REVIEW ARTICLE | SEPTEMBER 23 2024

When in-memory computing meets spiking neural networks—A perspective on device-circuit-system-and-algorithm co-design

Abhishek Moitra ; Abhiroop Bhattacharjee ; Yuhang Li ; Youngeun Kim ; Priyadarshini Panda

Appl. Phys. Rev. 11, 031325 (2024)
https://doi.org/10.1063/5.0211040

Applied Physics Reviews REVIEW pubs.aip.org/aip/are

When in-memory computing meets spiking neural networks—A perspective on device-circuit-system-and-algorithm co-design

Cite as: Appl. Phys. Rev. **11**, 031325 (2024); doi: 10.1063/5.0211040
Submitted: 28 March 2024 · Accepted: 30 August 2024 ·
Published Online: 23 September 2024

Abhishek Moitra,[a] Abhiroop Bhattacharjee,[a] Yuhang Li,[b] Youngeun Kim,[c] and Priyadarshini Panda[d]

AFFILIATIONS

Department of Electrical Engineering, Yale University, New Haven, Connecticut 06511, USA

[a]Authors to whom correspondence should be addressed: abhishek.moitra@yale.edu, and abhiroop.bhattacharjee@yale.edu
[b]Electronic mail: yuhang.li@yale.edu
[c]Electronic mail: youngeun.kim@yale.edu
[d]Electronic mail: priya.panda@yale.edu

ABSTRACT

This review explores the intersection of bio-plausible artificial intelligence in the form of spiking neural networks (SNNs) with the analog in-memory computing (IMC) domain, highlighting their collective potential for low-power edge computing environments. Through detailed investigation at the device, circuit, and system levels, we highlight the pivotal synergies between SNNs and IMC architectures. Additionally, we emphasize the critical need for comprehensive system-level analyses, considering the inter-dependencies among algorithms, devices, circuit, and system parameters, crucial for optimal performance. An in-depth analysis leads to the identification of key system-level bottlenecks arising from device limitations, which can be addressed using SNN-specific algorithm–hardware co-design techniques. This review underscores the imperative for holistic device to system design-space co-exploration, highlighting the critical aspects of hardware and algorithm research endeavors for low-power neuromorphic solutions.

TABLE OF CONTENTS

I. INTRODUCTION

Artificial intelligence (AI) has been at the forefront of technological innovation over the past decade. From the development of deep convolutional neural networks[1,2] that have revolutionized computer vision to the emergence of transformers[3] and large language models[4] that have transformed natural language processing, each generation of AI algorithm represents a significant leap in our ability to harness the power of data. The growth of AI is mainly attributed to the scale-up of high-power, server-class computing machines such as graphics processing units (GPUs).[5] However, GPUs draw a substantial amount of power, leading to increased operational costs and a larger carbon footprint.[5,6] As AI aims to become more ubiquitous and user-centric, there is a growing need for low-power AI algorithms and hardware accelerators. This shift is essential to move away from the current trend of expecting increased intelligence merely by scaling up compute/memory resources, which is neither sustainable nor practical for widespread deployment. However, this vision stands in contrast to the current algorithm and hardware progress trajectory, where the computational needs of AI algorithms are doubling every two months, far surpassing Moore's law of silicon scaling by a considerable margin.[7]

To this end, neuromorphic computing algorithms such as spiking neural networks (SNNs) leveraging brain-like computations have emerged as a suitable candidate toward low-power AI implementation.[8–10] An SNN contains numerous leaky-integrate-and-fire (LIF) neurons that store the spatiotemporal information in the form of membrane potential values over multiple timesteps. The information from one neuron is relayed to another neuron in the form of binary spikes. Overall, due to the sparse event-driven binary processing capabilities, SNNs show promise for low-power edge computing applications, such as in-sensor processing[11–14] and embedded intelligence,[15–20] among others. In fact, SNNs are increasingly being embraced by different industries for commercial products in image classification,[21] optimization,[22] agriculture,[23] and autonomous driving,[24] signaling widespread adoption and innovation potential. In recent years, SNNs have achieved comparable accuracy with standard artificial neural networks (ANNs) in large-scale tasks such as image classification on the Imagenet-1K dataset.[25,26] This has necessitated SNNs to scale up in terms of parameter count requiring significant compute and memory resources. Thus, their implementation on the emerging low-power hardware acceleration paradigm, in-memory computing (IMC), shows great promise as IMC facilitates highly parallel computation with high-memory bandwidth.

Traditional von-Neumann style accelerators such as graphics processing units (GPUs) and tensor processing units (TPUs) suffer from "memory wall bottleneck" owing to the heavy data movement (specifically, weights) between memory and compute units corresponding to the dot-product operations in neural networks.[27,28] IMC with nonvolatile memories facilitates analog dot-product operations while keeping the weights stationary on crossbars, thereby reducing the weight movement bottleneck significantly.[29,30] Furthermore, the costly digital multiplier–accumulator circuits of von-Neumann accelerators are reduced to analog crossbar operation on IMC leading to energy and area efficiency. Importantly, SNNs exhibit tight synergies with IMC hardware. Due to the high spike sparsity (90% across different layers[31]) and binary spike computations, SNNs implemented on IMC require low peripheral and data communication overhead that yield low-power and high-throughput benefits.[7,32]

Figure 1 illustrates the prevailing landscape of research endeavors spanning device, circuit, and algorithm within the realm of SNN research. On the algorithmic front, scholars have directed their effort toward harnessing spatiotemporal computation, exploiting biological LIF neuron functionality, and optimizing spike sparsity to enhance the efficiency of SNN algorithms.[9,33,34] Meanwhile, within the domain of devices and circuits, the community has prioritized objectives such as achieving multi-level conductance, mitigating device vulnerabilities, and optimizing crossbar connections, among other pursuits.[35–38] However, to date, these research endeavors have largely been independent of one another. There exists a critical need to integrate various device, circuit, and algorithmic parameters with system-level considerations to realize truly optimal solutions. A comprehensive system-level inquiry holds the potential to bridge the gap between device-circuit and algorithmic-level innovations. In pursuit of this objective, the present review delineates the key bottlenecks and opportunities associated with implementing SNNs on emerging IMC architectures. With this, the review furnishes strategic guidance to diverse research communities on innovations for optimal implementation of SNNs on IMC architectures.

In this review, we first discuss the algorithmic efficiencies and current state-of-the-art training algorithms for SNNs. Additionally, we highlight the widespread application space of SNNs. Thereafter, we highlight the existent synergies between SNNs and IMC architectures that make them a suitable candidate for low-power edge computing. Following this, we motivate the need for end-to-end device, circuit, and system-level analyses to understand the challenges of implementing SNNs on IMC platforms. We will also highlight the recent SNN-aware co-design strategies that can overcome the pressing bottlenecks. Finally, we highlight the key questions that lie ahead in the SNN–IMC research. We discuss the device and system-level parameters that are crucial toward SNN inference with future emphasis on sustainable online learning.

II. SNN ALGORITHM AND APPLICATION SPACE

A. Inherent efficiencies in SNNs

SNNs inherently possess several key efficiencies that are critical toward low-power edge implementation.

1. Binary spike processing

Taking cues from the brain, SNNs perform binary spike-driven data processing over multiple timesteps. As a result, multiply-and-accumulate (MAC) operations are merely reduced to efficient accumulation operations.[31,39]

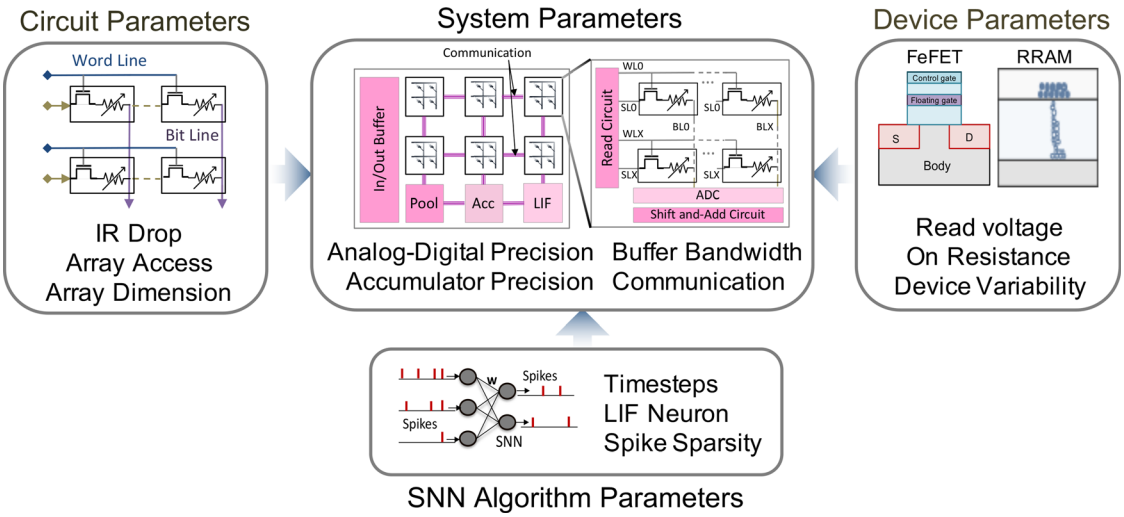

FIG. 1. Landscape of the spiking neural network (SNN) algorithm, in-memory computing (IMC) device, circuit, and system parameters. In order to reach fully optimal SNN–IMC implementations, there is a critical need to consider the existing co-dependencies between IMC device, circuit, system, and SNN algorithm parameters. LIF denotes leaky-integrate and fire neuron, a fundamental non-linear activation unit in SNNs. Relevant parameters for each domain that underlie hardware metrics such as performance, latency, energy efficiency, area, and power are mentioned.

2. Spatiotemporal complexity

At each timestep, the input spikes and SNN weights undergo spatial convolution yielding dot-product outputs. SNNs use a special non-linearity function called leaky-integrate and fire (LIF).[9,33] The dynamics of an LIF neuron is shown in Eq. (1). For neuron i, the membrane potential U is charged by the weighted summation of spikes from the previous layer's neuron j at every timestep t. Also, the leak factor $\lambda \in (0, 1)$ facilitates temporal leakage of the membrane potential

$$U_i^t = \lambda U_i^{t-1} + \sum_j w_{ij} o_j^t. \tag{1}$$

If at any timestep, the value U exceeds a particular threshold θ, the neuron generates a spike output and vice versa as shown in the following equation:

$$o_i^t = \begin{cases} 1, & \text{if} \quad U_i^t > \theta, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

This has been described in Fig. 2. It is worth noting that the number of timesteps for processing the neuronal dynamics will eventually determine the overall performance of an SNN.[40,41] Generally, SNNs with high timesteps yield better accuracy than that of SNNs with less timesteps. However, larger timesteps also translate to higher latency and energy consumption on hardware. As we will see later (in Fig. 8), timesteps become a critical control knob to determine the overall energy-vs-accuracy trade-off while designing SNNs.

3. Data sparsity

The LIF neuron activation yields high spike sparsity. This means that SNNs can represent data with very few spikes. At any given timestep, around 90% of the neurons in an SNN are not spiking. Compared to ANNs with the ReLU activation, SNNs exhibit at least 30%–40% higher neuronal sparsity.[9,33]

4. Event-driven computation

Due to the high spike sparsity, leveraging event-driven computation and communication can significantly improve the energy efficiency of hardware accelerators.[31,42,43]

B. State-of-the-art SNN training algorithms

In this section, we cover SNN training algorithms, including conventional learning algorithms like unsupervised Hebbian learning or spike timing-dependent plasticity (STDP), ANN–SNN conversion, and modern learning algorithms like backpropagation through time (BPTT). We will discuss the scalability and practicality of each algorithm.

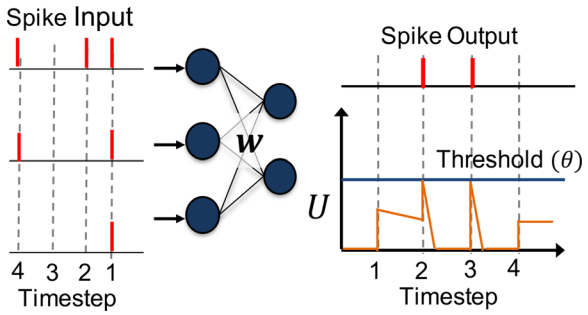

FIG. 2. Figure showing the functioning of an SNN. Input spikes are sent to the SNN across multiple timesteps. These binary spikes get multiplied with SNN weights (w) to generate multiply-and-accumulate (MAC) values which charge the membrane potential value U over multiple timesteps. At any time step, if the membrane potential exceeds a pre-defined threshold (θ), a spike output is generated.

1. Conventional learning algorithms

a. STDP learning. The spike timing-dependent plasticity[44] can be viewed as a spike-based formulation of a Hebbian learning rule,[45] where the synaptic weight is updated based on the tight temporal correlation of the firing time of the input and the output spikes. With STDP, if the presynaptic neuron fires briefly before the postsynaptic neuron (i.e., if the output spike timing is shortly after the input spike timing), the weight connecting them is strengthened. Otherwise, if the timing is reversed, the weight connection is weakened. This strengthening and weakening is called long-term potentiation (LTP) and long-term depression (LTD), respectively. STDP is a fully unsupervised learning algorithm that does not need a loss objective to update the weight neurons. Therefore, the SNNs trained with STDP are usually limited to thousands of neurons, which limits the scalability of this method.[46,47] Recent advances[48] propose to incorporate STDP with supervised learning to increase their scalability to complex tasks.

b. ANN–SNN conversion. Given that ANN training is easier than SNN training, a straightforward way to obtain the SNN is to first train an ANN with the same architecture and convert the neurons into spiking neurons. The conversion process typically involves finding the optimal threshold values [θ in Eq. (2)] for the membrane potential of the spiking neurons and scaling of the weights such that the spike rate in SNNs match the floating-point outputs of ANNs.[49–51] In a different ANN–SNN conversion method, Han *et al.*[52] achieved improved convergence and higher accuracy for converted SNNs by performing ANN–SNN conversion without resetting the LIF neuron. The ANN–SNN conversion shares several advantages compared to direct training of SNNs. For example, ANN training is easy to implement since the computing hardware (e.g., GPUs) and the deep-learning library (e.g., PyTorch) are well-established. In addition, the conversion process is also simple as it only involves changing the neuron type of the model. However, this method also incurs several disadvantages: (1) The converted SNN requires significantly larger timesteps to realize the original ANN performance.[9] Large timesteps will translate to high latency or energy consumption on hardware implementation. (2) The converted SNN shares the same architecture with the ANNs, which is not tailored for the spike-based mechanism and does not fully utilize the temporal dynamics of SNNs.[34]

2. Backpropagation through time

Previous methods like conversion or unsupervised learning suffer from either large timesteps or low scalability. The BPTT algorithm can address these two challenges. BPTT usually trains an SNN from scratch where the gradients are computed in a time step-unrolled computation graph [see Fig. 3(a)]. Formally, the gradient of the weight w_{ij}, denoted by Δw_{ij}, is accumulated over T timesteps as follows:

$$\Delta w_{ij} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}}{\partial o_i^t} \frac{\partial o_i^t}{\partial U_i^t} \frac{\partial U_i^t}{\partial w_{ij}}, \tag{3}$$

where $\mathscr{L}$ is the loss function being optimized. In the case of image classification, categorical cross-entropy loss is widely used. The challenge of applying gradient descent in SNNs is that the spike function returns zero gradient almost everywhere because of the thresholding function. Surrogate gradient descent[54–58] overcomes this problem by approximating the spike function into piece-wise linear, fast sigmoid, or exponential function.[59] For instance, the surrogate gradient descent method using a piece-wise linear approximation is defined as follows:

$$\frac{\partial o_i^t}{\partial U_i^t} = \xi \max\left\{0, 1 - \left|\frac{U_i^t - \theta}{\theta}\right|\right\}, \tag{4}$$

where ξ is a decay factor for back-propagated gradients and θ is the threshold value. The hyperparameter ξ should be set based on the total number of timesteps T. BPTT-based SNN reaches state-of-the-art accuracy on various tasks such as image recognition and event data processing at fewer timesteps. Hybrid training[60] combines BPTT and ANN–SNN conversion in order to achieve higher accuracy. In Fig. 3(b), we illustrate the accuracy of various SNN training methods over the last decade. Evidently, BPTT-based training exhibits high accuracy while scaling to large-scale datasets such as ImageNet at low time step overhead.

C. Application space for SNNs

This section reviews the recent academic studies and commercial application space of SNNs (summarized in Table I).

1. In-sensor processing and low-power healthcare

Due to their inherently sparse and binary spike representation, SNNs effectively reduce the bandwidth requirements for inter-chip interfacing. Works by Shaaban *et al.*[11] and MacLean *et al.*[12] employed efficient time-domain processing to replace computationally intensive preprocessing steps, while Zhou *et al.*[13] and Barchi *et al.*[14] directly interfaced sensors with SNNs for in-sensor processing.

The low-power nature of SNNs has also been leveraged in wearable healthcare devices. NeuroCARE[74] and Mosaic[73] offered tailored neuromorphic healthcare frameworks. Bian and Magno[16] and Li *et al.*[15] utilized SNNs for human activity recognition in wearables, and Tanzarella *et al.*[17] detected spinal motor neuron activity. Additionally, SNNs in brain–computer interface (BCI) applications, explored by Gong *et al.*[18] and Feng *et al.*,[78] leverage their energy efficiency for electroencephalography (EEG) analysis.

2. Emergence of event-driven and spike cameras

Recent research in emerging vision cameras has broadened the applicability of SNNs, including object detection tasks. Initiatives like spiking YOLO[10] pioneered ANN–SNN conversion, enhancing object detection efficiency across various datasets. Subsequent works, like Su *et al.*,[61] achieved state-of-the-art object detection via full-scale SNN training. In automotive applications, Lopez *et al.*[62] effectively utilized SNNs' sparse data representations for artifact detection. Salvatore *et al.*[63] demonstrated SNN robustness against noise and their effectiveness in satellite detection using event cameras. Amir *et al.*,[66] Maro *et al.*,[67] and Vasudevan *et al.*[68] have performed gesture recognition using event cameras, while Amir *et al.*[66] implemented their algorithm on the event-driven TrueNorth[21] neuromorphic processor, Maro *et al.*[67] implement their algorithm on an android smartphone. The DvsGesture, NavGesture, and SL-Animals-DVS datasets proposed by Amir *et al.*,[66] Maro *et al.*,[67] and Vasudevan *et al.*,[68] respectively, can serve as temporal datasets for benchmarking SNNs. Moreover, studies

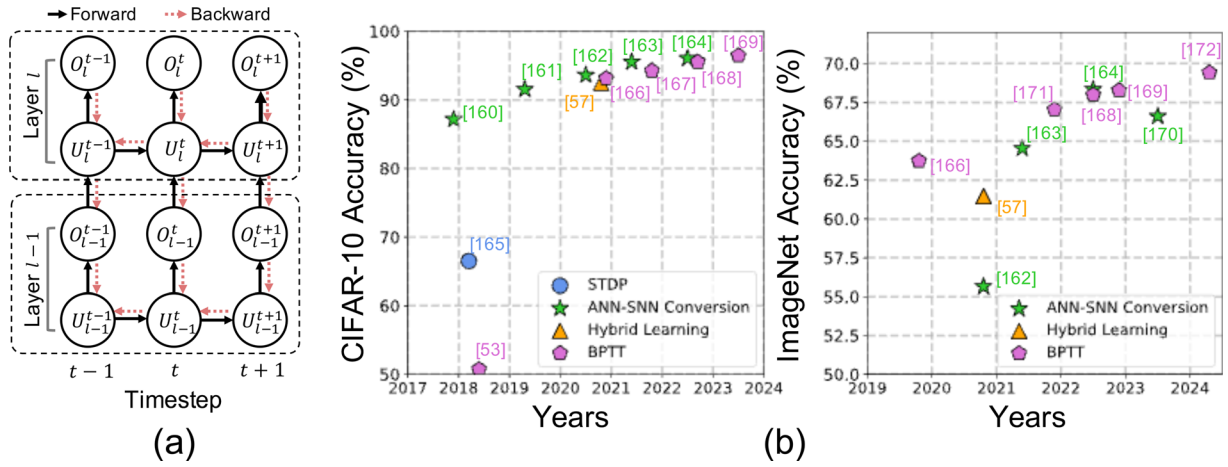

FIG. 3. (a) Training computation graph for the BPTT algorithm. The gradients pass through different layers and timesteps. (b) SNN accuracy on image classification tasks over the years. We show accuracy on two widely used benchmarks: CIFAR10 (Ref. 53) and ImageNet,[25] that comprise of 50 000, 1.2×10^6 images with ten classes and 1000 classes, respectively. SNNs trained on BPTT can achieve high accuracy while scaling to large-scale datasets at low timesteps.

such as Yang *et al.*[64] and Zheng *et al.*[65] leveraged event cameras' spike-driven nature for optical flow estimation.

Recent studies explore advanced spike cameras due to limitations of conventional RGB and event cameras. Works by Zhai *et al.*[69] and Chen *et al.*[70] utilize SNNs for optical flow estimation and image reconstruction, showcasing spike cameras as the new standard in vision sensing technology. Additionally, Xia *et al.*,[71] Zhai *et al.*,[69] and Chen *et al.*[70] demonstrated unsupervised SNN training for spike camera applications, while Zhao *et al.*[72] emphasized spike cameras' potential for high-speed object tracking, indicating ongoing advancements in spike camera technology.

TABLE I. Table showing the academic and industry application space of SNNs.

Academic studies	Efficient sensing	Radar data inference[11]
		Depth estimation[12]
		In-sensor processing[13,14]
	RGB cameras	Object detection[10,61]
		Automotive artifact detection[62]
	Event cameras	Satellite detection[63]
		Optical flow estimation[64,65]
		Gesture recognition[66–68]
	High-speed spike cameras	Optical flow estimation[69–71]
		Object tracking[65,72]
	Wearable healthcare	Human activity recognition[15–17]
		Anomaly detection[73,74]
		Flexible electronics[75]
		Brain–computer interface[18–20]
Commercial applications	Image classification	IBM TrueNorth[21]
	Optimization	Intel Loihi[22]
	SLAM and odometry	GrayScale-AI[76]
	Agriculture and healthcare	Andante[23]
	Autonomous driving	TSST[24]
	General intelligence	ORBAI[77]

3. Industry adoption of SNNs

Industry leaders are increasingly embracing SNN-based solutions for product design across diverse sectors. IBM utilizes SNNs for image classification and detection through their TrueNorth[21] platform. Intel leverages its Loihi chip[22] for optimization tasks, harnessing the power of SNNs. GrayScale-AI[76] employs SNNs for simultaneous localization and mapping (SLAM) and visual odometry applications, while Andante[23] focuses on edge computing solutions for agriculture. In the healthcare sector, Andante[23] is employing SNNs for glucose monitoring and ultrasound image processing. TSST[24] integrates SNNs with ferroelectric field-effect transistors (FeFETs) for autonomous driving applications, enhancing safety and efficiency. ORBAI[77] capitalizes on SNNs to emulate human brain characteristics, such as associativity and problem-solving, in the development of artificial general intelligence (AGI). Notably, companies like Apple and Mercedes are also exploring the potential of SNNs in various product design initiatives, signaling a widespread adoption of this innovative technology across industries.

III. IMC ACCELERATORS FOR SNNS

A. von-Neumann and IMC accelerators

In this section, we discuss the differences between von-Neumann and IMC architectures for inference applications. Note the discussion in Secs. III and IV will focus on inference, and Sec. V will highlight some opportunities and challenges for training with IMC–SNNs.

1. von-Neumann accelerators

Traditional von-Neumann AI accelerators [shown in Fig. 4(a)], such as GPUs and TPUs, contain an array of processing elements

(PE).[27,28,79] Each PE contains multipliers and accumulators that facilitate multi-bit MAC operations. For MAC operations, first, the weights/activation values are fetched from the off-chip DRAM memory to the input cache. Next, these values are transferred to the scratch pads of the PEs and the output is stored in the output cache and then sent back to the off-chip DRAM. During the inference of most modern deep-learning networks such as ANNs and SNNs, there is a significant data exchange (in form of weights, activations, and MAC outputs) between the DRAM and on-chip memories. The continual data movements and the constrained memory bandwidth contribute to the memory wall bottleneck in von-Neumann architectures.[29,30] Additionally, the MAC computation occurs in a cycle-to-cycle fashion. These factors degrade the throughput and energy efficiency of von-Neumann accelerators in edge computing scenarios.

2. IMC dot-product accelerators

To overcome the memory wall bottleneck in von-Neumann computing, IMC [shown in Fig. 4(b)] architectures co-locate the computation and memory units.[29,30] IMC architectures feature 2D memristive crossbars, with nonvolatile memory (NVM) devices situated at the cross-points. The NVM devices are interfaced in series with access transistors in a 1T–1R configuration to prevent sneak path currents in the crossbars.[80,81] Some NVM devices predominantly used are phase change memory (PCM),[82] resistive random-access memory (RRAM),[83] spin-torque-transfer magnetic RAM (STT–MRAM),[84] and ferroelectric field-effect transistor (FeFET).[85] All the weights of a neural network are stored on the crossbar encoded as synaptic conductances in the NVM devices. This eliminates the weight specific data movement between memories as observed in von-Neumann architectures.

For MAC computations, the digital inputs are converted to analog voltages by the digital-to-analog converter (DAC) and sent along the crossbar rows (or select lines). These voltages get multiplied with the device conductance using Ohm's law yielding currents, which get accumulated over the crossbar column (or bit line) according to Kirchoff's current law. The column currents represent the MAC operation result between inputs and weights. The analog-to-digital converter (ADC) converts column currents into digital outputs. Due to the analog nature of computing, IMC architectures can facilitate highly parallel MAC computations per cycle in an energy- and area-efficient manner.

B. Standard hardware evaluation metrics

In evaluating the efficacy of AI hardware accelerators, several key metrics are paramount. These metrics provide a quantitative basis for comparing accelerator designs and are crucial for identifying areas of improvement.

1. Power consumption and latency

Power consumption in an accelerator comprises both dynamic and static components. Dynamic power refers to the power consumed by the accelerator during active computation, whereas static power is the power consumed at idle state. The inclusion of more hardware resources directly escalates both dynamic and static power consumption.

Latency measures the time required for an AI workload to complete its execution on the accelerator for a given input. It is a critical factor in determining the speed at which the accelerator can process data, affecting its real-time performance and user experience.

2. Throughput (TOPS)

As shown in Eq. (5), throughput reflects the rate at which operations are executed per second in the accelerator. This metric is particularly relevant for AI accelerators, where an "operation" implies a multiply-and-accumulate computation

$$\text{TOPS} = \frac{\text{Total number of operations (in Tera)}}{\text{Latency}}. \tag{5}$$

3. Energy efficiency (TOPS/W)

Energy efficiency, expressed as TOPS-per-watt (TOPS/W), gauges the number of operations an accelerator can perform per watt of power consumed. This metric is instrumental in assessing the

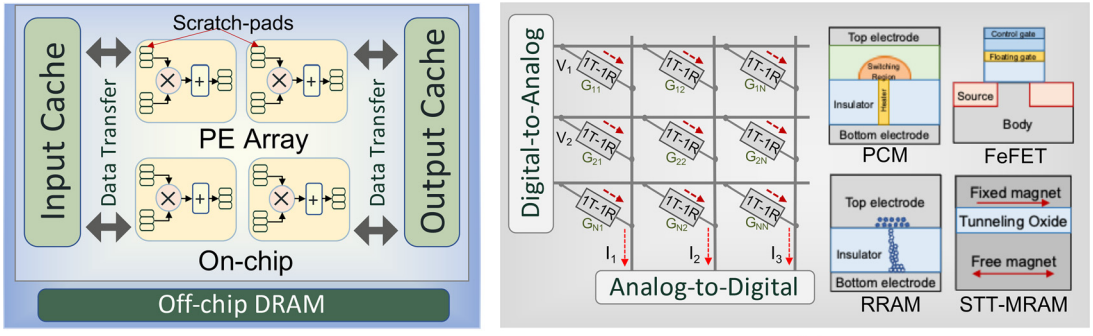

FIG. 4. Figure showing (a) von-Neumann accelerators containing on-chip cache, scratch pad memories, multipliers, and accumulators for performing MAC operations. (b) IMC architectures containing 2D arrays of 1 transistor–1 memristor (1T–1R) devices. They perform fast analog dot-products minimizing data transfer to mitigate the memory wall bottleneck typical in von-Neumann architectures. Over the years, different nonvolatile memory (NVM) devices like phase change memory (PCM), ferroelectric field-effect transistor (FeFET), resistive random-access memory (RRAM), and spin torque transfer-based magnetic RAM (STT–MRAM) have been used as memristors.

sustainability and cost-effectiveness of an accelerator. It is computed as follows:

$$\text{TOPS/W} = \frac{\text{Total number of operations (in Tera)}}{\text{Latency} \times \text{Power}}. \quad (6)$$

It underscores the importance of optimizing both the computational throughput and power efficiency to enhance the overall performance of hardware accelerators.

4. Area efficiency (TOPS/mm^2)

Area efficiency, measured in TOPS-per-square-millimeter (TOPS/mm^2), evaluates the computational density of an accelerator, showcasing how effectively it utilizes its physical space to perform operations. It is computed as follows:

$$\text{TOPS/mm}^2 = \frac{\text{Total number of operations (in Tera)}}{\text{Latency} \times \text{Accelerator Area}}. \quad (7)$$

It underscores the accelerator's ability to maximize its computational output relative to its size, indicating the efficiency of hardware design in terms of area utilization.

C. Synergies between IMC Accelerators and SNNs

1. Energy efficiency of IMC accelerators

Figure 5(a) exhaustively compares the different AI acceleration platforms used today. While GPUs and CPUs offer extensive backend support (such as CUDA[92] for Nvidia GPUs) for AI acceleration, their hardware architecture is fixed and not suitable for extremely low-power applications (<1 W). To this end, systolic accelerators such as Eyeriss[79,90] [denoted as Eyeriss ANN in Fig. 5(a)] have used ANN-centric dataflow and architecture modifications in order to achieve low-power and energy-efficient acceleration. With IMC architecture (denoted as Neurosim ANN), the energy efficiency is further improved. This is mainly attributed to the reduced weight data movement across memories and the analog dot-product operations.

2. SNNs on IMC can further energy efficiency

Evidently, due to the sparse and binary spike computations, SNNs can further improve the energy efficiency and reduce power consumption in both systolic (SATA SNN[31]) and IMC accelerators (SpikeSim SNN[40]) compared to ANNs. To properly benchmark the improvements of SNNs, over ANNs, in this section, we have used SATA[31] and SpikeSim[40] for SNN implementation as they closely represent the ANN-implementation architectures of Eyeriss[79,90] and Neurosim,[91] respectively. However, in the case of SNNs implemented on systolic accelerators like SATA,[31] the SNN is still memory bottlenecked. This leads to a 13× TOPS/W improvement at 6.5× lower power compared to Eyeriss[79,90] ANN. In contrast, the SNNs implemented on IMC architectures are not memory bound. Instead, IMC architectures typically suffer from the peripheral overhead of ADCs and communication circuits. Interestingly, as seen in Fig. 5(b), SNNs possess high activation sparsity compared to ANNs. To this end, the highly sparse binary computation in SNNs can be heavily leveraged to attain extremely low ADC precision, which in turn reduces the communication overhead. Thus, SpikeSim SNN[40] (with algorithmic optimizations like MINT[39] and DT–SNN[41] explained in Sec. IV C) yields 26× higher energy efficiency at 4× lower power compared to Neurosim ANN. The peripheral overhead reduction in SNN yields several synergistic benefits including 4× lower area [Fig. 5(c)], 1.6× higher TOPS [Fig. 5(d)], and 6.67× higher TOPS/mm^2 [Fig. 5(e)] compared to Neurosim ANN.[91]

IV. SYSTEM-LEVEL ANALYSES OF IMC–SNN

A. IMC hardware evaluation platform

Often, large-scale deep SNNs mandate the need for multiple crossbars integrated with numerous digital peripheral modules. For accurate system-level analyses, IMC-realistic evaluation platforms are necessary. To this end, this section highlights the extensive research performed toward large-scale ANN implementations on IMC architectures. Thereafter, it highlights the key modifications that some of the recent SNN–IMC evaluation platforms[40] entail to incorporate SNNs.

1. ANN–IMC evaluation platforms

In an early instance of ANN-based IMC deployment, ISAAC[93] introduced a pipelined accelerator featuring on-chip embedded DRAM (eDRAM) for inter-stage data storage. The researchers conducted extensive design-space exploration to determine an optimal configuration of memristor, ADCs, and eDRAM resources. In another study, MNSIM[94] proposed a unified framework integrating analog and digital IMC platforms for ANN implementations. Additionally, Neurosim-v1[91] presented an end-to-end evaluation framework spanning device, circuit, and system levels for ANNs. The study validated simulation-based findings through post-tapeout testing, demonstrating minimal discrepancies between real and simulated outcomes.[95] More recent endeavors, like SIAM,[96] introduced IMC-based chiplet architectures tailored for ANN execution. Despite the proliferation of IMC platforms for ANNs, they lack the key SNN-specific modules such as the LIF neuron module and temporal dataflow, which are crucial for accommodating SNNs.

2. SNN–IMC evaluation platforms

Over the previous years, there have been several IMC-based SNN platforms. Liu *et al.*[97] proposed an analog crossbar approach for implementing feedforward and Hopfield networks, devoid of convolutional network-based dataflow. Narayanan *et al.*[98] and Zhao *et al.*[99] have constructed small feedforward SNNs trained using STDP algorithms. Bohnsting *et al.*[100] and ReSPARC,[43] along with Kulkarni *et al.*,[101] have undertaken large-scale SNN deployments employing analog synapses and neurons, though these implementations lack open-source availability. To this end, SpikeSim[40] proposes the first open-source end-to-end IMC hardware evaluation platform for benchmarking large-scale SNNs. This review will extensively utilize SpikeSim to perform end-to-end system-level analyses of IMC-implemented SNNs. It should be noted that, although the results are based on SpikeSim, the analyses are applicable to implementing an SNN on any IMC architecture.

a. SpikeSim—An SNN–IMC evaluation platform. Similar to prior ANN works, SpikeSim[40] contains a tiled hierarchical architecture containing tiles, processing elements (PEs) and crossbars as shown in Fig. 6(a). The tiles are connected by a network-on-chip (NoC)

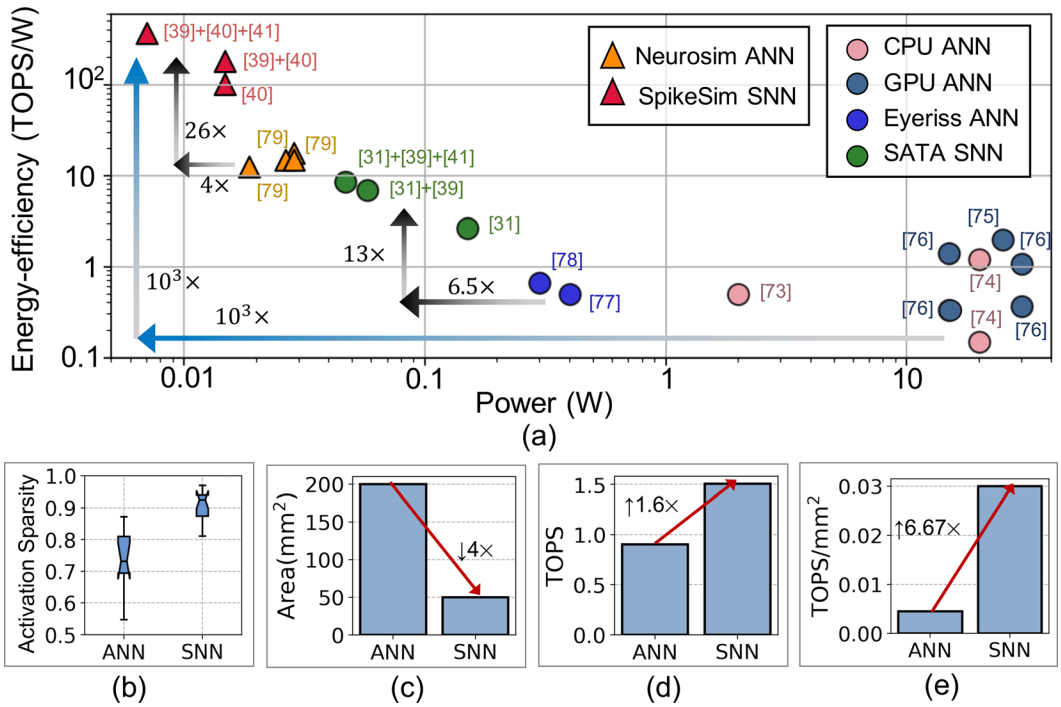

FIG. 5. (a) Plot of energy efficiency (measured in TOPS/W) vs power for different low-power edge AI accelerators. ANN workloads are deployed on CPU (Intel Movidius,[86] Kalray[87]), GPU (Nvidia Jetson Orin Nano,[88] Nvidia Xavier[89]), systolic accelerators [Eyeriss-V1,[79] Eyeriss-V2 (Ref. 90)], and IMC (Neurosim[91]) accelerators. SNNs are deployed on SATA[31] systolic array and SpikeSim[40] IMC accelerator platforms. SATA[31] and SpikeSim[40] are SNN-specific accelerators that closely resemble the Eyeriss[79] and Neurosim[91] platforms, respectively, and thus facilitate a fair comparison. "+" denotes the conjunction of two approaches. Arrows are used to show the reduction and improvements in power efficiency and energy efficiency, respectively. Higher energy efficiency at lower power signifies a good AI accelerator platform. (b) Plot comparing the activation sparsity averaged across different layers of the ANN and SNN. Plots comparing (c) IMC chip area (d) TOPS, and (e) TOPS/mm^2 of ANN and SNN implemented on the Neurosim[91] and SpikeSim[40] platforms, respectively. For all implementations, we use 8-bit VGG16 ANN and SNN (with four timesteps) trained on the CIFAR10 dataset. The SpikeSim[40] and Neurosim[91]-based hardware parameters are shown in Tables III and IV in Appendix, respectively.

interconnect, while the PEs and crossbars are connected by H-Tree interconnects. The crossbars, PE, and Tiles work in tandem to compute the MAC output at a particular time step. Accumulators at each hierarchy add the partial sums to deliver the final MAC output.

SpikeSim entails several architectural modifications for implementing SNNs. First, the authors implement a digital neuron module that facilitates the LIF activation function. The LIF module [see Fig. 6(b)] is implemented at the global hierarchy and contains a U_{mem} cache memory to store the membrane potential values over multiple timesteps. At each time step, the membrane potential value is read from the U_{mem} cache, added to the MAC output of the current time step and written back. Second, the authors leverage the binary spike nature of SNNs to replace the dual-crossbar approach with a cost-efficient digital DIFF module to carry out signed-MAC operations. Finally, the authors employ an SNN-specific layer-scheduled dataflow that improves the throughput and hardware utilization of IMC-implemented SNNs compared to the tick-batched dataflow[42] used in SNN-specific systolic array accelerators.

During inference, the SNN model weights are partitioned onto multiple crossbars and the layer-scheduled dataflow is applied. Simultaneously, SpikeSim further optimizes the floor-planning, NoC, and the neuron module overhead. Following this, SpikeSim employs two engines [shown in Fig. 6(c)]- the non-ideality computation engine (NICE) and the energy-latency-area (ELA) engine, to compute hardware-realistic accuracy and energy-latency-area metrics, respectively.

B. Need for system-level analyses of IMC-SNN

1. Device innovations have been system agnostic

Over the years, comprehensive research efforts have introduced numerous synaptic NVM devices showcasing plasticity akin to neurons in the brain.[102–105] These devices aim to enable low-power unsupervised learning methods such as STDP on memristive crossbars. However, given the current scale of learning tasks, BPTT-based SNN training algorithms have become increasingly pervasive as shown in Sec. II B. Interestingly, BPTT-trained SNNs do not require plasticity-aware synaptic devices. In fact, today's device research is geared toward achieving multi-level synaptic devices with a greater number of stable conductance states,[85] higher On/Off ratios,[106] and lower read voltages to avoid write disturbances and lower read energy during inference.[106–108] Concurrent studies are focused on investigating neuromimetic properties in emerging NVM devices for emulating analog spiking neurons.[19,109–113] New neuron models have also been proposed for improving SNN convergence and hardware implementations.[11,52]

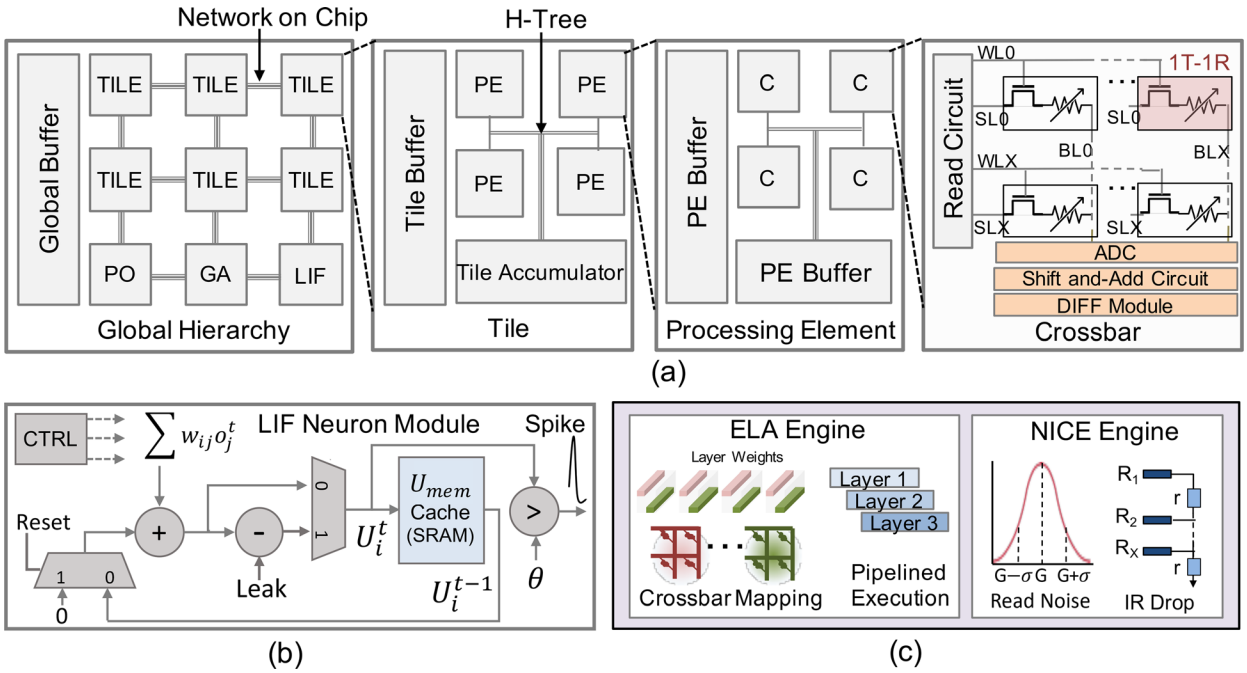

FIG. 6. (a) Figure showing the hierarchical architecture of SpikeSim[40] containing tiles, processing elements (PEs), and crossbars. The tiles and crossbars/PEs are connected by the network-on-chip (NoC) and H-Tree interconnect, respectively. The DIFF module replaces the dual-crossbar approach to perform signed-MAC operations using a single crossbar. (b) The LIF neuron module stores the temporal membrane potentials in the U_{mem} cache and facilitates the leaky-integrate-and-fire activation. (c) The energy-latency-area (ELA) engine and non-ideality computation engine (NICE) perform hardware-realistic energy, latency, area, and accuracy evaluation during SNN inference, respectively. For SNN–IMC hardware evaluation, the SpikeSim parameters are shown in Table III in Appendix.

This research aims to enable seamless integration with analog crossbar-based dot-product engines, contributing to the development of low-power neuromorphic systems. FeFET devices having tunable hysteretic behavior and low-power switching capabilities, have shown promise in emulating the firing patterns observed in biological neurons.[19,114] Device researchers are also exploring the use of NVM devices like FeFETs as NVM memcapacitors for neuromorphic computing.[115–117] Memcapacitive crossbars, unlike the memristive ones, perform analog dot-products in the charge-domain with low dynamic power at negligible static power dissipation. Also, the immunity of memcapacitive crossbars to sneak path currents eliminates the need for access transistors, thereby reducing design complexity and crossbar area.[118] Thus far, research at the device level has proceeded in isolation, devoid of consideration for broader system-level impact, resulting in a discernible gap in the efficient deployment of SNNs on IMCs.

2. Co-dependence among device and system parameters

Table II shows the system-level parameters that are codependent on the different NVM device parameters. Increasing the device precision (number of stable conductance states) reduces the crossbar count required for IMC mapping. However, it also increases the ADC precision to process larger crossbar currents and, in turn, the bandwidth requirements of the buffers and the communication circuits are high (refer Fig. 11 for details). Similarly, increasing the device On resistances and the On/Off ratios increases read parallelism (the number of crossbar rows read in parallel). However, it slows down the system's frequency of operation [refer Fig. 10(d) for details]. Lower operation frequency will also lower the demand for communication and buffer bandwidths. Finally, increasing parameters such as the read voltage (considering no write disturbances) of the NVM devices increases the system's frequency of operation, and hence the requirement for higher interconnect and buffer bandwidths. Therefore, it is imperative to grasp the co-dependencies among device, circuit, and system-level parameters to fully analyze the energy-latency-area-accuracy landscape of SNNs implemented on IMC architectures.

C. SNN system-level bottlenecks and mitigation strategies

The intrinsic energy efficiency of SNNs may be compromised without a thorough understanding of hardware bottlenecks. This section delineates the primary obstacles hindering the efficient integration of SNNs on IMC architectures.

1. The LIF neuron module

Figures 7(a)–7(c) show the area distribution across different modules of the SpikeSim platform for an 8-bit VGG16 SNN trained on CIFAR10[53] (image dimensions of 32×32), Caltech-101[119] (image dimensions of 48×48) and TinyImagenet[120] (image dimensions of 64×64) datasets, respectively. Evidently, due to the large U_{mem} [refer Fig. 6(b)] cache memory, the LIF neuronal module contributes 11.7%–

TABLE II. Table showing the codependency between different device and system parameters. Non-codependent device and system parameters are denoted by ×. Dev., Xbar, Acc., Comm. BW refer to device, crossbar, accumulator, communication bandwidth, respectively.

System / Device	ADC precision	Acc precision	Comm. BW	Xbar Count	Buffer BW
Dev. precision ↑	↑	↑	↑	↓	↑
On resistance ↑	×	×	↓	×	↓
On/Off ratio ↑	×	×	↓	×	↓
Read voltage ↑	×	×	↑	×	↑

25% toward the overall chip area. The LIF module, thus, poses a bottleneck when implementing SNNs trained on large datasets like ImageNet[25] with image dimensions (224×224 and 384×384) on IMC architectures. Interestingly, the LIF module requires $>1000\times$ higher on-chip area compared to ReLU module in ANNs that merely requires a comparator.

a. Co-design-based mitigation strategies. In SpikeSim,[40] the authors propose a simple approach of channel scaling, wherein the number of output channels in the first layer of the network are scaled down yielding $2\times$ reduction in LIF module area. In MINT,[39] the authors apply sophisticated weight and membrane potential quantization-aware SNN training to reduce the LIF overhead. MINT is able to reduce the weight and membrane potential precision as low as 2 bits while maintaining iso-accuracy with SNN trained on FP32 precision. Additionally, another recent work[121] performed sharing of LIF membrane potentials over multiple SNN layers to reduce the LIF memory overhead. The authors perform inter-layer and intra-layer membrane potential sharing to achieve over $4\times$ reduction in the LIF memory area at iso-accuracy.

b. Device research for LIF area mitigation. Device researchers are exploring novel neuromimetic devices emulating biological neuronal functionalities. Recent works have leveraged the fast and low-power switching dynamics of a FeFET-based relaxation oscillator configuration to generate biological spiking patterns at area-efficient form factors.[19,114] In another work, Mohanan *et al.*[122] used nanoporous graphene-based memristive devices to compactly emulate LIF neurons in current SNN workloads showing threshold control, leaky integration, and reset behaviors. The spiking activity of the LIF neuron is tunable by varying various circuit and device parameters, allowing it to cover a broad frequency spectrum.[122] Likewise, Zhou *et al.*[123] have proposed a compact RRAM-based LIF neuron circuit closely integrated with analog RRAM crossbars. This provided a unified path to carry out dot-products and LIF activation functionalities in the analog domain. However, given the large number of spatial channels required by large-scale BPTT-trained SNN models, directly integrating the analog LIF neurons with the crossbars remains an unsolved problem.

2. Temporal computation in SNNs

As SNNs process data over multiple timesteps, an increase in timesteps linearly escalates the energy-delay product (EDP) across MAC, communication, and LIF activation operations [Fig. 8(a)]. Interestingly, the crossbar compute arrays, digital peripherals, and the communication circuits get activated multiple times in a particular time step in order to compute the weighted summation output. In contrast, the LIF activation is performed once every time step. Therefore, the MAC and communication operations significantly contribute to the EDP (80% of the overall EDP). Consequently, reducing the number of timesteps can significantly enhance efficiency. However, Fig. 8(b) illustrates a trade-off between timesteps, energy efficiency, and accuracy. While reducing timesteps improves energy efficiency, it also leads to lower SNN accuracy. Therefore, the development of effective algorithms exploiting spatiotemporal complexity is imperative. Note that while Fig. 8 uses SpikeSim for evaluation, the time step is an intrinsic parameter of the SNN algorithm. Consequently, the linear increase in EDP with timesteps will remain consistent regardless of the IMC platform.

a. Co-design-based timestep minimization strategies. Over the years, training algorithms such as BNTT,[9] along with neuromorphic data augmentation techniques[124] and encoding methods such as direct encoding,[125] have effectively exploited the spatiotemporal complexity of SNNs resulting in a drastic reduction (of the order $10\times$) in the number of timesteps. A more recent approach by Li *et al.*[41] called DT–SNN leverages the difficulty of the input images to scale the number of timesteps in the SNN. During training, the authors use a joint training loss to train an SNN on different count of timesteps. During inference, the authors use an entropy metric to determine the confidence of prediction per time step. An image with lower entropy is deemed easy and inferred at an early time step and difficult images with higher entropy are inferred at latter timesteps. The authors of DT–SNN achieve an overall 81% EDP reduction, with iso or higher accuracy than a standard SNN using fixed number of timesteps for inference across all inputs.

3. Vulnerability toward IMC non-idealities

The practical implementation of NVM devices is constrained by finite conductance levels, limited On/Off resistances, and inherent non-idealities that can adversely affect the inference accuracy of AI workloads.[40,91,126,127] Depending upon the origin, these non-idealities can be classified into device and circuit non-idealities as shown in Fig. 9.

a. Device non-idealities. Stochastic read noise predominantly originates from random telegraphic noise, flicker noise ($1/f$ noise), and thermal noise in the NVM synapses.[126–131] Read noise is modeled as a Gaussian distribution around the programmed conductance during each read cycle, with a standard deviation (σ) increasing with the NVM conductance.[128,130,131] Structural relaxation within NVM devices

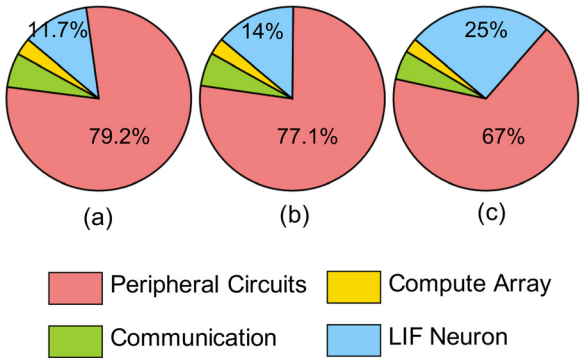

FIG. 7. Area distribution of 8-bit VGG16 SNN on SpikeSim (with hardware parameters shown in Table III in Appendix) with (a) CIFAR10 (b) Caltech-101, and (c) TinyImagenet datasets. Peripheral circuits include ADCs, accumulators, buffers, read circuit, shift-and-add circuit, and DIFF module. Communication involves NoC and H-Tree circuits. Compute array consists of all the 2D 1T–1R crossbar arrays excluding the peripheral circuits. These trends will remain consistent irrespective of the SNN–IMC platform used, as they are determined by the memory cell area and the dataset feature size.

over time leads to another non-ideality called temporal drift, influencing the retention of programmed conductance in crossbars.[130,132] A popular model describing the temporal conductance drift is given as $G(t) = G_0 * (t/t_0)^{-\nu}$, where G_0 represents the initially programmed conductance at time t_0, and ν denotes the drift coefficient. Stuck-at-fault is another non-ideality arising from fabrication defects or extensive crossbar utilization. Stuck-at-faults manifest into fixated NVM synapses in crossbars (to set or reset states), rendering them non-programmable.[133,134] Despite the inherent robustness of NVM crossbars

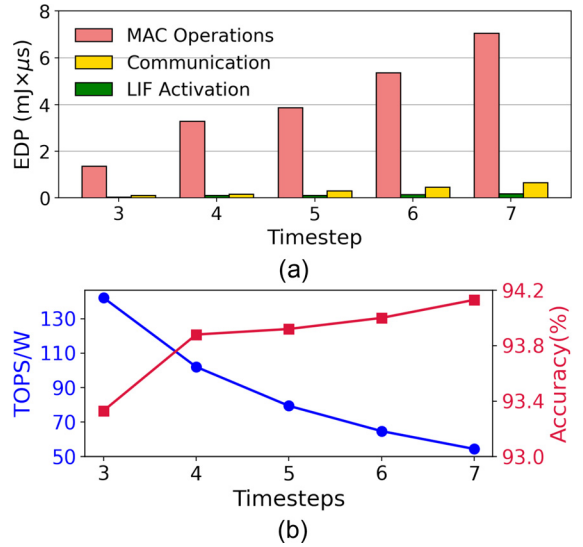

FIG. 8. Figure showing the (a) trends in energy-delay product (EDP) across the MAC operations, LIF activations, and communication circuits. (b) Trends of TOPS/W and accuracy with increasing timesteps. Results correspond to 8-bit VGG16 SNN implemented with SpikeSim trained on the CIFAR10 dataset. SpikeSim parameters for SNN implementation are shown in Table III in Appendix.

to variations, stuck-at-faults can significantly degrade the performance of ANN/SNN workloads.

b. Circuit-level non-idealities. It includes the parasitic resistances in the crossbar metal lines denoted as r_{par}. During analog dot-product operations, the interconnect parasitics lead to stray IR drops, causing the output currents to deviate from their ideal values and resulting in substantial accuracy losses.[135–137] Balancing crossbar size to improve parallelism during inference, while restraining the impact of resistive non-idealities, becomes a delicate trade-off. Furthermore, the presence of access transistors in series with NVM devices in 1T–1R synapses is crucial for eliminating sneak paths and incorrect programming of the NVM devices.[80,81] Nonetheless, it introduces 1T–1R non-linearities arising from the non-linear I–V characteristics of the access transistors.[36]

c. Effect of IMC non-ideality on SNN. For an 8-bit VGG16 SNN model trained with the CIFAR10 dataset, the impact of parasitic resistances and stochastic read noise on the hardware inference accuracy is shown in Fig. 10(a). At higher timesteps (timesteps ≥ 4), we find the non-ideal inference accuracy declines dramatically, owing to significant non-ideality error accumulation over multiple timesteps of computations.[137]

d. Non-ideality-aware training of SNNs. Ensuring robustness to crossbar non-idealities involves iterative offline training of SNN models with noise injection using hardware-realistic noise models.[138–142] Recently, the AIHWKit toolkit from IBM offers statistical empirical models for emulating device-level and parasitic resistive non-idealities for PCM crossbars in PyTorch.[143,144] AIHWKit-based noise-aware training has demonstrated state-of-the-art hardware accuracies across diverse tasks, spanning computer vision to natural language processing. However, non-ideality-aware training is not scalable to today's large-scale SNN models. This is due to the bottleneck of noise injection, especially for the input voltage-dependent resistive parasitic non-idealities, which escalates training latency. Additionally, SNNs incur high training costs due to computations across multiple time steps. This has been depicted in Fig. 10(b), where an iteration of non-ideality-aware training increases the training latency by greater than an order of magnitude compared to standard training. This underscores the need for training-less methods for non-ideality mitigation.

e. Training-less non-ideality mitigation strategies. Recent methods have proposed transformations on NVM conductances during mapping onto crossbars, increasing the proportion of low conductance synapses to mitigate crossbar non-idealities.[36,37,145] Based on this principle, the NICE engine in the SpikeSim framework shows significantly improved SNN inference accuracies on non-ideal crossbars with no additional hardware costs.[40] In addition, a recent work has shown that simple noise-aware adaptation of the batch-normalization (BN) parameters of a BPTT-trained SNN can fully recover the inference accuracy lost due to the non-idealities.[137] This is corroborated in Fig. 10(c) across crossbar sizes of 32×32 and 64×64. Noise-aware BN adaptation is a fully weight-static approach, implying that NVM synapses need not be re-programmed or reconfigured during inference to mitigate non-idealities.[146] Noise-aware BN adaptation incurs nearly

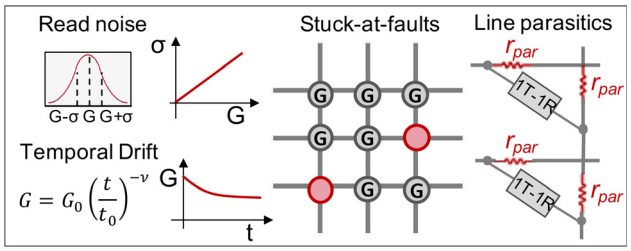

FIG. 9. Non-idealities impacting inference on NVM crossbars. The memristive device-level non-idealities are read noise,[128–131] temporal conductance drift,[130,132] and stuck-at-faults.[133,134] The parasitic resistances[135,136] of the metal lines in the crossbars (r_{par}) and the transistor non-linearities in the 1T–1R synapses[36] comprise the circuit-level non-idealities.

an order of magnitude lower latency than one epoch of standard SNN training [see Fig. 10(b)].

f. Choice of NVM device for non-ideality mitigation. While RRAMs and PCMs are extensively studied for multi-level crossbar synapses, their susceptibility to read noise is a concern. In contrast, FeFET-based synapses, with increased CMOS-compatibility and high On/Off ratios (>100), show promise in reducing read noise.[106] PCMs have exhibited high retention capabilities (>10 years) for temporal drift,[130,147,148] while FeFETs show poorer retention ($\sim 10^3$ to 10^4 s) due to polarization degradation from charge traps, defects, and oxide breakdown.[149] To minimize stray IR drops, synapses with high On resistance (typically > 100 kΩ) are favored.[139]

However, excessively high On resistances diminish the crossbar currents, impacting the readout by sense amplifiers or ADCs.[107] This is corroborated in Fig. 10(d), where increasing On resistance of the synaptic devices leads to higher accuracy for the VGG16 SNN on 64 × 64 crossbars, while reducing the TOPS at the system level. The reduction in TOPS manifests from the reduced crossbar currents driving the ADCs. Furthermore, with research on memcapacitive NVM devices gaining momentum,[115–117] it is noteworthy that the crossbars operating in the charge-domain eliminate the impact of circuit-level non-idealities such as stray IR drops and 1T–1R non-linearities.[115]

V. DISCUSSION AND FUTURE DIRECTIONS

A. Does IMC need very high device precisions?

The device community has always focused on targeting higher number of stable conductance values in NVM devices without considering broader system-level implications. One might naturally assume that enhancing the precision of NVM devices will reduce the number of crossbars (and their associated peripherals) needed to implement SNN layers. However, at higher device precisions, the ADC precision needs to increase (and hence, the ADC area and energy) to avoid quantization errors in the accumulated column currents, resulting in an expanded area and energy at the system level. The co-dependence of device precision with ADC precision is illustrated in Fig. 11(a). For the 8-bit VGG16 SNN model, the optimal NVM device and ADC precisions are found to be 4 and 5 bits, respectively. This yields the best energy and area expenditures at the system level as shown in Fig. 11(b). Therefore, to attain considerable energy and area efficiency, large device precision is not paramount. It must be noted that these trends are IMC platform agnostic as they are solely governed by the device precision.

B. FeFETs as a promising device for U_{mem} cache

In light of the LIF area overhead discussed in Sec. IV C 1, utilizing an NVM device like FeFET for constructing U_{mem} cache could drastically reduce the LIF area by upto 7× compared to the traditional SRAM cache [see Fig. 12(a)]. However, as illustrated in Fig. 12(b), current FeFET technology necessitates multiple write cycles for programming (refer Sec. V C for details on writing into NVM devices), leading to 3× greater write energy than that of SRAM caches. This increased write energy stems from the need to perform U_{mem} write operations over multiple timesteps during SNN inference. Despite FeFETs showing superior noise resilience compared to RRAMs and PCMs, FeFETs continue to display read and write variabilities, potentially decreasing the SNN accuracy by 3%–4% [Fig. 12(c)] across a range of datasets. Note, the relatively short retention time of FeFETs ($\sim 10^3$ to 10^4 s) is unlikely to pose concerns given that the LIF cache is updated at a significantly higher frequency, ranging from tens to hundreds of MHz.

C. Opportunities for IMC-SNNs in online learning

In the recent years, there has been a growing interest in online learning on edge devices.[150] Data privacy concerns make learning on

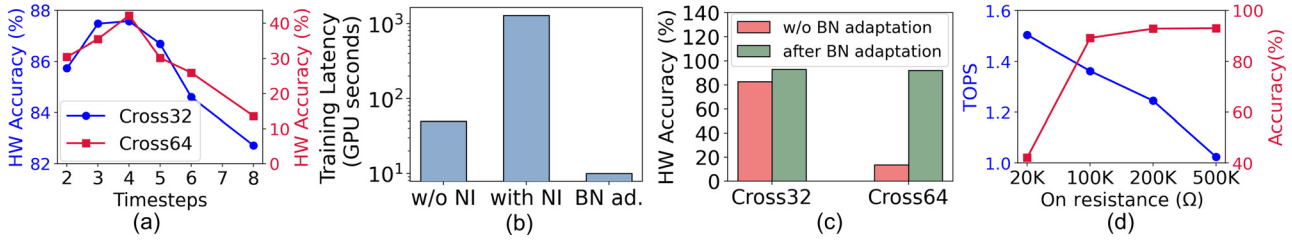

FIG. 10. (a) Plot showing the impact of crossbar non-idealities on SNN inference accuracy. We consider the effect of stray IR drops due to resistive non-idealities ($r_{par} = 5\Omega$) as well as stochastic Gaussian read noise ($\sigma = 0.1$) in the RRAM devices. (b) Plot showing the GPU training latency per iteration for non-ideality-aware training (with NI) compared against standard GPU training (without non-ideality or w/o NI) and training-less noise-aware BN adaptation (BN ad.). Evaluations are performed on the Nvidia RTX2080Ti GPU. (c) Plot showing the effectiveness of noise-aware BN adaptation in mitigating crossbar non-idealities. (d) Plot showing the trend of system-level TOPS and non-ideal accuracy by varying On resistance of the synaptic RRAM devices. For (a), (c), and (d), evaluations are performed on an 8-bit VGG16 SNN implemented on SpikeSim using parameters shown in Table III in Appendix. The accuracy is affected by the device-level parameters and therefore will show similar trends irrespective of the SNN–IMC evaluation platform used. Cross32 and Cross64 denote crossbars of sizes 32 × 32 and 64 × 64, respectively. All evaluations are performed on the CIFAR10 dataset.

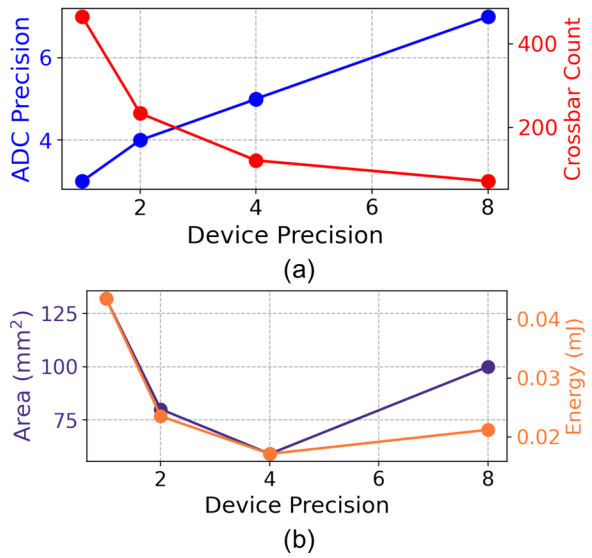

FIG. 11. Figure showing the (a) trend of required ADC precision and crossbar count with increasing device precision. (b) The trend in area and energy upon increasing device precision. All evaluations are performed on SpikeSim with 8-bit VGG16 SNN implemented using hardware parameters shown in Table III in Appendix.

edge devices imperative because it allows sensitive or personal information to be processed directly on-device, without sending it across the internet or to centralized servers. Furthermore, for applications requiring real-time or near-real-time responses, such as autonomous vehicles or emergency response systems, online learning can save huge communication latency and bandwidth as data do not need to be communicated back and forth across the internet.[150,151] The contemporary SNNs demonstrate ability to make accurate predictions with minimal temporal samples. They are particularly suited for edge devices due to their ability to operate at low power and their efficiency in handling highly sparse time-series data, which is common in real-world sensory inputs.[123,152] As discussed in Sec. III C, the strong synergies between SNNs and IMC crossbars, particularly the reduced communication and ADC overheads to process binary and sparse spike data, show potential for employing SNNs on IMC crossbars for online learning.

1. Device challenges toward SNN online learning

In online training, writing into NVM devices involves selecting specific synapses by applying pulses across rows (select lines or SLs) and columns (bit lines or BLs) [see Fig. 13(a)], followed by modulating voltage or current to adjust the synaptic conductance.[91] Each write cycle can degrade the NVM device's material, affecting its lifespan, making high-endurance devices preferable. Additionally, as programming each device requires multiple pulses, write operations are delay and energy-intensive.[153] This is demonstrated in Figs. 13(a) and 13(b). Write challenges also stem from the stochastic write noise and asymmetric conductance updates in the NVM devices [see Fig. 13(c)], which, although negligible during inference, significantly impact weight re-programming during online learning.[91,139,153] These non-idealities necessitate repeated write operations to achieve the desired conductance level, affecting the energy, speed, and device endurance.

2. Hardware requisites for online learning

Write noise mitigation strategies, in general, include error correction codes,[154] write verification-and-retry mechanisms,[155] and structural advancements in the NVM devices.[107,108] FeFETs are less susceptible to write noise compared to RRAMs and PCMs, owing to deterministic polarization-switching at low voltages. However, FeFETs, in general, show limited endurance ($\sim 10^4$–10^{10} cycles) owing to mobility degradation and charge trapping phenomena.[156–160] The low endurance of FeFETs can become problematic during online learning as the weights of the SNN need to be updated frequently. Furthermore, RRAMs and PCMs rely on filament formation mechanism and high thermal energy for state change, respectively, leading to high write energy and latency. In contrast, FeFETs offer substantial write energy and latency reductions due to the rapid and low-voltage switching of ferroelectric layers.

3. Hardware-friendly online learning paradigms

Current BPTT-based algorithms entail huge memory and computational costs for facilitating backpropagation on hardware over multiple timesteps.[31] The ability to update model parameters locally and independently at each layer is important for online and continual learning paradigms. This is crucial for applications that require the model to adapt continuously to new data without the need for re-

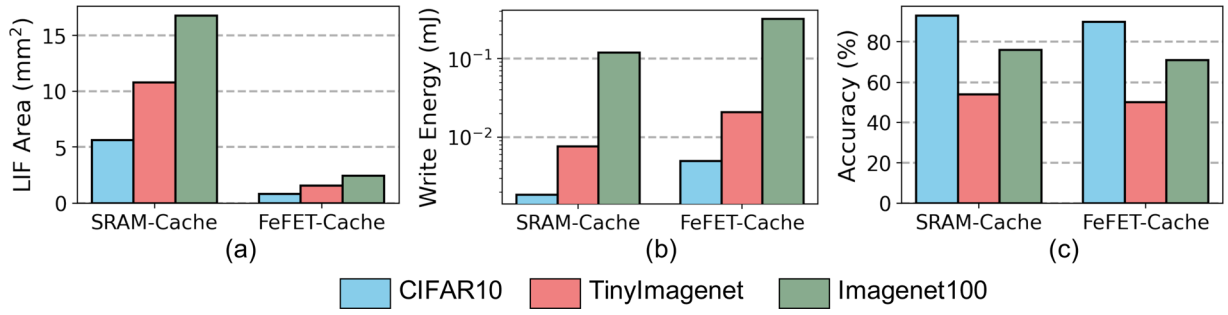

FIG. 12. Figure comparing the (a) LIF module area, (b) Write Energy, (c) Accuracy of SpikeSim-implemented SNNs implemented with SRAM and FeFET-based U_{mem} cache inside the LIF module. Results are shown across the CIFAR10, TinyImagenet, and Imagenet100 datasets for an 8-bit VGG16 SpikeSim-implemented SNN with parameters shown in Table III in Appendix. These trends will remain consistent irrespective of the SNN–IMC platform used, as they are determined by the memory cell area and the dataset feature size.

training from scratch. To this end, local gradient-based learning methods, such as direct feedback alignment (DFA),[161] show great promise in reducing training latency and improving TOPS/W at the system level over traditional backpropagation.[162–164] Additionally, emerging learning algorithms exploiting the eligibility traces in SNNs can achieve bioplausible[165] and memory-efficient online learning at the edge.[166]

D. Need for layer-specific peripheral circuit co-optimization

So far, all optimizations that have taken place have been implemented homogeneously across different SNN layers, regardless of the layer-specific computational complexity. However, it is important to note that different layers have different compute complexity and therefore will require specific device-circuit-system and algorithmic parameter optimization. Recent works have proposed layer-specific device[167] and peripheral circuit parameters[168] to obtain optimal energy and area efficiencies. To optimize the communication overhead, work by Krishnan *et al.*[169] has proposed layer-specific tile sizes to minimize inter-tile communications. Here, it is important to highlight that despite the two-dimensional integration using NoCs in a typical silicon fabrication process, the on-chip connectivity still falls short of the three-dimensional connectivity observed in the brain.[170] Consequently, more recently, 3D crossbar-based IMC architectures[171,172] have emerged as a viable solution to address this communication bottleneck. Nevertheless, all these studies have primarily focused on optimizations within the ANN domain, underscoring the importance of conducting layer-specific optimizations tailored for SNNs.

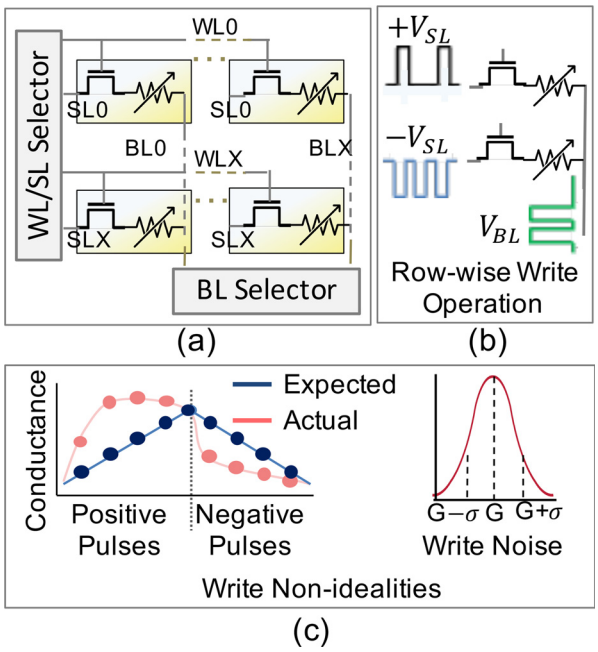

FIG. 13. Figure showing (a) the essential circuits required to facilitate weight updates in IMC architectures, (b) costly NVM write operation over multiple write pulses, and[91,153] (c) the write non-idealities[91,139] such as asymmetric weight update and write noise.

VI. CONCLUSION

The review delineates the pivotal synergies between SNNs and IMC architectures, showcasing their efficacy in ultra-low-power edge computing scenarios. SNNs are being actively used for various commercial applications requiring extensive academic studies across multiple application spaces. To achieve optimal low-power edge implementations, the review motivates system-level analyses by considering the co-dependencies between algorithm, device, circuit, and system parameters. Furthermore, we point out the bottlenecks at the system level that arise from implementing SNNs on IMC architectures due to NVM device limitations. To this end, our review delves into several device, circuit, and system-aware co-design-based strategies that have been developed to overcome the inherent bottlenecks. Finally, we emphasize on future device research landscape to facilitate energy-efficient IMC–SNN deployment with key focus on online learning, emerging neuronal devices, and effective design-space co-exploration.

ACKNOWLEDGMENTS

This work was supported in part by CoCoSys, a JUMP2.0 center sponsored by DARPA and SRC, the National Science Foundation (CAREER Award, Grant Nos. 2312366 and 2318152), and the DoE MMICC center SEA-CROGS (Award No. DE-SC0023198).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Abhishek Moitra and Abhiroop Bhattacharjee have contributed equally to this paper.

Abhishek Moitra: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Abhiroop Bhattacharjee:** Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Yuhang Li:** Conceptualization (supporting); Data curation (supporting); Formal analysis (supporting); Investigation (supporting); Methodology (supporting); Validation (supporting); Visualization (supporting); Writing – original draft (supporting). **Youngeun Kim:** Formal analysis (supporting); Validation (supporting); Visualization (supporting); Writing – review & editing (supporting). **Priyadarshini Panda:** Conceptualization (equal); Funding acquisition (lead); Investigation (supporting); Project administration (lead); Resources (lead); Software (lead); Supervision (lead); Validation (supporting); Writing – review & editing (supporting).

DATA AVAILABILITY

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

TABLE III. Table with values of various circuit and device parameters used for SpikeSim[40] evaluation unless otherwise mentioned.

SpikeSim evaluation parameters	
Technology	65 nm CMOS
NoC topology	Mesh
NoC width	32 bits
Crossbar size	64 × 64
Clock frequency	250 MHz
Crossbars/PE	9
PE/Tile	8
Buffer sizes	Global-20 KB, Tile-10 KB, PE-5 KB
Read voltage	0.1 V
Device	RRAM
Device precision	4 bits
On/Off ratio	10 (R_{on} = 20 k Ω)

TABLE IV. Table with values of various circuit and device parameters used for Neurosim[91] evaluation unless otherwise mentioned.

Neurosim evaluation parameters	
Technology	65 nm CMOS
Crossbar size	64 × 64
Clock frequency	250 MHz
Crossbars/PE	9
PE/Tile	8
Buffer sizes	Global-20 KB, Tile-10 KB, PE-5 KB
Read voltage	0.1 V
Device	RRAM
Device precision	4 bits
On/Off ratio	10 (R_{on} = 20 k Ω)

APPENDIX: HARDWARE EVALUATION PARAMETERS

We have added Tables III & IV with values of circuit and device parameters used for SpikeSim & NeuroSim based evaluation, respectively, in this work.

REFERENCES

[1]A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2012), Vol. 25.

[2]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.

[3]A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16 × 16 words: Transformers for image recognition at scale," arXiv:2010.11929 (2020).

[4]J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805 (2018).

[5]A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "AI and ML accelerator survey and trends," in *IEEE High Performance Extreme Computing Conference (HPEC)* (IEEE, 2022), pp. 1–10.

[6]V. Kandiah, S. Peverelle, M. Khairy, J. Pan, A. Manjunath, T. G. Rogers, T. M. Aamodt, and N. Hardavellas, "AccelWattch: A power modeling framework for modern GPUs," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture* (ACM, 2021), pp. 738–753.

[7]A. Mehonic and A. J. Kenyon, "Brain-inspired computing needs a master plan," Nature **604**, 255–260 (2022).

[8]K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," Nature **575**, 607–617 (2019).

[9]Y. Kim and P. Panda, "Revisiting batch normalization for training low-latency deep spiking neural networks from scratch," Front. Neurosci. **15**, 773954 (2021).

[10]S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: Spiking neural network for energy-efficient object detection," in *Proceedings of the* AAAI *Conference Artificial Intelligence* (AAAI, 2020), Vol. 34, pp. 11270–11277.

[11]A. Shaaban, M. Strobel, W. Furtner, R. Weigel, and F. Lurz, "RT-SCNNs: Real-time spiking convolutional neural networks for a novel hand gesture recognition using time-domain mm-wave radar data," Int. J. Microwave Wireless Technol. **2024**, 1–13.

[12]J. MacLean, B. Stewart, and I. Gyongy, "TDC-less direct time-of-flight imaging using spiking neural networks," arXiv:2401.10793 (2024).

[13]Y. Zhou, J. Fu, Z. Chen, F. Zhuge, Y. Wang, J. Yan, S. Ma, L. Xu, H. Yuan, M. Chan *et al.*, "Computational event-driven vision sensors for in-sensor spiking neural networks," Nat. Electron. **6**, 870–878 (2023).

[14]F. Barchi, L. Zanatta, E. Parisi, A. Burrello, D. Brunelli, A. Bartolini, and A. Acquaviva, "Spiking neural network-based near-sensor computing for damage detection in structural health monitoring," Future Internet **13**, 219 (2021).

[15]Y. Li, R. Yin, Y. Kim, and P. Panda, "Efficient human activity recognition with spatio-temporal spiking neural networks," Front. Neurosci. **17**, 1233037 (2023a).

[16]S. Bian and M. Magno, "Evaluating spiking neural network on neuromorphic platform for human activity recognition," in *Proceedings of the 2023 ACM International Symposium on Wearable Computers* (ACM, 2023), pp. 82–86.

[17]S. Tanzarella, M. Iacono, E. Donati, D. Farina, and C. Bartolozzi, "Neuromorphic decoding of spinal motor neuron behaviour during natural hand movements for a new generation of wearable neural interfaces," IEEE Trans. Neural Syst. Rehabil. Eng. **31**, 3035–3046 (2023).

[18]P. Gong, P. Wang, Y. Zhou, and D. Zhang, "A spiking neural network with adaptive graph convolution and LSTM for EEG-based brain–computer interfaces," IEEE Trans. Neural Syst. Rehabil. Eng. **31**, 1440–1450 (2023).

[19]Y. Fang, J. Gomez, Z. Wang, S. Datta, A. I. Khan, and A. Raychowdhury, "Neuro-mimetic dynamics of a ferroelectric FET-based spiking neuron," IEEE Electron Device Lett. **40**, 1213–1216 (2019).

[20]Y. Qi, J. Chen, and Y. Wang, "Neuromorphic computing facilitates deep brain–machine fusion for high-performance neuroprosthesis," Front. Neurosci. **17**, 1153985 (2023).

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

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

[23]See https://www.andante-ai.eu/ for "Andante."

[24]See https://tsst.demcon.com/technology/collaboration/ulpec/ for "TSST."

[25]J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2009), pp. 248–255.

[26]Y. Li, T. Geller, Y. Kim, and P. Panda, "SEENN: Towards temporal spiking early exit neural networks," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2024), Vol. 36.

[27]N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture* (ACM, 2017), pp. 1–12.

[28]R. Xu, F. Han, and Q. Ta, "Deep learning at scale on NVIDIA v100 accelerators," in *IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (IEEE, 2018), pp. 23–32.
[29]N. Verma, H. Jia, H. Valavi, Y. Tang, M. Ozatay, L.-Y. Chen, B. Zhang, and P. Deaville, "In-memory computing: Advances and prospects," IEEE Solid-State Circuits Mag. **11**, 43–55 (2019).
[30]A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," Nat. Nanotechnol. **15**, 529–544 (2020).
[31]R. Yin, A. Moitra, A. Bhattacharjee, Y. Kim, and P. Panda, "SATA: Sparsity-aware training accelerator for spiking neural networks," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **42**, 1926–1938 (2022).
[32]N. R. Shanbhag and S. K. Roy, "Comprehending in-memory computing trends via proper benchmarking," in *IEEE Custom Integrated Circuits Conference (CICC)* (IEEE, 2022), pp. 01–07.
[33]Y. Kim, Y. Li, H. Park, Y. Venkatesha, R. Yin, and P. Panda, "Exploring lottery ticket hypothesis in spiking neural networks," in *European Conference on Computer Vision* (Springer, 2022), pp. 102–120.
[34]Y. Kim, Y. Li, H. Park, Y. Venkatesha, and P. Panda, "Neural architecture search for spiking neural networks," in *European Conference on Computer Vision* (Springer, 2022), pp. 36–56.
[35]M. Rao, H. Tang, J. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang *et al.*, "Thousands of conductance levels in memristors integrated on CMOS," Nature **615**, 823–829 (2023).
[36]A. Bhattacharjee, L. Bhatnagar, Y. Kim, and P. Panda, "NEAT: Nonlinearity aware training for accurate, energy-efficient, and robust implementation of neural networks on 1T–1R crossbars," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **41**, 2625–2637 (2021).
[37]A. Bhattacharjee and P. Panda, "*SwitchX*: Gmin–Gmax switching for energy-efficient and robust implementation of binarized neural networks on ReRAM Xbars," ACM Trans. Des. Autom. Electron. Syst. **28**, 60 (2023).
[38]A. Moitra and P. Panda, "DetectX – Adversarial input detection using current signatures in memristive Xbar arrays," IEEE Trans. Circuits Syst. I **68**, 4482–4494 (2021).
[39]R. Yin, Y. Li, A. Moitra, and P. Panda, "MINT: Multiplier-less INTeger quantization for spiking neural networks," in *29th Asia and South Pacific Design Automation Conference (ASP-DAC)* (IEEE, 2024).
[40]A. Moitra, A. Bhattacharjee, R. Kuang, G. Krishnan, Y. Cao, and P. Panda, "SpikeSim: An end-to-end compute-in-memory hardware evaluation tool for benchmarking spiking neural networks," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **42**, 3815–3828 (2023a).
[41]Y. Li, A. Moitra, T. Geller, and P. Panda, "Input-aware dynamic timestep spiking neural networks for efficient in-memory computing," in *60th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2023), pp. 1–6.
[42]S. Narayanan, K. Taht, R. Balasubramonian, E. Giacomin, and P.-E. Gaillardon, "SpinalFlow: An architecture and dataflow tailored for spiking neural networks," in *ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (IEEE, 2020), pp. 349–362.
[43]A. Ankit, A. Sengupta, P. Panda, and K. Roy, "RESPARC: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017* (ACM, 2017), pp. 1–6.
[44]N. Caporale and Y. Dan, "Spike timing–dependent plasticity: A Hebbian learning rule," Annu. Rev. Neurosci. **31**, 25–46 (2008).
[45]Y. Munakata and J. Pfaffly, "Hebbian learning and development," Dev. Sci. **7**, 141–148 (2004).
[46]T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," PLoS Comput. Biol. **3**, e31 (2007).
[47]C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity," IEEE Trans. Cognitive Dev. Syst. **11**, 384–394 (2018a).
[48]C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with STDP-based unsupervised pre-training followed by supervised fine-tuning," Front. Neurosci. **12**, 435 (2018b).
[49]Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," Int. J. Comput. Vision **113**, 54–66 (2015).
[50]P. U. Diehl, D. Neil, J. Binas, M. Cook, and S. C. Liu, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2015).
[51]B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding," in European Conference on Computer Vision (Springer, 2020), pp. 388–404.
[52]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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 13558–13567.
[53]A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (Canadian Institute for Advanced Research)," 2010.
[54]E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," IEEE Signal Process. Mag. **36**, 61–63 (2019).
[55]Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," Front. Neurosci. **12**, 331 (2018).
[56]Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI, 2019), Vol. 33, pp. 1311–1318.
[57]J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," Front. Neurosci. **10**, 508 (2016).
[58]A. Shaban, S. S. Bezugam, and M. Suri, "An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation," Nat. Commun. **12**, 4234 (2021).
[59]S. M. Bohte, "Error-backpropagation in networks of fractionally predictive spiking neurons," in Artificial Neural Networks and Machine Learning–ICANN 2011, edited by T. Honkela, W. Duch, M. Girolami, and S. Kaski (Springer Berlin Heidelberg, Berlin, Heidelberg, 2011), pp. 60–68.
[60]N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," arXiv:2005.01807 (2020).
[61]Q. Su, Y. Chou, Y. Hu, J. Li, S. Mei, Z. Zhang, and G. Li, "Deep directly-trained spiking neural networks for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE, 2023), pp. 6555–6565.
[62]J. López-Randulfe, T. Duswald, Z. Bing, and A. Knoll, "Spiking neural network for Fourier transform and object detection for automotive radar," Front. Neurorob. **15**, 688344 (2021).
[63]N. Salvatore and J. Fletcher, "Dynamic vision-based satellite detection: A time-based encoding approach with spiking neural networks," in *International Conference on Computer Vision Systems* (Springer, 2023), pp. 285–298.
[64]F. Yang, L. Su, J. Zhao, X. Chen, X. Wang, N. Jiang, and Q. Hu, "SA-FlowNet: Event-based self-attention optical flow estimation with spiking-analogue neural networks," IET Comput. Vision **17**, 925–935 (2023).
[65]Y. Zheng, Z. Yu, S. Wang, and T. Huang, "Spike-based motion estimation for object tracking through bio-inspired unsupervised learning," IEEE Trans. Image Process. **32**, 335–349 (2022).
[66]A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017), pp. 7243–7252.
[67]J.-M. Maro, S.-H. Ieng, and R. Benosman, "Event-based gesture recognition with dynamic background suppression using smartphone computational capabilities," Front. Neurosci. **14**, 275 (2020).
[68]A. Vasudevan, P. Negri, B. Linares-Barranco, and T. Serrano-Gotarredona, "Introduction and analysis of an event-based sign language dataset," in *15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)* (IEEE, 2020), pp. 675–682.
[69]M. Zhai, K. Ni, J. Xie, and H. Gao, "Spike-based optical flow estimation via contrastive learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2023), pp. 1–5.
[70]S. Chen, Z. Yu, and T. Huang, "Self-supervised joint dynamic scene reconstruction and optical flow estimation for spiking camera," in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI, 2023), Vol. 37, pp. 350–358.

[71]L. Xia, Z. Ding, R. Zhao, J. Zhang, L. Ma, Z. Yu, T. Huang, and R. Xiong, Unsupervised optical flow estimation with dynamic timing representation for spike camera," arXiv:2307.06003 (2023).
[72]J. Zhao, S. Zhang, Z. Yu, and T. Huang, "SpiReco: Fast and efficient recognition of high-speed moving objects with spike cameras," IEEE Trans. Circuits Syst. Video Technol. **34**, 5856–5867 (2024).
[73]T. Dalgaty, F. Moro, Y. Demirağ, A. De Pra, G. Indiveri, E. Vianello, and M. Payvand, "Mosaic: In-memory computing and routing for small-world spike-based neuromorphic systems," Nat. Commun. **15**, 142 (2024).
[74]F. Tian, J. Yang, S. Zhao, and M. Sawan, "NeuroCARE: A generic neuromorphic edge computing framework for healthcare applications," Front. Neurosci. **17**, 1093865 (2023).
[75]F. Wang, T. Zhang, C. Dou, Y. Shi, and L. Pan, "Neuromorphic devices, circuits, and their applications in flexible electronics," IEEE J. Flexible Electron. **3**, 42–56 (2023a).
[76]See https://grayscale.ai/technology "Grayscale AI," 2024.
[77]See https://www.orbai.com/ "Orbai," 2023.
[78]L. Feng, H. Shan, Z. Fan, Y. Zhang, L. Yang, and Z. Zhu, "Towards neuromorphic brain–computer interfaces: Model and circuit co-design of the spiking EEGNet," Microelectron. J. **137**, 105808 (2023).
[79]Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits **52**, 127–138 (2017).
[80]J. Yoon, Y. Ji, S.-K. Lee, J. Hyon, and J. M. Tour, "Low-temperature-processed SiO_x one diode–one resistor crossbar array and its flexible memory application," Adv. Electron. Mater. **4**, 1700665 (2018).
[81]X. Feng, S. Li, S. L. Wong, S. Tong, L. Chen, P. Zhang, L. Wang, X. Fong, D. Chi, and K.-W. Ang, "Self-selective multi-terminal memtransistor crossbar array for in-memory computing," ACS Nano **15**, 1764–1774 (2021).
[82]G. W. Burr, M. J. Brightsky, A. Sebastian, H.-Y. Cheng, J.-Y. Wu, S. Kim, N. E. Sosa, N. Papandreou, H.-L. Lung, H. Pozidis *et al.*, "Recent progress in phase-change memory technology," IEEE J. Emerging Sel. Top. Circuits Syst. **6**, 146–162 (2016).
[83]S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, and H.-S. P. Wong, "An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation," IEEE Trans. Electron Devices **58**, 2729–2737 (2011).
[84]X. Wang, Y. Chen, H. Xi, H. Li, and D. Dimitrov, "Spintronic memristor through spin-torque-induced magnetization motion," IEEE Electron Device Lett. **30**, 294–297 (2009).
[85]M. Jerry, P.-Y. Chen, J. Zhang, P. Sharma, K. Ni, S. Yu, and S. Datta, "Ferroelectric FET analog synapse for acceleration of deep neural network training," in *IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2017), pp. 6.2.1–6.2.4.
[86]See https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html "Intel Movidius," 2023.
[87]See https://www.kalrayinc.com/products/dpu-processors/ "Kalray dpu," 2024.
[88]See https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/ "NVIDIA Orin series," 2024a.
[89]See https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/ "NVIDIA Xavier series," 2024b.
[90]Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," IEEE J. Emerging Sel. Top. Circuits Syst. **9**, 292–308 (2019).
[91]P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **37**, 3067–3080 (2018).
[92]D. Luebke, "CUDA: Scalable parallel programming for high-performance scientific computing," in *5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro* (IEEE, 2008), pp. 836–838.
[93]A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ACM SIGARCH Comput. Archit. News **44**, 14–26 (2016).
[94]L. Xia, B. Li, T. Tang, P. Gu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "MNSIM: Simulation platform for memristor-based neuromorphic computing system," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **37**, 1009–1022 (2018).
[95]A. Lu, X. Peng, W. Li, H. Jiang, and S. Yu, "NeuroSim validation with 40 nm RRAM compute-in-memory macro," in *IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (IEEE, 2021), pp. 1–4.
[96]G. Krishnan, S. K. Mandal, M. Pannala, C. Chakrabarti, J.-S. Seo, U. Y. Ogras, and Y. Cao, "SIAM: Chiplet-based scalable in-memory acceleration with mesh for deep neural networks," ACM Trans. Embedded Comput. Syst. **20**, 68 (2021).
[97]C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Y. Chen, H. Li, Q. Wu, and H. Jiang, "A spiking neuromorphic design with resistive crossbar," in *Proceedings of the 52nd Annual Design Automation Conference* (ACM, 2015), pp. 1–6.
[98]S. Narayanan, A. Shafiee, and R. Balasubramonian, "Inxs: Bridging the throughput and energy gap for spiking neural networks," in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2017), pp. 2451–2459.
[99]C. Zhao, K. Hamedani, J. Li, and Y. Yi, "Analog spike-timing-dependent resistive crossbar design for brain inspired computing," IEEE J. Emerging Sel. Top. Circuits Syst. **8**, 38–50 (2018).
[100]T. Bohnstingl, A. Pantazi, and E. Eleftheriou, "Accelerating spiking neural networks using memristive crossbar arrays," in *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)* (IEEE, 2020), pp. 1–4.
[101]S. R. Kulkarni, S. Yin, J.-s. Seo, and B. Rajendran, "An on-chip learning accelerator for spiking neural networks using STT–RAM crossbar arrays," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE, 2020), pp. 1019–1024.
[102]Y. Van De Burgt, E. Lubberman, E. J. Fuller, S. T. Keene, G. C. Faria, S. Agarwal, M. J. Marinella, A. Alec Talin, and A. Salleo, "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing," Nat. Mater. **16**, 414–418 (2017).
[103]M. Suri, O. Bichler, D. Querlioz, O. Cueto, L. Perniola, V. Sousa, D. Vuillaume, C. Gamrat, and B. DeSalvo, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *International Electron Devices Meeting* (IEEE, 2011), pp. 4.4.1–4.4.4.
[104]S. La Barbera, D. Vuillaume, and F. Alibart, "Filamentary switching: Synaptic plasticity through device volatility," ACS Nano **9**, 941–949 (2015).
[105]I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," Nat. Commun. **9**, 2514 (2018).
[106]S. Zhang, Y. Liu, J. Zhou, M. Ma, A. Gao, B. Zheng, L. Li, X. Su, G. Han, J. Zhang *et al.*, "Low voltage operating 2D MoS_2 ferroelectric memory transistor with $Hf_{1-x}Zr_xO_2$ gate structure," Nanoscale Res. Lett. **15**(1), 1–9 (2020).
[107]N. Jao, Y. Xiao, A. K. Saha, S. K. Gupta, and V. Narayanan, "Design space exploration of ferroelectric tunnel junction toward crossbar memories," IEEE J. Explor. Solid-State Comput. Devices Circuits **7**, 115–122 (2021).
[108]Y. Raffel, S. De, M. Lederer, R. R. Olivo, R. Hoffmann, S. Thunder, L. Pirro, S. Beyer, T. Chohan, T. Kämpfe *et al.*, "Synergistic approach of interfacial layer engineering and READ-voltage optimization in HfO_2-based FeFETs for in-memory-computing applications," ACS Appl. Electron. Mater. **4**, 5292–5300 (2022).
[109]P. Stoliar, J. Tranchant, B. Corraze, E. Janod, M.-P. Besland, F. Tesler, M. Rozenberg, and L. Cario, "A leaky-integrate-and-fire neuron analog realized with a mott insulator," Adv. Funct. Mater. **27**, 1604740 (2017).
[110]X. Zhang, W. Wang, Q. Liu, X. Zhao, J. Wei, R. Cao, Z. Yao, X. Zhu, F. Zhang, H. Lv *et al.*, "An artificial neuron based on a threshold switching memristor," IEEE Electron Device Lett. **39**, 308–311 (2018).
[111]Z. Wang, M. Yin, T. Zhang, Y. Cai, Y. Wang, Y. Yang, and R. Huang, "Engineering incremental resistive switching in TaO_x based memristors for brain-inspired computing," Nanoscale **8**, 14015–14022 (2016).
[112]T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Stochastic phase-change neurons," Nat. Nanotechnol. **11**, 693–699 (2016).
[113]R. Schroedter, A. S. Demirkol, A. Ascoli, B. Max, F. Nebe, T. Mikolajick, and R. Tetzlaff, "A pseudo-memcapacitive neurotransistor for spiking neural networks," in *12th International Conference on Modern Circuits and Systems Technologies (MOCAST)* (IEEE, 2023), pp. 1–5.
[114]Z. Wang and A. I. Khan, "Ferroelectric relaxation oscillators and spiking neurons," IEEE J. Explor. Solid-State Comput. Devices Circuits **5**, 151–157 (2019).
[115]T.-H. Kim, O. Phadke, Y.-C. Luo, H. Mulaosmanovic, J. Mueller, S. Duenkel, S. Beyer, A. I. Khan, S. Datta, and S. Yu, "Tunable non-volatile gate-to-source/

drain capacitance of FeFET for capacitive synapse," IEEE Electron Device Lett. **44**, 1628–1631 (2023a).
[116]S. Hwang, J. Yu, G. H. Lee, M. S. Song, J. Chang, K. K. Min, T. Jang, J.-H. Lee, B.-G. Park, and H. Kim, "Capacitor-based synaptic devices for hardware spiking neural networks," IEEE Electron Device Lett. **43**, 549–552 (2022).
[117]W. E. Engeler, "Capacitive structures for weighted summation as used in neural nets," US Patent 5,039,871, 1991.
[118]W. Chen, "Selector-free cross-point memory architecture based on ferroelectric MFM capacitors," in *IEEE 11th International Memory Workshop (IMW)* (IEEE, 2019), pp. 1–2.
[119]F.-F. Li, M. Andreeto, M. Ranzato, and P. Perona, "Caltech 101," CaltechDATA, 2022.
[120]Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," CS 231N **7**, 3 (2015).
[121]Y. Kim, Y. Li, A. Moitra, R. Yin, and P. Panda, "Sharing leaky-integrate-and-fire neurons for memory-efficient spiking neural networks," arXiv:2305.18360 (2023b).
[122]K. U. Mohanan, S. M. Sattari-Esfahlan, E.-S. Cho, and C.-H. Kim, "Optimization of leaky integrate-and-fire neuron circuits based on nanoporous graphene memristors," IEEE J. Electron Devices Soc. **12**, 88–95 (2024).
[123]P. Zhou, D.-U. Choi, W. D. Lu, S.-M. Kang, and J. K. Eshraghian, "Gradient-based neuromorphic learning on dynamical RRAM arrays," IEEE J. Emerging Sel. Top. Circuits Syst. **12**, 888–897 (2022).
[124]Y. Li, Y. Kim, H. Park, T. Geller, and P. Panda, "Neuromorphic data augmentation for training spiking neural networks," in *European Conference on Computer Vision* (Springer, 2022), pp. 631–649.
[125]Y. Kim, H. Park, A. Moitra, A. Bhattacharjee, Y. Venkatesha, and P. Panda, "Rate coding or direct coding: Which one is better for accurate, robust, and energy-efficient spiking neural networks?," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2022), pp. 71–75.
[126]P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in *IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2017), pp. 6.1.1–6.1.4.
[127]W. Wan, R. Kubendran, S. B. Eryilmaz, W. Zhang, Y. Liao, D. Wu, S. Deiss, B. Gao, P. Raina, S. Joshi *et al.*, "33.1 A 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models," in *IEEE International Solid-State Circuits Conference-(ISSCC)* (IEEE, 2020), pp. 498–500.
[128]S. Agarwal, S. J. Plimpton, D. R. Hughart, A. H. Hsia, I. Richter, J. A. Cox, C. D. James, and M. J. Marinella, "Resistive memory device requirements for a neural algorithm accelerator," in *2016 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2016), pp. 929–938.
[129]D. Veksler, G. Bersuker, L. Vandelli, A. Padovani, L. Larcher, A. Muraviev, B. Chakrabarti, E. Vogel, D. Gilmer, and P. Kirsch, "Random telegraph noise (RTN) in scaled RRAM devices," in *2013 IEEE International Reliability Physics Symposium (IRPS)* (IEEE, 2013), pp. MY.10.1–MY.10.4.
[130]S. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "A phase-change memory model for neuromorphic computing," J. Appl. Phys. **124**, 152135 (2018).
[131]X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," IEEE J. Emerging Sel. Top. Circuits Syst. **9**, 570–579 (2019).
[132]P.-Y. Chen and S. Yu, "Reliability perspective of resistive synaptic devices on the neuromorphic system performance," in *IEEE International Reliability Physics Symposium (IRPS)* (IEEE, 2018), pp. 5C.4-1–5C.4-4.
[133]I. Yeo, M. Chu, S.-G. Gi, H. Hwang, and B.-G. Lee, "Stuck-at-fault tolerant schemes for memristor crossbar array-based neural networks," IEEE Trans. Electron Devices **66**, 2937–2945 (2019).
[134]B. Zhang, N. Uysal, D. Fan, and R. Ewetz, "Handling stuck-at-faults in memristor crossbar arrays using matrix transformations," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (ACM, 2019), pp. 438–443.
[135]F. Zhang and M. Hu, "Mitigate parasitic resistance in resistive crossbar-based convolutional neural networks," ACM J. Emerging Technol. Comput. Syst. **16**, 25 (2020).
[136]S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "RxNN: A framework for evaluating deep neural networks on resistive crossbars," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **40**, 326–338 (2020).
[137]A. Bhattacharjee, Y. Kim, A. Moitra, and P. Panda, "Examining the robustness of spiking neural networks on non-ideal memristive crossbars," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design* (ACM, 2022), pp. 1–6.
[138]B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Proceedings of the 52nd Annual Design Automation Conference* (ACM, 2015), pp. 1–6.
[139]S. Roy, S. Sridharan, S. Jain, and A. Raghunathan, "TxSim: Modeling training of deep neural networks on resistive crossbar systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **29**, 730–738 (2021).
[140]I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "GENIEx: A generalized approach to emulating non-ideality in memristive Xbars using neural networks," in *57th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2020), pp. 1–6.
[141]G. Charan, J. Hazra, K. Beckmann, X. Du, G. Krishnan, R. V. Joshi, N. C. Cady, and Y. Cao, "Accurate inference with inaccurate RRAM devices: Statistical data, model transfer, and on-line adaptation," in *57th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2020), pp. 1–6.
[142]M. Dampfhoffer, J. M. Lopez, T. Mesquida, A. Valentian, and L. Anghel, "Improving the robustness of neural networks to noisy multi-level non-volatile memory-based synapses," in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2023), pp. 1–8.
[143]M. J. Rasch, D. Moreda, T. Gokmen, M. Le Gallo, F. Carta, C. Goldberg, K. El Maghraoui, A. Sebastian, and V. Narayanan, "A flexible and fast PyTorch toolkit for simulating training and inference on analog crossbar arrays," in *IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (IEEE, 2021), pp. 1–4.
[144]M. J. Rasch, C. Mackin, M. Le Gallo, A. Chen, A. Fasoli, F. Odermatt, N. Li, S. Nandakumar, P. Narayanan, H. Tsai *et al.*, "Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators," Nat. Commun. **14**, 5282 (2023).
[145]A. Bhattacharjee, A. Moitra, and P. Panda, "ClipFormer: Key-value clipping of transformers on memristive crossbars for write noise mitigation," arXiv:2402.02586 (2024).
[146]A. Bhattacharjee, A. Moitra, Y. Kim, Y. Venkatesha, and P. Panda, "Examining the role and limits of batchnorm optimization to mitigate diverse hardware-noise in in-memory computing," arXiv:2305.18416 (2023).
[147]A. Pirovano, A. Redaelli, F. Pellizzer, F. Ottogalli, M. Tosi, D. Ielmini, A. L. Lacaita, and R. Bez, "Reliability study of phase-change nonvolatile memories," IEEE Trans. Device Mater. Rel. **4**, 422–427 (2004).
[148]M. Le Gallo and A. Sebastian, "Phase-change memory," in *Memristive Devices for Brain-Inspired Computing* (Elsevier, 2020), pp. 63–96.
[149]S. Huang, X. Zhong, Y. Zhang, Q. Tan, J. Wang, and Y. Zhou, "A retention model for ferroelectric-gate field-effect transistor," IEEE Trans. Electron Devices **58**, 3388–3394 (2011).
[150]J. Chen and X. Ran, "Deep learning with edge computing: A review," Proc. IEEE **107**, 1655–1674 (2019).
[151]S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," Proc. IEEE **107**, 1697–1716 (2019).
[152]J. L. Lobo, J. Del Ser, A. Bifet, and N. Kasabov, "Spiking neural networks and online learning: An overview and perspectives," Neural Networks **121**, 88–100 (2020).
[153]M. J. Marinella, S. Agarwal, A. Hsia, I. Richter, R. Jacobs-Gedrim, J. Niroula, S. J. Plimpton, E. Ipek, and C. D. James, "Multiscale co-design analysis of energy, latency, area, and accuracy of a ReRAM analog neural training accelerator," IEEE J. Emerging Sel. Top. Circuits Syst. **8**, 86–101 (2018).
[154]D. Niu, Y. Xiao, and Y. Xie, "Low power memristor-based ReRAM design with error correcting code," in *17th Asia and South Pacific Design Automation Conference* (IEEE, 2012), pp. 79–84.
[155]W. Li, X. Sun, S. Huang, H. Jiang, and S. Yu, "A 40-nm MLC–RRAM compute-in-memory macro with sparsity control, on-chip write-verify, and temperature-independent ADC references," IEEE J. Solid-State Circuits **57**, 2868–2877 (2022c).

[156]J. Duan, H. Xu, S. Zhao, F. Tian, J. Xiang, K. Han, T. Li, X. Wang, W. Wang, and T. Ye, "Impact of mobility degradation on endurance fatigue of FEFET with TiN/$Hf_{0.5}Zr_{0.5}O_2$/SiO_x/Si (MFIS) gate structure," J. Appl. Phys. **131**, 134102 (2022).
[157]M. Pesic, A. Padovani, S. Slcsazeck, T. Mikolajick, and L. Larcher, "Deconvoluting charge trapping and nucleation interplay in FEFETs: Kinetics and reliability," in *IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2018), pp. 25.1.1–25.1.4.
[158]A. J. Tan, Y.-H. Liao, L.-C. Wang, N. Shanker, J.-H. Bae, C. Hu, and S. Salahuddin, "Ferroelectric HfO_2 memory transistors with high-κ interfacial layer and write endurance exceeding 10^{10} cycles," IEEE Electron Device Lett. **42**, 994–997 (2021).
[159]T. Ali, P. Polakowski, S. Riedel, T. Büttner, T. Kämpfe, M. Rudolph, B. Pätzold, K. Seidel, D. Löhr, R. Hoffmann *et al.*, "High endurance ferroelectric hafnium oxide-based FEFET memory without retention penalty," IEEE Trans. Electron Devices **65**, 3769–3774 (2018).
[160]E. Yurchuk, S. Mueller, D. Martin, S. Slesazeck, U. Schroeder, T. Mikolajick, J. Müller, J. Paul, R. Hoffmann, J. Sundqvist *et al.*, "Origin of the endurance degradation in the novel HfO_2-based 1T ferroelectric non-volatile memories," in *IEEE International Reliability Physics Symposium* (IEEE, 2014), pp. 2E–5.
[161]A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2016), Vol. 29.
[162]T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," Nat. Commun. **7**, 13276 (2016).
[163]B. Crafton, M. West, P. Basnet, E. Vogel, and A. Raychowdhury, "Local learning in RRAM neural networks with sparse direct feedback alignment," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)* (IEEE, 2019), pp. 1–6.
[164]Y. Lu, X. Li, L. Yan, T. Zhang, Y. Yang, Z. Song, and R. Huang, "Accelerated local training of cnns by optimized direct feedback alignment based on stochasticity of 4 mb c-doped $Ge_2Sb_2Te_5$ PCM chip in 40 nm node," in *IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2020), pp. 36.3.1–36.3.4.
[165]G. Bellec, F. Scherr, E. Hajek, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Eligibility traces provide a data-inspired alternative to backpropagation through time," in Real Neurons and Hidden Units: Future Directions at the Intersection of Neuroscience and Artificial Intelligence@ NeurIPS 2019, 2019.
[166]C. Frenkel and G. Indiveri, "Reckon: A 28 nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales," in *IEEE International Solid-State Circuits Conference (ISSCC)* (IEEE, 2022), Vol. 65, pp. 1–3.
[167]A. Bhattacharjee, A. Moitra, and P. Panda, "HyDe: A hybrid PCM/FEFET/SRAM device-search for optimizing area and energy-efficiencies in analog IMC platforms," IEEE J. Emerging Sel. Top. Circuits Syst. **13**, 1073–1082 (2023).
[168]A. Moitra, A. Bhattacharjee, Y. Kim, and P. Panda, "XPert: Peripheral circuit & neural architecture co-search for area and energy-efficient Xbar-based computing," in *60th ACM/IEEE Design Automation Conference (DAC)* (IEEE, 2023b), pp. 1–6.
[169]G. Krishnan, S. K. Mandal, C. Chakrabarti, J.-s. Seo, U. Y. Ogras, and Y. Cao, "Interconnect-aware area and energy optimization for in-memory acceleration of DNNs," IEEE Des. Test **37**, 79–87 (2020).
[170]F. P. Ulloa Severino, J. Ban, Q. Song, M. Tang, G. Bianconi, G. Cheng, and V. Torre, "The role of dimensionality in neuronal network dynamics," Sci. Rep. **6**, 29640 (2016).
[171]Z. Wang, J. Sun, A. Goksoy, S. K. Mandal, J.-S. Seo, C. Chakrabarti, U. Y. Ogras, V. Chhabria, and Y. Cao, "Benchmarking heterogeneous integration with 2.5D/3D interconnect modeling," in *IEEE 15th International Conference on ASIC (ASICON)* (IEEE, 2023), pp. 1–4.
[172]G. Murali, X. Sun, S. Yu, and S. K. Lim, "Heterogeneous mixed-signal monolithic 3-D in-memory computing using resistive RAM," IEEE Trans. Very Large Scale Integration (VLSI) Syst. **29**, 386–396 (2021).