**, 508 (2016).
54. G. Bellec et al., "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. 32nd Int. Conf. Neural Inform. Process. Syst. (NIPS'18)*, Curran Associates Inc., Red Hook, NY, pp. 795–805 (2018).
55. E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process Mag.* **36**(6), 51–63 (2019).

56. G. Datta et al., "Towards energy-efficient quantized deep spiking neural networks for hyperspectral image classification," arXiv:2107.11979 (2021).
57. E. O. Neftci et al., "Event-driven random back-propagation: enabling neuromorphic deep learning machines," *Front. Neurosci.* **11**, 324 (2017).
58. D. Kwon et al., "On-chip training spiking neural networks using approximated backpropagation with analog synaptic devices," *Front. Neurosci.* **14**, 423 (2020).
59. I. M. Comsa et al., "Temporal coding in spiking neural networks with alpha synaptic function," in *IEEE Int. Conf. Acoust., Speech and Signal Process.*, IEEE, pp. 8529–8533 (2020).
60. H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Networks Learn. Syst.* **29**(7), 3227–3235 (2018).
61. N. Rathi et al., "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," in *Int. Conf. Learn. Represent.* (2020).
62. D. Elbrecht et al., "Training spiking neural networks using combined learning approaches," in *IEEE Symp. Ser. Comput. Intell.*, IEEE, pp. 1995–2001 (2020).
63. D. O. Hebb, *The Organisation of Behaviour: A Neuropsychological Theory*, Science ed., John Wiley and Sons, Inc., New York (1949).
64. J. Hertz et al., "Introduction to the theory of neural computation," *Phys. Today* **44**(12), 70 (1991).
65. E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Biol.* **15**(3), 267–273 (1982).
66. E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex," *J. Neurosci.* **2**(1), 32–48 (1982).
67. G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.* **18**(24), 10464–10472 (1998).
68. P. Falez et al., "Unsupervised visual feature learning with spike-timing-dependent plasticity: how far are we from traditional feature learning approaches?" *Pattern Recognit.* **93**, 418–429 (2019).
69. Z. Wang et al., "Fully memristive neural networks for pattern classification with unsupervised learning," *Nat. Electron.* **1**(2), 137–145 (2018).
70. I. Boybat et al., "Neuromorphic computing with multi-memristive synapses," *Nat. Commun.* **9**(1), 2514 (2018).
71. A. Serb et al., "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," *Nat. Commun.* **7**(1), 12611 (2016).
72. E. Covi et al., "Analog memristive synapse in spiking networks implementing unsupervised learning," *Front. Neurosci.* **10**, 482 (2016).
73. M. Mozafari et al., "Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks," *Pattern Recognit.* **94**, 87–95 (2019).
74. C. S. Chane et al., "Event-based tone mapping for asynchronous time-based image sensor," *Front. Neurosci.* **10**, 391 (2016).
75. R. Sun et al., "Data-driven technology in event-based vision," *Complexity* **2021**, 6689337 (2021).
76. N. Bolger, A. Davis, and E. Rafaeli, "Diary methods: capturing life as it is lived," *Annual review of psychology* **54**(1), 579–616 (2003).
77. Y. Hu et al., "Ddd20 end-to-end event camera driving dataset: fusing frames and events with deep learning for improved steering prediction," in *IEEE 23rd Int. Conf. Intell. Transp. Syst.*, IEEE, pp. 1–6 (2020).
78. "The event-camera dataset and simulator: event-based data for pose estimation, visual odometry, and SLAM," http://rpg.ifi.uzh.ch/davis_data.html.
79. P. Lichtsteiner and T. Delbruck, "A 64x64 AER logarithmic temporal derivative silicon retina," in *Res. Microelectron. and Electron.*, IEEE, Vol. 2, pp. 202–205 (2005).
80. P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 30 mW asynchronous vision sensor that responds to relative intensity change," in *IEEE Int. Solid State Circuits Conf.-Dig. Tech. Papers*, IEEE, pp. 2060–2069 (2006).

81. P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits* **43**(2), 566–576 (2008).
82. J. P. Boettiger, "A comparative evaluation of the detection and tracking capability between novel event-based and conventional frame-based sensors" (2020).
83. C. Mead, "Adaptive retina," in *Analog VLSI Implementation of Neural Systems*, C. Mead and M. Ismail, Eds., pp. 239–246, Springer, Boston, MA (1989).
84. C. A. Mead and M. A. Mahowald, "A silicon model of early visual processing," *Neural Netw.* **1**(1), 91–97 (1988).
85. C. Mead, "A sensitive electronic photoreceptor," in *1985 Chapel Hill Conference on Very Large Scale Integration*, Computer Science Press, Rockville, MD, pp. 463–471 (1985).
86. T. Delbrueck and C. A. Mead, "Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits" California Institute of Technology, Pasadena, CA (Unpublished). https://authors.library.caltech.edu/60113/
87. T. Delbrück, "Investigations of analog VLSI visual transduction and motion processing," Diss., California Institute of Technology (1993).
88. C. Scheerlinck, "How to see with an event camera," PhD Thesis, College of Engineering and Computer Science, The Australian National University (2021).
89. T. Delbrück and C. A. Mead, "Analog VLSI adaptive logarithmic wide-dynamic-range photoreceptor," in *Proc. IEEE Int. Symp. Circuits and Syst.*, Vol. 4, pp. 339–342 (1994).
90. S. Furber, "Large-scale neuromorphic computing systems," *J. Neural Eng.* **13**(5), 051001 (2016).
91. C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley Longman Publishing Co., Inc. (1989).
92. G. Indiveri et al., "Neuromorphic silicon neuron circuits," *Front. Neurosci.* **5**, 73 (2011).
93. B. V. Benjamin et al., "Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE* **102**(5), 699–716 (2014).
94. A. Neckar et al., "Braindrop: a mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proc. IEEE* **107**(1), 144–164 (2019).
95. J. Schemmel et al., "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 1947–1950 (2010).
96. S. Billaudelle et al., "Structural plasticity on an accelerated analog neuromorphic hardware system," *Neural Netw.* **133**, 11–20 (2021).
97. S. B. Furber et al., "The spinnaker project," *Proc. IEEE* **102**(5), 652–665 (2014).
98. G. Dikov et al., "Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware," *Lect. Notes Comput. Sci.* **10384**, 119–137 (2017).
99. G. Haessig et al., "Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system," *IEEE Trans. Biomed. Circuits Syst.* **12**(4), 860–870 (2018).
100. A. Glover et al., "ATIS+ spinnaker: a fully event-based visual tracking demonstration," arXiv:1912.01320 (2019).
101. T. Serrano-Gotarredona et al., "ConvNets experiments on spinnaker," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 2405–2408 (2015).
102. A. Andreopoulos et al., "A low power, high throughput, fully event-based stereo system," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognit.*, pp. 7532–7542 (2018).
103. S. Kim et al., "Spiking-YOLO: spiking neural network for energy-efficient object detection," in *Proc. AAAI Conf. Artif. Intell.*, Vol. 34, pp. 11270–11277 (2020).
104. M. Davies et al., "Loihi: a neuromorphic manycore processor with on-chip learning," *IEEE Micro* **38**(1), 82–99 (2018).
105. M. Davies et al., "Advancing neuromorphic computing with Loihi: a survey of results and outlook," *Proc. IEEE* **109**(5), 911–934 (2021).
106. C. Frenkel et al., "A 0.086-mm^2 12.7-pj/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS," *IEEE Trans. Biomed. Circuits Syst.* **13**(1), 145–158 (2019).
107. C. Frenkel, J.-D. Legat, and D. Bol, "MorphIC: a 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning," *IEEE Trans. Biomed. Circuits Syst.* **13**(5), 999–1010 (2019).

108. N. Qiao et al., "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Front. Neurosci.* **9**, 141 (2015).
109. S. Moradi et al., "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.* **12**(1), 106–122 (2018).
110. N. Zheng and P. Mazumder, "A low-power hardware architecture for on-line supervised learning in multi-layer spiking neural networks," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 1–5 (2018).
111. K. Roy et al., "In-memory computing in emerging memory technologies for machine learning: an overview," in *57th ACM/IEEE Des. Autom. Conf.*, IEEE, pp. 1–6 (2020).
112. V. Saxena, "Mixed-signal neuromorphic computing circuits using hybrid CMOS-RRAM integration," *IEEE Trans. Circuits Syst. II Express Briefs* **68**, 581–586 (2021).
113. B.-S. Lee et al., "Nanoscale nuclei in phase change materials: origin of different crystallization mechanisms of $Ge_2Sb_2Te_5$ and AgInSbTe," *J. Appl. Phys.* **115**(6), 063506 (2014).
114. A. Gholami et al., "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*, Chapman and Hall/CRC, pp. 291–326 (2021).
115. J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Front. Neurosci.* **7**, 118 (2013).
116. X. Guo et al., "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded nor flash memory technology," in *IEEE Int. Electron Devices Meeting*, IEEE, pp. 5–6 (2017).
117. S. Dutta et al., "Supervised learning in all FeFET-based spiking neural network: opportunities and challenges," *Front. Neurosci.* **14**, 634 (2020).
118. P. Chi et al., "Prime: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," *ACM SIGARCH Comput. Architect. News* **44**(3), 27–39 (2016).
119. M. Cheng et al., "Time: a training-in-memory architecture for memristor-based deep neural networks," in *54th ACM/EDAC/IEEE Design Autom. Conf.*, IEEE, pp. 1–6 (2017).
120. A. Ankit et al., "PUMA: a programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. Twenty-Fourth Int. Conf. Architect. Support Programm. Languages and Oper. Syst.*, pp. 715–731 (2019).
121. F. Su et al., "A 462GOPs/J RRAM-based nonvolatile intelligent processor for energy harvesting IoE system featuring nonvolatile logics and processing-in-memory," in *Symp. VLSI Technol.*, IEEE, pp. T260–T261 (2017).
122. C. Merkel, "Current-mode memristor crossbars for neuromorphic computing," in *Proc. 7th Annu. Neuro-Inspired Comput. Elements Workshop*, pp. 1–6 (2019).
123. C. Li et al., "Analogue signal and image processing with large memristor crossbars," *Nat. Electron.* **1**(1), 52–59 (2018).
124. M. A. Zidan et al., "Memristor-based memory: the sneak paths problem and solutions," *Microelectron. J.* **44**(2), 176–183 (2013).
125. A. Dozortsev, I. Goldshtein, and S. Kvatinsky, "Analysis of the row grounding technique in a memristor-based crossbar array," *Int. J. Circuit Theory Appl.* **46**(1), 122–137 (2018).
126. M. A. Zidan et al., "Memristor multiport readout: a closed-form solution for sneak paths," *IEEE Trans. Nanotechnol.* **13**(2), 274–282 (2014).
127. L. Shi et al., "Research progress on solutions to the sneak path issue in memristor crossbar arrays," *Nanoscale Adv.* **2**(5), 1811–1827 (2020).
128. H. Mulaosmanovic et al., "Novel ferroelectric FET based synapse for neuromorphic systems," in *Symp. VLSI Technol.*, IEEE, pp. T176–T177 (2017).
129. P. Yao et al., "Face classification using electronic synapses," *Nat. Commun.* **8**(1), 15199 (2017).
130. F. M. Bayat et al., "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nat. Commun.* **9**(1), 2331 (2018).
131. X. Wu et al., "A CMOS spiking neuron for brain-inspired neural networks with resistive synapses and in situ learning," *IEEE Trans. Circuits Syst. II* **62**(11), 1088–1092 (2015).

132. C. D. Schuman et al., "A survey of neuromorphic computing and neural networks in hardware," arXiv:1705.06963 (2017).
133. A. Valentian et al., "Fully integrated spiking neural network with analog neurons and RRAM synapses," in *IEEE Int. Electron Devices Meeting*, IEEE, pp. 14–23 (2019).
134. C. Du et al., "Biorealistic implementation of synaptic functions with oxide memristors through internal ionic dynamics," *Adv. Funct. Mater.* **25**(27), 4290–4299 (2015).
135. T. Hirtzlin et al., "Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays," *Front. Neurosci.* **13**, 1383 (2020).
136. G. H. Kim et al., "Four-bits-per-cell operation in an HfO_2-based resistive switching device," *Small* **13**(40), 1701781 (2017).
137. R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip training of memristor based deep neural networks," in *Int. Joint Conf. Neural Networks*, IEEE, pp. 3527–3534 (2017).
138. A. M. Zyarah et al., "ZIKSA: on-chip learning accelerator with memristor crossbars for multilevel neural networks," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 1–4 (2017).
139. A. M. Zyarah, N. Soures, and D. Kudithipudi, "On-device learning in memristor spiking neural networks," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 1–5 (2018).
140. C. Merkel and D. Kudithipudi, "A stochastic learning algorithm for neuromemristive systems," in *27th IEEE Int. Syst.-on-Chip Conf.*, IEEE, pp. 359–364 (2014).
141. E. Covi et al., "HfO_2-based memristors for neuromorphic applications," in *IEEE Int. Symp. Circuits and Syst.*, IEEE, pp. 393–396 (2016).
142. C. Merkel and D. Kudithipudi, "Unsupervised learning in neuromemristive systems," in *Natl. Aerosp. and Electron. Conf.*, IEEE, pp. 336–338 (2015).
143. C. Merkel et al., "Neuromemristive systems: boosting efficiency through brain-inspired computing," *Computer* **49**(10), 56–64 (2016).
144. D. Kuzum, S. Yu, and H. P. Wong, "Synaptic electronics: materials, devices and applications," *Nanotechnology* **24**(38), 382001 (2013).
145. Y. Zhang et al., "Brain-inspired computing with memristors: challenges in devices, circuits, and systems," *Appl. Phys. Rev.* **7**(1), 011308 (2020).
146. F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and in situ training," *Nat. Commun.* **4**(1), 1–7 (2013).
147. C. Merkel and D. Kudithipudi, "Comparison of off-chip training methods for neuromemristive systems," in *28th Int. Conf. VLSI Design*, IEEE, pp. 99–104 (2015).
148. P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," *Nature* **577**(7792), 641–646 (2020).
149. Q. Xu et al., "Reliability-driven neuromorphic computing systems design," in *Des., Autom. and Test Eur. Conf. and Exhib.*, IEEE, pp. 1586–1591 (2021).
150. C.-Y. Chen and K. Chakrabarty, "Efficient identification of critical faults in memristor-based inferencing accelerators," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2021).
151. Y. Ji et al., "NEUTRAMS: neural network transformation and co-design under neuromorphic hardware constraints," in *49th Annu. IEEE/ACM Int. Symp. Microarchitect.*, IEEE, pp. 1–13 (2016).
152. A. Balaji et al., "Mapping spiking neural networks to neuromorphic hardware," *IEEE Trans. Very Large Scale Integr. Syst.* **28**(1), 76–86 (2020).
153. A. Balaji et al., "Run-time mapping of spiking neural networks to neuromorphic hardware," *J. Signal Process. Syst.* **92**(11), 1293–1302 (2020).
154. W. Zhao et al., "A review of recent advances of binary neural networks for edge computing," *IEEE J. Miniaturization Air Space Syst.* **2**(1), 25–35 (2020).
155. S. R. Kheradpisheh et al., "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.* **99**, 56–67 (2018).
156. G. Li et al., "Training deep neural networks with discrete state transition," *Neurocomputing* **272**, 154–162 (2018).
157. M. Santello, N. Toni, and A. Volterra, "Astrocyte function from information processing to cognition and cognitive impairment," *Nat. Neurosci.* **22**(2), 154–166 (2019).
158. G. I. Parisi et al., "Continual lifelong learning with neural networks: a review," *Neural Netw.* **113**, 54–71 (2019).

159. S. Nazari et al., "A digital implementation of neuron–astrocyte interaction for neuromorphic applications," *Neural Netw.* **66**, 79–90 (2015).
160. G. Karimi et al., "A neuromorphic real-time VLSI design of Ca^{2+} dynamic in an astrocyte," *Neurocomputing* **272**, 197–203 (2018).
161. C. Szegedy et al., "Intriguing properties of neural networks," in *Int. Conf. Learn. Represent.* (2013).
162. X. Yuan et al., "Adversarial examples: attacks and defenses for deep learning," *IEEE Trans. Neural Networks Learn. Syst.* **30**(9), 2805–2824 (2019).
163. S. Barve et al., "Adversarial attack mitigation approaches using RRAM-neuromorphic architectures," in *Proc. 2021 on Great Lakes Symp. VLSI*, pp. 201–206 (2021).
164. H.-P. Cheng et al., "Adverquil: an efficient adversarial detection and alleviation technique for black-box neuromorphic computing systems," in *Proc. 24th Asia and South Pacific Des. Autom. Conf.*, pp. 518–525 (2019).
165. M. Gorsline, J. Smith, and C. Merkel, "On the adversarial robustness of quantized neural networks," in *Proc. Great Lakes Symp. VLSI*, pp. 189–194 (2021).
166. T. Li and C. Merkel, "Model extraction and adversarial attacks on neural networks using switching power information," in *Int. Conf. Artif. Neural Networks* (2021).
167. A. S. Rakin et al., "T-BFA: targeted bit-flip adversarial weight attack," *IEEE Trans. Pattern Anal. Mach. Intell.* (2021).

Hagar Hendy received her MSc degree from the Faculty of Engineering, Ain Shams University, Cairo, Egypt, where her research focused on designing read/write circuits for multibit memristor memories. She is currently working toward her PhD in electrical and computer engineering at Rochester Institute of Technology. Her research focuses on neuromorphic circuit design based on memristor devices.

Cory Merkel is an assistant professor with the Department of Computer Engineering, Rochester Institute of Technology (RIT). He received his BS and MS degrees in computer engineering in 2011 and his PhD in microsystems engineering in 2015 from RIT. From 2016 to 2018, he was a research electronics engineer with the Information Directorate, Air Force Research Lab. His current research focuses on mapping of AI algorithms, primarily artificial neural networks, to mixed-signal hardware, design of brain-inspired computing systems using emerging technologies, and trustworthy neuromorphic systems.