Exploring Spike-Based Learning for Neuromorphic Computing: Prospects and Perspectives

Nitin Rathi, Amogh Agrawal, Chankyu Lee, Adarsh Kumar Kosta, Kaushik Roy
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA.
{*rathi2, agrawa64, lee2216, akosta, kaushik*}*@purdue.edu*

***Abstract*—Spiking neural networks (SNNs) operating with sparse binary signals (spikes) implemented on event-driven hardware can potentially be more energy-efficient than traditional artificial neural networks (ANNs). However, SNNs perform computations over time, and the neuron activation function does not have a well-defined derivative leading to unique training challenges. In this paper, we discuss the various spike representations and training mechanisms for deep SNNs. Additionally, we review applications that go beyond classification, like gesture recognition, motion estimation, and sequential learning. The unique features of SNNs, such as high activation sparsity and spike-based computations, can be leveraged in hardware implementations for energy-efficient processing. To that effect, we discuss various SNN implementations, both using digital ASICs as well as analog in-memory computing primitives. Finally, we present an outlook on future applications and open research areas for both SNN algorithms and hardware implementations.**

***Index Terms*—Spiking neural networks, event cameras, spiking backpropagation, liquid state machine, in-memory computing**

I. Introduction

Building machines having brain-like capabilities has been a persistent dream of computer scientists. With recent advancements in artificial intelligence, especially deep artificial neural networks (ANNs), today's computers achieve super-human performance in several cognitive tasks – for example *AlphaGo* beating the *Go* master in 2016. However, the unprecedented success of deep ANNs is accompanied with significant power and energy cost [1]. While the human brain's energy budget is considered ∼20W, including simultaneous recognition, reasoning and control [2], conventional computing systems consume an order of magnitude higher power for classification alone. This remarkable ability of the human brain has lead researchers to seek inspiration from neuroscience for building neuromorphic computing systems [3] that can operate with brain-like efficiency, especially for battery-operated and resource-constrained edge devices. Among other things, the efficiency of biological systems is attributed to highly parallel, sparse, and event-driven computations. Spiking neural networks (SNNs) operating on sparse binary signals ('spikes') in an event-driven manner implemented on neuromorphic hardware can potentially address the energy problem in ANNs. The spike-based computation in SNNs replaces the energy intensive multiply-and-accumulate operation (required in ANNs) with simple additions. With all its appeal for energy efficiency, the success of SNNs was delayed due to the lack of accurate training methods. In recent years, SNNs trained with conversion and gradient-based learning algorithms have achieved similar inference accuracy as ANNs for complex image classification tasks [4]–[6]. Hardware implementation of these SNN algorithms are equally important in order to achieve energy-efficiency. Unlike the organic fabric of the human brain, the silicon computing fabric has very different characteristics and limitations. Thus, simply mimicking the neuronal functionality of the brain might not lead to optimal design. However, taking key inspirations, for example, event-driven hardware to leverage the highly sparse spike-based processing and co-location of memory and compute units to minimize the data-transfers, might lead the way to energy-efficient edge intelligence.

In this review article, we first describe the various aspects of SNNs including the brain-inspired neuro-synaptic models and spike-based representation. Next, we discuss various challenges in SNN training, and the current state-of-the-art methodologies to efficiently train low-latency and highly sparse SNNs. Then we review various applications of SNN algorithms beyond standard classification tasks, such as gesture recognition, sequential learning, etc. Further we briefly discuss the various attempts to build specialized hardware for SNN processing, including both digital neuromorphic systems, as well as analog in-memory computing based systems. Finally, we discuss the algorithmic and hardware outlook highlighting various challenges in SNN implementations which are still open areas of research.

II. Neuron Models and Input Representation

A. Neuron Models

The neuron models employed in SNNs are designed to transmit output signals in the form of binary events (*spikes*) over time. Typically, spiking neuron models are characterized by an internal state (*membrane potential*) and a firing threshold; the neuron generates an output spike when the internal state crosses the threshold value. The two prevalent models: integrate-and-fire (IF), and leaky-integrate-and-fire (LIF) have been widely adopted for various cognitive tasks such as image classification [5], [6] and motion estimation [7]. The iterative IF/LIF model for a neuron can be described by

$$u_i^t = \lambda_i u_i^{t-1} + \sum_j w_{ij} o_j^t - v_i o_i^{t-1} \tag{1}$$

where u is the membrane potential, λ is the leak factor with a value in $(0-1]$ ($\lambda = 1$ represents the IF neuron), w is the weight connecting pre-neuron j and post-neuron i, o is

978-3-9819263-5-4/DATE21/©2021 EDAA

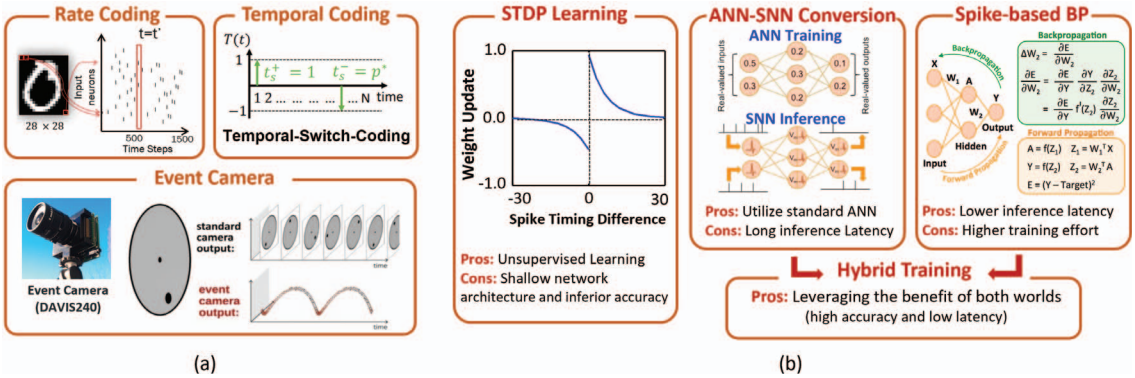

Fig. 1. (a) Spike input representation (rate coding/temporal coding/event camera); and (b) training mechanisms (STDP/ANN-SNN conversion/spike-based BP/hybrid training) for spiking neural networks.

the spike output, v is the firing threshold, and t represents the timestep. The neuron generates a spike ($o = 1$) when the membrane potential (u) crosses the threshold voltage. Additionally, there are conductance based models, such as Hodgkin-Huxley model [8], that define the dynamics based on different ion channels observed in more realistic neuronal mechanisms. However, there is a lack of understanding of training the complex neuron models and, thus the recent advances in SNN learning algorithms have been mostly concentrated on the general and simple IF/LIF neuron models.

B. Spike Input Representation

In general, two types of sensory data can be used as inputs for neuromorphic computing: analog-based and spike-based data. Because SNNs operate and communicate with spike inputs, analog-based data need to be converted to spikes to pass through the network. Accordingly, the spike encoding stage is essentially crucial for determining the overall performance of the SNNs. The analog values need to be encoded with high precision to achieve better accuracy, and at the same time the spikes need to be sparse and encoded in lower number of time-steps for better energy-efficiency. Sometimes, these requirements are at odds and both accuracy and energy can not be improved at the same time [4]–[6]. To that effect, rate coding [5], direct input coding [4], [9], and temporal coding [10]–[12] have been widely used to convert static images into spike signals (Fig. 1(a)). Another popular method, dynamic vision sensors (DVS) [13] or event cameras, directly capture the information of scenes and generates a stream of spikes. Unlike standard frame-based cameras that sample individual pixel intensities, event cameras asynchronously detect intensity changes on each pixel element, generating spatio-temporal event inputs for SNNs without necessitating spike encoding stage.

1) Rate coding: In rate coding, the analog value is represented as the firing rate of the neuron. In Poisson encoding, a type of rate coding, at each timestep the normalised analog value is compared with a random number. The neuron fires (output '1') if the analog value is greater, otherwise stays inactive (output '0'). The number of timesteps[1] determines the discretization error in the representation of the analog value by spike-train. This leads to adopting a large number of timesteps for high accuracy at the expense of high inference latency [5].

2) Direct input coding: The high latency in rate coding can be reduced by introducing an encoding layer with parameters that learns the spike representations with gradient-based training [4]. The encoding layer composed of IF/LIF neurons directly receives the raw analog signals and converts it to spikes based on the trained weights, and the threshold/leak. This encoding scheme achieves the lowest latency (< 10 timesteps) on state-of-the-art networks for image classification tasks [9].

3) Temporal coding: While the rate coding represents the information as the average firing rate over time, the temporal coding encodes the analog inputs in the timing instances of the input spikes. For example, the logarithmic temporal coding [10] encodes the analog value as a binary number with a predefined number of bits. The number of bits serve as a proxy for time and a spike is generated for each active bit in the binary number. For sparser representations, the spike is generated only for the most significant bit. Another example is the rank order coding [11] that represents the information as the order of firing instances instead of using the precise timing of the spike. The input neurons encoding larger analog values fire earlier compared to neurons encoding smaller values. In temporal-switch coding (TSC) [12], the analog value is encoded using two spikes and the time difference between the spikes is proportional to the encoded value. Although temporal coding methods can encode information with less number of spikes, the lack of appropriate training algorithms for temporally coded SNNs results in sub-optimal performance compared to rate-coded SNNs [14].

4) Event Cameras: Event cameras are bio-inspired vision sensors that directly provide a stream of spike events (without spike encoding stage) as an outcome of detecting

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

intensity changes at each pixel element. Event cameras hold great promise for applications requiring low-latency and low-energy inferences owing to their high temporal resolution, high dynamic range and low power consumption compared to standard frame-based sensors. While ANNs are not compatible with the asynchronous and discrete nature of event camera outputs in their native form, SNNs turn out to be capable of directly handling event camera outputs. To that effect, SNNs have emerged as the ideal candidate, offering asynchronous computations and inherently exploiting the sparsity of input events. For SNNs, the main advantage of using event camera is that no complicated spike encoding stage is required, thereby circumventing the input discretization error. However, event camera data involves some hurdles such as "spike vanishing" phenomenon (i.e. the number of spikes rapidly reduce in the deeper layers of SNNs). This hinders learning, leading to performance degradation, especially for extremely sparse event streams of DVS sensors. Such issues have been efficiently addressed by hybridizing SNNs and ANNs in a network where SNNs form the initial layers, enabling effective event stream handling [7]. While the later layers are ANNs, addressing the "spike vanishing" issue and helping to retain performance.

III. Training Mechanisms

In this section, we discuss the various training mechanisms for SNNs as shown in Fig. 1(b).

A. Bio-plausible local learning rules

ANNs trained with gradient-descent employ error-backpropagation from the final layer to the input layer. The weight update for a layer is computed based on all synaptic weights from that layer to the final layer, which is memory intensive. Hence, various local learning rules are proposed that compute the weight update based on local information available at each layer. Spike-timing-dependent-plasticity (STDP), a local Hebbian learning rule, postulates that the strength of the synapse is dependent only on the spike timing difference of the pre- and post-neuron connected to the synapse. A two-layered fully-connected SNN trained with STDP weight updates can perform handwritten digit classification reasonably well [15]. Also, efficient feature extractions have been reported using layer-wise STDP learning for multi-layer convolutional SNNs [16]. The standard backpropagation algorithm has been adapted to a local learning rule by introducing a random matrix to replace the weights of the latter layers [17]. However, the local learning rules are not scalable to deeper networks and other methods are explored to train deep SNNs.

B. ANN to SNN Conversion Networks

The gradient-based methods applied in ANNs compute the derivative of each operation performed in the forward pass. Unfortunately, the IF/LIF neurons employing the spike function in SNNs do not have a continuous derivative. The derivative of the spike function (Dirac delta) is undefined at the time instance of spike and '0' otherwise. To get around the non-differentiability, several works have proposed SNNs converted from trained ANNs for efficient event-driven inference [4], [5], [12]. In this process, an ANN with ReLU neurons is trained with standard backpropagation with some restrictions (no bias term, average pooling, no batch normalization). The inclusion of bias term and batch-normalization makes the threshold selection process more difficult [5]. Although, some works have shown that some of the restrictions can be relaxed [4]. Next, an SNN with IF neurons and iso-architecture as ANN is initialized with the weights of the trained ANN. The underlying principle of this process is that a ReLU neuron can be mapped to an IF neuron with minimum information loss. The major bottleneck of these methods is to determine the proper firing threshold of the IF neurons that can balance the accuracy-latency tradeoff. Generally, the threshold is computed as the maximum pre-activation of the IF neuron resulting in high inference accuracy at the cost of high inference latency ($2000-2500$ timesteps) [5]. The latency can be reduced at the cost of accuracy by choosing a threshold less than the maximum pre-activation. However, the ANN-SNN conversion has a major drawback: the absence of the temporal information. The quintessential parameter 'time' is not utilized in the training and conversion process that leads to higher inference latency. The backpropagation based algorithms, described next, utilize the temporal statistics during the training and have lower inference latency compared to conversion algorithms.

C. Spike-based Backpropagation

There have been several surrogate gradient proposals to address the discontinuous derivative of the spike functions, enabling direct training of deep SNNs using input spikes [18], [19]. The surrogate derivative methods approximately estimate the spike function as a continuous function that serves as the surrogate for the real gradient. SNNs trained with surrogate derivative method perform backpropagation through time (BPTT) to achieve competitive accuracy and low inference latency (≤ 100 timesteps). The authors in [19] demonstrated training VGG and residual SNN architectures consisting of 9-11 layers. However, the training is compute and memory intensive in terms of total training iterations compared to conversion techniques. The multiple-iteration training effort with exploding memory requirement has limited the application of spike-based backpropagation method from building very deep network architecture (beyond 11 layers).

D. Hybrid Training

Recently, a hybrid mechanism was proposed to circumvent the considerable training costs of spike-based backpropagation as well as maintain low inference latency ($100-250$ timesteps) [6]. This method involves both ANN-SNN conversion and spike-based backpropagation. A trained ANN is converted to an SNN as described earlier and the weights of the converted SNN are further fine-tuned with surrogate gradient method and BPTT technique. The authors showed a faster convergence (≤ 20 epochs) in SNN training due to the precursory initialization from the ANN-SNN conversion. The authors in [9]

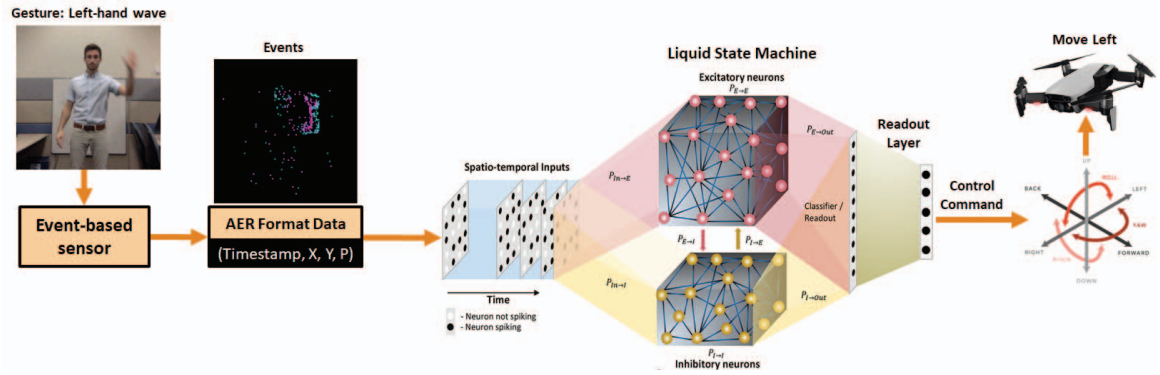

Fig. 2. Event-based gesture recognition and control using a Liquid state machine running on a resource constrained edge device (drone).

further improved the inference latency ($5-10$ timesteps) by employing hybrid training along with direct input coding, and threshold/leak optimization on the large-scale image dataset, ImageNet. The hybrid training presents a practically feasible method to train very deep SNNs with limited computing resources, which is otherwise challenging with only spike-based backpropagation from random initialization [18], [19].

IV. Applications

In this section, we discuss the application of SNNs beyond image classification tasks (motion estimation, speech recognition, reinforcement learning), and recurrent network architectures for SNNs.

A. Motion Estimation

The advent of event-based sensors (as discussed in Section II) have opened up promising opportunities for SNNs owing to their inherent compatibility. While ANN-based methods are susceptible for losing temporal information by accumulating the event camera outputs over a period of time, SNNs are capable of directly handling the asynchronous event streams, exploiting rich spatio-temporal dynamics. To take advantage of this benefit, the authors in Spike-FlowNet [7] process spatio-temporal event-inputs from a DVS camera to predict the optical flow (motion). The approach involved using a hybrid SNN-ANN network architecture trained end-to-end using surrogate gradient based BPTT. This work outperformed state-of-the-art fully ANN based approaches while being more energy-efficient, thereby establishing the importance of SNNs for event-inputs. Research in matching SNN and event-based sensors is still in an early stage, requiring a paradigm shift in network architectures and learning algorithms.

B. Recurrent SNNs

The image classification tasks employ layer-based convolutional and fully-connected SNNs. However, real-world tasks involving spatio-temporal (video) and sequential (speech, handwriting) inputs would require the usage of recurrent layer-based SNNs. These would be highly compute and memory intensive when used with gradient-based learning, limiting implementation on resource constrained edge-devices. Liquid state machines (LSM) have been explored as lightweight alternate architectures to handle spatio-temporal inputs in a bio-plausible manner [20]. LSM, a recurrent SNN, consists of a sparsely connected reservoir (liquid) of excitatory and inhibitory spiking neurons. The synaptic connections and their weights are randomly initialized and fixed a priori. This leads to a simplistic lightweight structure while still inherently capturing the spatio-temporal input information. Spike inputs are passed to the liquid and output class is predicted using a readout layer. Several works have successfully employed LSMs for a variety of applications ranging from gesture recognition [21], video activity recognition [22], reinforcement learning [23], etc, with low compute costs. Fig. 2 shows a LSM based gesture recognition application on the IBM DVS-gesture dataset containing spike inputs from an event-camera. The major challenge with LSM lies in its inability to scale well for real-life complex computing tasks without drastically increasing the liquid size. Various efforts have been made towards improving the learning capability of LSMs without increasing their size significantly. Authors in [21], [24] explore mechanisms for training the liquid connections to improve application accuracy at the cost of added complexity. Other efforts focus on optimizing the network architecture itself. Authors in [22] propose an architecture involving multiple layers of liquids to form a deep hierarchical LSM connected using an attention modulated readout layer. On the other hand, [25] proposes to divide a large liquid into smaller ensembles to process different parts of the input effectively, reducing liquid size while maintaining accuracy.

V. Hardware Framework for SNNs

In this section, we discuss the hardware implementation for running SNNs. Based on the various SNN algorithms described above, a few key features can be identified which present excellent opportunities to be leveraged in hardware for energy-efficient processing. First, SNNs exhibit a very high level of sparsity in spikes (typically $\sim$90% [6]). This can be leveraged by an even-driven hardware, that processes only if the neuron fires and skips the computations for other neurons that do not fire, leading to significant reduction in energy as well as latency. Secondly, a distributed, yet integrated compute and memory fabric, unlike conventional von-Neumann systems, leverages the data parallelism in SNNs

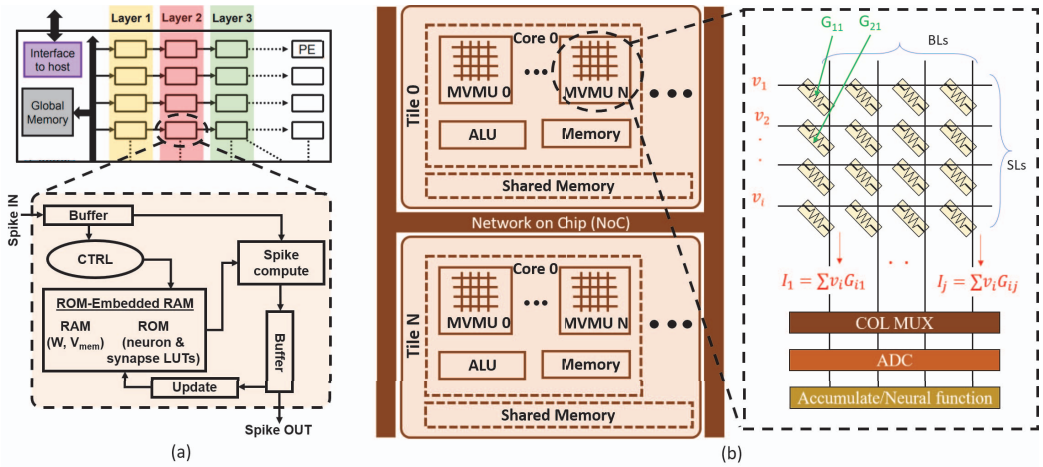

Fig. 3. (a) Event-driven, distributed architecture with LUT-based computing for neuro-synaptic models and learning-rules in SNNs. (b) Analog in-memory computing architecture using embedded non-volatile memory arrays for performing highly parallel matrix-vector-multiplication operation.

and reduces data-transfers between memory and compute units. Thirdly, spike-based information representation simplify the synaptic computations to accumulate operations only, in contrast to multiply-accumulate in traditional ANNs. Lastly, an efficient implementation of complex dynamical functions for various neuro-synaptic models with minimal circuit primitives is required. Thus, different computing paradigms have been explored, with an aim to build energy-efficient and fast accelerators specifically tailored for SNNs, to advance the neuromorphic computing research.

A. Event-driven Hardware

Although standard CPUs and GPUs have been shown to reasonably serve the ANN workloads, the lack of sparsity support makes them extremely inefficient for SNNs, especially with large number of timesteps involved [26]. To explicitly leverage the event-driven nature of spike-based computing, neuromorphic chips such as the IBM's *TrueNorth* [27] and Intel's *Loihi* [26] have been demonstrated. The *TrueNorth* chip consists of a 2-D array of 4096 neurosynaptic cores, consisting of 1 million neurons and 256 million synapses in total. The *Loihi* chip consists of 128 neuromorphic cores, consisting about 130,000 neurons and 130 million synapses, with a large flavor of neuronal features and learning rules. Both designs follow an event-driven asynchronous network-on-chip (NoC) infrastructure to communicate spikes among cores, as and when they are generated. Such a distributed connection of multiple cores capable of working in parallel, along with event-driven NoC makes these architectures inherently different from conventional systems, and more suited to spike-based processing.

B. Lookup-table based Computing

In order to support a range of neuro-synaptic models, such as IF, LIF, Izhikevich, Hodgkin-Huxley etc., *TrueNorth* and *Loihi* rely on digital circuits in each neuromorphic core for solving differential equations with transcendental functions and high-order polynomials. Another alternative is to use lookup tables (LUTs) storing the pre-computed values for such transcendental functions, making the computations simpler. Authors in [28] explored an event-driven distributed processing architecture, *SPARE*, using ROM-embedded SRAMs, which embeds LUTs within the SRAM arrays with negligible overhead, effectively doubling the memory density. The proposed architecture is illustrated in Fig. 3(a), consisting of a 2-D array of Processing Elements (PEs), wherein each PE has its own ROM-embedded RAM to store the weights and neuron state along with LUTs of neuro-synaptic models and learning rules. All computations are executed locally within each PE, and only the spikes (1-bit) are transmitted among PEs, leading to simplified PE structure and interconnect fabric. Moreover, the event-controller within the PEs skips the computations for '0' spikes, leading to benefits arising from spike sparsity.

C. Analog In-memory Computing

While digital systems are accurate, robust, and scalable, analog computing offers higher parallelism and energy-efficiency at the cost of approximations. The error resiliency of SNNs (and even ANNs) allows leveraging such analog-based systems to perform the SNN computations, within the memory arrays. The embedded non-volatile memory arrays can be operated to perform highly-parallel matrix-vector-multiplication (MVM) operations [29]. The concept is illustrated in Fig. 3(b). In the context of SNNs, the matrix is stored in the memory cells representing the synaptic weights of the neural network, while the input voltages fed into the array through the sourcelines represent the input spikes. The resulting synaptic current is obtained on the bitlines through the Kirchoff's laws and is converted to a digital value using an analog-to-digital converter (ADC), for accumulating across multiple crossbar arrays and/or neuronal activation.

VI. Discussion

SNNs introduce unique features like time, membrane potential, threshold, and leak, that if handled properly in both algorithm and hardware, can offer new opportunities. Although temporal coding attempts to utilize time to encode analog input, but further research is needed to explore methods to

employ time in more engaging ways that is not possible in ANNs. Similarly, membrane potential provides an inherent memory that can be explored to include new functionalities. Finally, threshold and leak can precisely control the neuron's output over time that can be exploited to improve the performance. The efficacy of the SNNs largely depends on the underlying hardware architecture. We discussed in Section III that with advancements in SNN training algorithms, the latency of inference has been significantly reduced from thousands of timesteps to a few tens of timesteps while maintaining significant spike sparsity. This clearly highlights their prowess towards utilizing the temporal information in a much more productive manner, potentially achieving real-time and energy-efficient inference. The specialized SNN accelerators discussed in Section V efficiently handle spike-based computing and are event-driven architectures, thereby directly leveraging the low timesteps and highly sparse SNNs. However, these architectures do not address the problem of neuron membrane potentials, which is a large data structure unique to SNNs with often high-precision requirements. Thus, the data movement of the partial membrane potential across multiple cores in each timestep can lead to significant memory bottleneck [30], and needs to be addressed in future SNN hardware implementations. On the other hand, analog computing based systems are susceptible to process variations and device/circuit non-idealities, which may lead to accuracy degradation. Moreover, such systems are largely dominated in area and power by the peripheral ADCs. These are some of the major challenges that need to be addressed for their widespread adoption.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation, Vannevar Bush Faculty Fellowship, Sandia National Laboratory, ONR MURI, and by the Center for Brain-Inspired Computing (C-BRIC), one of six centers in JUMP, funded by Semiconductor Research Corporation (SRC) and DARPA.

REFERENCES

[1] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud)*. IEEE, 2016, pp. 477–484.

[2] D. D. Cox and T. Dean, "Neural networks and neuroscience-inspired computer vision," *Current Biology*, vol. 24, pp. R921–R929, 2014.

[3] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[4] B. Rueckauer *et al.*, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[5] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[6] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," in *International Conference on Learning Representations*, 2020.

[7] C. Lee, A. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 366–382.

[8] A. L. Hodgkin and A. F. Huxley, "Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo," *The Journal of physiology*, vol. 116, no. 4, pp. 449–472, 1952.

[9] N. Rathi and K. Roy, "Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks," *arXiv preprint arXiv:2008.03658*, 2020.

[10] M. Zhang, Z. Gu, N. Zheng, D. Ma, and G. Pan, "Efficient spiking neural networks with logarithmic temporal coding," *IEEE Access*, 2020.

[11] A. Delorme *et al.*, "Networks of integrate-and-fire neurons using rank order coding b: Spike timing dependent plasticity and emergence of orientation selectivity," *Neurocomputing*, vol. 38, pp. 539–545, 2001.

[12] B. Han and K. Roy, "Deep spiking neural network: Energy efficiency through time based coding," in *European Conference on Computer Vision*, 2020.

[13] C. Brandli *et al.*, "A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct 2014.

[14] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[15] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.

[16] C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 3, pp. 384–394, 2018.

[17] T. P. Lillicrap *et al.*, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, no. 1, pp. 1–10, 2016.

[18] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," *arXiv preprint arXiv:1901.09948*, 2019.

[19] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020.

[20] W. Maass *et al.*, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002. [Online]. Available: https://doi.org/10.1162/089976602760407955

[21] P. Panda and N. Srinivasa, "Learning to recognize actions from limited training examples using a recurrent spiking neural model," *CoRR*, vol. abs/1710.07354, 2017.

[22] N. Soures and D. Kudithipudi, "Deep liquid state machines with neural plasticity for video activity recognition," *Frontiers in Neuroscience*, vol. 13, p. 686, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2019.00686

[23] W. Ponghiran, G. Srinivasan, and K. Roy, "Reinforcement learning with low-complexity liquid state machines," *Frontiers in Neuroscience*, vol. 13, p. 883, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2019.00883

[24] F. Xue, Hang Guan, and X. Li, "Improving liquid state machine with hybrid plasticity," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016, pp. 1955–1959.

[25] G. Srinivasan, P. Panda, and K. Roy, "Spilinc: Spiking liquid-ensemble computing for unsupervised speech and image recognition," *Frontiers in Neuroscience*, vol. 12, p. 524, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00524

[26] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[27] P. A. Merolla, J. V. Arthur *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[28] A. Agrawal, A. Ankit, and K. Roy, "Spare: Spiking neural network acceleration using rom-embedded rams as in-memory-computation primitives," *IEEE Transactions on Computers*, vol. 68, pp. 1190–1200, 2019.

[29] A. Ankit, I. E. Hajj *et al.*, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.

[30] S. Narayanan, K. Taht *et al.*, "Spinalflow: an architecture and dataflow tailored for spiking neural networks," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 349–362.