Reliability-Performance Trade-offs in Neuromorphic Computing

Twisha Titirsha and Anup Das
Electrical and Computer Engineering Drexel University, Philadelphia, PA, USA
Email: {tt624,anup.das}@drexel.edu

***Abstract*—Neuromorphic architectures built with Non-Volatile Memory (NVM) can significantly improve the energy efficiency of machine learning tasks designed with Spiking Neural Networks (SNNs). A major source of voltage drop in a crossbar of these architectures are the parasitic components on the crossbar's bitlines and wordlines, which are deliberately made longer to achieve lower cost-per-bit. We observe that the parasitic voltage drops create a significant asymmetry in programming speed and reliability of NVM cells in a crossbar. Specifically, NVM cells that are on shorter current paths are faster to program but have lower endurance than those on longer current paths, and vice versa. This asymmetry in neuromorphic architectures create reliability-performance trade-offs, which can be exploited efficiently using SNN mapping techniques. In this work, we demonstrate such trade-offs using a previously-proposed SNN mapping technique with 10 workloads from contemporary machine learning tasks for a state-of-the art neuromoorphic hardware.**

***Index Terms*—Neuromorphic Computing, Non-Volatile Memory (NVM), Phase-Change Memory (PCM), Endurance**

I. Introduction

Spiking Neural Networks (SNNs) are emerging machine learning models with spike-based computation and bio-inspired learning algorithms. Event-driven neuromorphic hardware such as TrueNorth [1], Loihi [2], and DYNAP-SE [3] implements biological neurons and synapses to execute SNN-based machine learning tasks in an energy-efficient manner. This makes neuromorphic hardware suitable for energy-constrained platforms such as the embedded systems [4] and edge devices of the Internet-of-Things (IoTs) [5].

A neuromorphic hardware is implemented as a tile-based architecture, the tiles are interconnected using a shared interconnect such as the Network-on-Chip (NoC) [6] and Segmented Bus [7]. Each tile consists of a crossbar, which can implement a fixed number of neurons and synapses. A crossbar in a neuromorphic hardware is an $n \times n$ organization, with n bitlines (columns) and n worklines (rows). A silicon neuron is mapped along each wordline of a crossbar, while a synaptic cell is placed at the cross-section of each bitline and wordline using an access device such as a transistor or a diode [8].

Recently, Non-Volatile Memory (NVM) such as Phase-Change Memory (PCM), Oxide-based Resistive RAM (OxRRAM), and and Spin-Transfer Torque Magnetic or Spin-Orbit-Torque RAM (STT- and SoT-MRAM) are used as synaptic cells to increase integration density and reduce energy consumption of crossbars in neuromorphic hardware [9]–[11].

A major source of voltage drops in a crossbar are the parasitic resistance and capacitance on its bitlines and wordlines, which are deliberately made longer to achieve lower cost-per-bit. In fact, for a PCM-based crossbar, each neuron is approximately 18x the size of a PCM cell [12]. To amortize this large size, systems designers implement larger crossbars, e.g., 128×128 for DYNAP-SE and 256×256 for TrueNorth. For such large crossbar sizes, the current on the longest path in a crossbar becomes significantly lower than the current on its shortest path for the same spike voltage generated from a neuron and the same conductance programmed on the enabled synaptic cell in these paths.[1]

Current asymmetry leads to a difference in performance and reliability of NVM cells. Higher current through an NVM cell can lead to faster programming of the cell. This means that NVM cells on shorter current paths are faster to access and program. However, NVMs also have limited endurance, ranging from 10^5 (for Flash) to 10^{10} (for OxRRAM), with PCM somewhere in between ($\approx 10^7$). An NVM cell's endurance is strongly dependent on the programming current. We build the case for PCM, where the conductance change is induced by Joule heating of the chalcogenide material in the cell. The endurance of the material depends on the self-heating temperature, which is dependent on the programming current. Therefore, the NVM cells on shorter current paths have higher self-heating temperature, and therefore lower endurances.

In recent years, many approaches are proposed to map SNNs to neuromorphic hardware. This includes the performance-oriented SNN mapping technique of [13], [14], the dataflow-based mapping technique of [15], [16], the energy-aware mapping technique of [17]–[19], the circuit aging-aware mapping technique of [20]–[23], and the run-time SNN mapping technique of [24]. Unfortunately, none of these approaches exploit the reliability and performance trade-offs of NVM cells in neuromorphic computing. In this paper, we take one such mapping approach – SpiNeMap, and show the significant variations in endurance and speed during its mapping explorations.

The remainder of this paper is organized as follows. We provide a background of PCM and neuromorphic architectures in Section II. Next, we formulate the endurance-access speed trade-offs for a single PCM cell and integrate such trade-offs at the crossbar-level in Section III. Next, we discuss the mapping exploration of SpiNeMap in Section IV. We present our evaluation in Section V and conclusion in Section VI.

[1]The length of a current path in a crossbar is measured in terms of the number of parasitic components that are encountered on the path.

II. Background

In this section, we discuss the background on Phase-Change Memory (PCM) to aid the understanding of the trade-offs in neuromorphic computing. We also provide a discussion on machine learning approaches using SNNs, and how such approaches can be mapped to hardware consisting of PCM cells organized into crossbars.

A PCM cell is built with chalcogenide alloy, e.g., $Ge_2Sb_2Te_5$ (GST) [25], and is connected to a bitline and a wordline using an access device. The GST alloy can either be in an amorphous (high resistance) state, or in one of the partially crystallized (low resistance) states. PCM is recently explored as scalable DRAM alternative for conventional computing [26]–[31]. This work explores PCM for neuromorphic computing. For such computing architectures, the weight of a synaptic connection is programmed as conductance of a PCM cell by driving current and inducing Joule heating in the cell.

In many machine learning approaches such as online learning using Spike-Timing Dependent Plasticity (STDP) [32], one-shot learning [33], life-long learning [34], and reinforcement learning [35], it is necessary to update synaptic weights based on the input excitation. To facilitate such synaptic updates, a PCM cell's state must be switched by driving current through it using the spikes generated from neurons. However, frequent switching of a PCM cell's state may lead to endurance issues, where the cell fails to be programmed correctly, leading to a degradation of machine learning performance. Furthermore, a key requirement in such online learning use-cases is their real-time performance, i.e., the weight updates must be completed within a small time interval.

To understand the workload-dependent performance and endurance trade-offs associated with PCM cells in a neuromorphic architecture, Figure 1a shows a simple SNN with two input and one output neurons. Figure 1b illustrates the mapping of this SNN to a crossbar. As seen from this figure, the synaptic weights w_1 and w_2 are programmed as conductances. The spike voltages are multiplied with the conductances to generate current, which gets integrated along the columns. The current strength guides the update of conductance of the PCM cell enabled along the current path. Clearly, the weight update frequency depends on the spikes generated from the hardware neurons mapped along the rows of a crossbar. The latter depends on how neurons and synapses of a machine learning model, e.g., Figure 1a are mapped to the corresponding resources on a crossbar. To elaborate on this, Figure 1c illustrates the utilization of a different set of PCM cells to realize the SNN of Figure 1a. If the PCM cells in a crossbar have different performance and endurance characteristics (which we demonstrate in this work), then the mapping of neurons and synapses of a machine learning model plays a critical role in system-level performance and reliability.

III. Endurance-Performance Trade-offs in Neuromorphic Hardware

We formulate the endurance-performance trade-offs for a single PCM cell. To establish the relationship, we consider the GST material of a PCM cell to be in a crystalline state. The

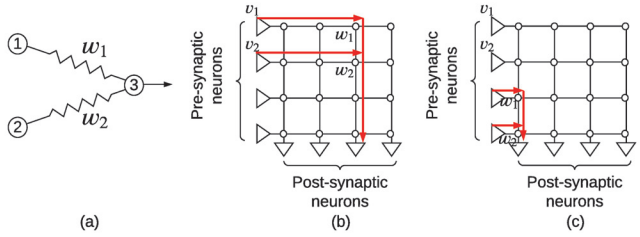

Fig. 1. (a) A simple spiking neural network, (b) Mapping of the network to a crossbar, (c) A different mapping of the network to the hardware.

amorphization process, i.e., the crystalline-to-amorphous state transition involves driving a very high current through the cell for a short duration. This high current raises the temperature of the GST material through Joule heating in the heater attached to the GST, which transitions the material to its amorphous state. The crystalline fraction (V_c) is computed using the Johnson-Mehl-Avrami (JMA) equation [36] as

$$V_c = \exp\left[-\alpha \times \frac{(T_{SH} - T_{amb})}{T_m} \times t\right], \tag{1}$$

where t is the time, T_m is the melting temperature of the GST material, and α is a fitting constant. The exponential decay of V_c in Equation 1 implies that, higher the self-heating temperature (T_{SH}), faster is the reduction of the crystalline volume, i.e., faster is the amorphization process.

The self-heating temperature is related to the square of programming current (I_{prog}) as

$$T_{SH} = k \cdot I_{\text{prog}}^2 \tag{2}$$

where k is a constant.

From Equations 1 and 2 we can conclude that higher the programming current, higher is the self-heating temperature, and hence, faster is the programming of the cell.

However, with increase in self-heating temperature, the endurance of a PCM cell reduces. Using the phenomenological endurance model [37], endurance of a PCM cell can be expressed as

$$\text{Endurance} \approx \exp\left(\frac{\gamma}{T_{SH}}\right), \tag{3}$$

where γ is a fitting parameter.

From Equations 2 and 3, we conclude that higher the programming current, higher is the self-heating temperature and therefore, lower is the endurance.

Figure 2 shows the current through the PCM cells in a 128x128 PCM crossbar. This current variation is due to the difference in the length of current paths from pre-synaptic neurons to post-synaptic neurons in the crossbar, where the length of a current path is measured in terms of the number of parasitic elements on the path. These current values are obtained for a 65nm technology node and at 300K temperature corner. As can be clearly seen from the figure, current through PCM cells on the top-right corner of the crossbar is lower than through PCM cells located at the bottom-left corner. Therefore, cells at the top-right corner are slower to program and have higher endurance, while those at the bottom-left corner are

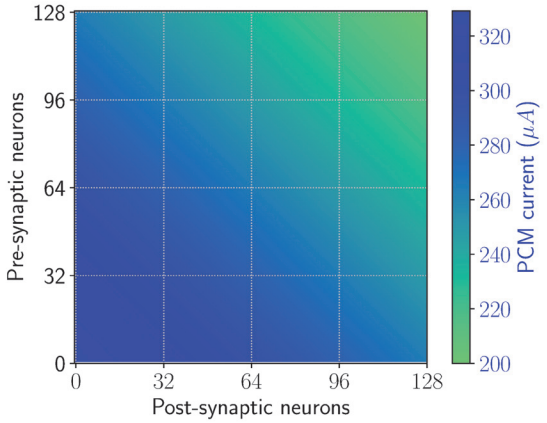

Fig. 2. Current map in a 128x128 crossbar.

faster to program and have lower endurance. Table I summarizes these findings.

TABLE I
SUMMARY OF PERFORMANCE-ENDURANCE TRADE-OFFS.

Location	Performance	Endurance
Top-right corner	Low	High
Bottom-left corner	High	Low

IV. Mapping Explorations

In this section, we present the mapping exploration of SpiNeMap [18] and show the performance-endurance trade-offs that are obtained during its design-space exploration.

SpiNeMap uses an instance of the Particle Swarm Optimization (PSO) [38], a meta-heuristic approach to map neurons and synapses to the hardware. To this end, SpiNeMap first partitions a spiking neural network based application into clusters, where each cluster can fit onto the resources of a crossbar. The clusters are then mapped to the crossbars using the PSO. In general, PSO finds the optimum solution to a fitness function F. Each solution is represented as a particle in the swarm. Each particle has a velocity with which it moves in the search space to find the optimum solution. During the movement, a particle updates its position and velocity according to its own experience (closeness to the optimum) and also experience of its neighbors. We introduce the following notations for PSO.

$$\begin{aligned} D &= \text{dimensions of the search space} \\ n_p &= \text{number of particles in the swarm} \\ \boldsymbol{\Theta} = \{\theta_l \in \mathbb{R}^D\}_{l=0}^{n_p-1} &= \text{positions of particles in the swarm} \\ \mathbf{V} = \{\mathbf{v}_l \in \mathbb{R}^D\}_{l=0}^{n_p-1} &= \text{velocity of particles in the swarm} \end{aligned} \tag{4}$$

Position and velocity updates are performed according to the following equation.

$$\begin{aligned} \boldsymbol{\Theta}(t+1) &= \boldsymbol{\Theta}(t) + \mathbf{V}(t+1) \\ \mathbf{V}(t+1) &= \mathbf{V}(t) + \varphi_1 \cdot \Big(P_{\text{best}} - \boldsymbol{\Theta}(t)\Big) + \varphi_2 \cdot \Big(G_{\text{best}} - \boldsymbol{\Theta}(t)\Big) \end{aligned} \tag{5}$$

where t is the iteration number, φ_1, φ_2 are constants and P_{best} (and G_{best}) is the particles own (and neighbors) experience. In Figure 3, we illustrate the iterative approach to find an optimal solution using PSO. The PSO algorithm starts with an initial neighborhood of swarms. In this example, we illustrate 3 swarms, each with 3 particles (see Figure 3a). Each particle jumps to a new location with a velocity determined as a function of local best (within swarms), and global best. This continues until the sub-swarms converge (see Figure 3b). In the third step, the swarm regroups and the position and velocity update steps are repeated (see Figure 3c). We continue these iterations until a predefined convergence criteria is reached.

SpiNeMap uses PSO to minimize the number of spikes communicated on the global interconnect, which leads to a reduction in the energy consumption.

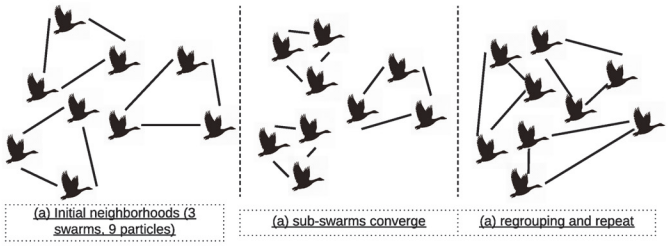

Fig. 3. Illustrating the iterative steps of PSO.

Figure 4 illustrates the performance-endurance trade-offs obtained during the mapping exploration of SpiNeMap. The figure plots the performance and endurance obtained for different design solutions generated during the design-space exploration using the PSO. The figure also shows two solutions – one with highest endurance and one with the highest performance. We note that the highest performance mapping is generated by DFSynthesizer [13], which maps neurons and synapses to hadware, minimizing the execution time of applications.

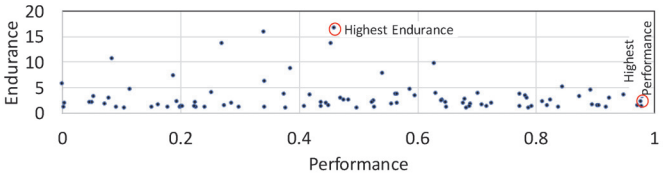

Fig. 4. Performance-endurance trade-offs using SpiNeMap.

V. Evaluation

A. Evaluation Framework

We evaluated 10 machine learning applications that are representative of three most commonly used neural network classes — convolutional neural network (CNN), multi-layer perceptron (MLP), and recurrent neural network (RNN). These applications are 1) LeNet [39] based handwritten digit recognition with 28×28 images of handwritten digits from the MNIST dataset [40]; 2) AlexNet [41] for Imagenet classification [42]; 3) VGG16 [43], also for Imagenet classification [42]; 4) ECG-based heart-beat classification (HeartClass) [44], [45] using electrocardiogram (ECG) data from the Physionet database [46];

5) multi-layer perceptron (MLP)-based handwritten digit recognition (MLP-MNIST) [47] using the MNIST database; 6) edge detection (EdgeDet) [48] on 64×64 images using difference-of-Gaussian; 7) image smoothing (ImgSmooth) [48] on 64×64 images; 8) heart-rate estimation (HeartEstm) [49] using ECG data; 9) RNN-based predictive visual pursuit (VisualPursuit) [50]; and 10) recurrent digit recognition (R-DigitRecog) [47]. Table II summarizes the topology, the number of neurons and synapses of these applications, and their baseline accuracy. To demonstrate the trade-offs, we enable STDP-based weight updates [32] in each of these applications.[2] But our approach is not limited to STDP.

TABLE II
EVALUATED APPLICATIONS.

Class	Applications	Synapses	Neurons	Topology	Accuracy
CNN	LeNet [39]	282,936	20,602	CNN	85.1%
	AlexNet [41]	38,730,222	230,443	CNN	90.7%
	VGG16 [43]	99,080,704	554,059	CNN	69.8 %
	HeartClass [45]	1,049,249	153,730	CNN	63.7%
MLP	DigitRecogMLP	79,400	884	FeedForward (784, 100, 10)	91.6%
	EdgeDet [48]	114,057	6,120	FeedForward (4096, 1024, 1024, 1024)	100%
	ImgSmooth [48]	9,025	4,096	FeedForward (4096, 1024)	100%
RNN	HeartEstm [49]	66,406	166	Recurrent Reservoir	100%
	VisualPursuit [50]	163,880	205	Recurrent Reservoir	47.3%
	R-DigitRecog [47]	11,442	567	Recurrent Reservoir	83.6%

We model the DYNAP-SE neuromorphic hardware [3] with the following configurations.

- A tiled array of 4 tiles, each with a 128x128 crossbar. There are 65,536 memristors per crossbar.
- Spikes are digitized and communicated between cores through a mesh routing network using the Address Event Representation (AER) protocol.
- Each synaptic element is a PCM-based memristor.

Table III reports the hardware parameters of DYNAP-SE.

TABLE III
MAJOR SIMULATION PARAMETERS EXTRACTED FROM [3].

Neuron technology	65nm CMOS
Synapse technology	PCM
Supply voltage	1.0V
Energy per spike	50pJ at 30Hz spike frequency
Energy per routing	147pJ
Switch bandwidth	1.8G. Events/s

We evaluate the following metrics.

- **Performance:** This is the time it takes to execute an application on the hardware model.
- **Effective lifetime:** This is the minimum effective lifetime of all PCM cells in the hardware. The *effective lifetime* ($\mathcal{L}_{i,j}$), defined for the PCM cell connecting the i^{th} pre-synaptic neuron with j^{th} post-synaptic neuron in a memristive crossbar as

$$\mathcal{L}_{i,j} = \mathcal{E}_{i,j}/a_{i,j}, \tag{6}$$

where $a_{i,j}$ is the number of spikes propagating through the PCM cell in a given SNN workload and $\mathcal{E}_{i,j}$ is its endurance.

[2]Spike-Timing Dependent Plasticity (STDP) [51] is a learning mechanism in SNNs, where the synaptic weight between a pre- and a post-synaptic neuron is updated based on the timing of pre-synaptic inputs relative to the post-synaptic spike.

B. Performance

Figure 5 compares the performance of DFSynthesizer, a performance-oriented technique to map SNNs to neuromorphic hardware and SpiNeMap, which minimizes the number of spikes on the shared interconnect. We observe that compared to DFSynthesizer, the performance using SpiNeMap is an average 10% lower for these applications.

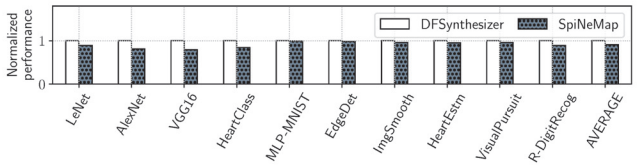

Fig. 5. Performance normalized to DFSynthesizer.

C. Effective Lifetime

Figure 6 plots the normalized lifetime of DFSynthesizer and SpiNeMap for the evaluated applications. Lifetime results are normalized to the lifetime obtained using the mapping that generates the highest effective lifetime (see Figure 4). We observe that lifetime using the mapping of DFSynthesizer is on average 30% lower, while that using SpiNeMap is 19% lower than the highest lifetime.

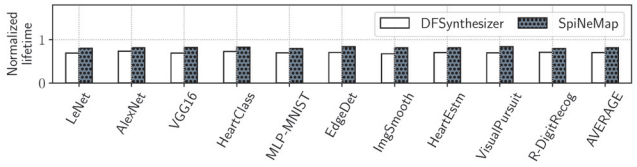

Fig. 6. Lifetime normalized to mapping with highest lifetime.

VI. CONCLUSIONS

In this work, we show the trade-offs between performance and lifetime of neuromorphic hardware with PCM-based crossbars. Specifically, we show that in a PCM-based crossbar, the PCM cells that are located on the bottom-left corner are faster to access but have lower lifetime than PCM cells on the top-right corner, which are slower but have higher lifetime. Existing SNN-mapping techniques do not explore this trade-offs in mapping neurons and synapses to hardware. The design space exploration of these mapping techniques often select mapping that generate high performance or optimize for energy consumption. Therefore, the lifetime obtained using these techniques is significantly lower than the highest lifetime. A possible future direction is therefore, to explore the trade-offs during the design-space exploration. This will enable generating SNN mapping that are balanced in terms of lifetime, performance, and energy consumption.

ACKNOWLEDGMENT

This work is supported by the National Science Foundation Award CCF-1937419 (RTML: Small: Design of System Software to Facilitate Real-Time Neuromorphic Computing).

REFERENCES

[1] M. V. Debole *et al.*, "TrueNorth: Accelerating from zero to 64 million neurons in 10 years," *Computer*, 2019.
[2] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, 2018.
[3] S. Moradi *et al.*, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *TBCAS*, 2017.
[4] E. A. Lee *et al.*, *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.
[5] W. Shi *et al.*, "Edge computing: Vision and challenges," *IOTJ*, 2016.
[6] L. Benini *et al.*, "Networks on chip: A new paradigm for systems on chip design," in *DATE*, 2002.
[7] A. Balaji *et al.*, "Exploration of segmented bus as scalable global interconnect for neuromorphic computing," in *GLSVLSI*, 2019.
[8] F. Catthoor *et al.*, "Very large-scale neuromorphic systems for biological signal processing," in *CMOS Circuits for Biological Sensing and Processing*, 2018.
[9] G. W. Burr *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, 2017.
[10] A. Mallik *et al.*, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSIT*, 2017.
[11] P. Wijesinghe *et al.*, "An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain," *TETCI*, 2018.
[12] G. Indiveri, "A low-power adaptive integrate-and-fire neuron circuit," in *ISCAS*, 2003.
[13] S. Song *et al.*, "Compiling spiking neural networks to neuromorphic hardware," in *LCTES*, 2020.
[14] A. Balaji *et al.*, "Enabling resource-aware mapping of spiking neural networks via spatial decomposition," *Embedded Systems Letters*, 2020.
[15] A. Das *et al.*, "Dataflow-based mapping of spiking neural networks on neuromorphic hardware," in *GLSVLSI*, 2018.
[16] A. Balaji *et al.*, "A framework for the analysis of throughput-constraints of snns on neuromorphic hardware," in *ISVLSI*, 2019.
[17] A. Balaji *et al.*, "PyCARL: A PyNN interface for hardware-software co-simulation of spiking neural network," in *IJCNN*, 2020.
[18] A. Balaji *et al.*, "Mapping spiking neural networks to neuromorphic hardware," *TVLSI*, 2020.
[19] A. Das *et al.*, "Mapping of local and global synapses on spiking neuromorphic hardware," in *DATE*, 2018.
[20] A. Balaji *et al.*, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *CAL*, 2019.
[21] S. Song *et al.*, "Improving dependability of neuromorphic computing with non-volatile memory," in *EDCC*, 2020.
[22] S. Song *et al.*, "A case for lifetime reliability-aware neuromorphic computing," in *MWSCAS*, 2020.
[23] T. Titirsha *et al.*, "Thermal-aware compilation of spiking neural networks to neuromorphic hardware," in *LCPC*, 2020.
[24] A. Balaji *et al.*, "Run-time mapping of spiking neural networks to neuromorphic hardware," *JSPS*, 2020.
[25] S. Ovshinsky, "Reversible electrical switching phenomena in disordered structures," *Physical Review Letters*, 1968.
[26] B. C. Lee *et al.*, "Architecting phase change memory as a scalable dram alternative," in *ISCA*, 2009.
[27] M. K. Qureshi *et al.*, "Scalable high performance main memory system using phase-change memory technology," in *ISCA*, 2009.
[28] S. Song *et al.*, "Enabling and exploiting partition-level parallelism (palp) in phase change memories," *TECS*, 2019.
[29] S. Song *et al.*, "Exploiting inter-and intra-memory asymmetries for data mapping in hybrid tiered-memories," in *ISMM*, 2020.
[30] S. Song *et al.*, "Improving phase change memory performance with data content aware access," in *ISMM*, 2020.
[31] S. Song *et al.*, "Aging Aware Request Scheduling for Non-Volatile Main Memory," in *ASP-DAC*, 2021.
[32] S. R. Kheradpisheh *et al.*, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, 2018.
[33] L. Fei-Fei *et al.*, "One-shot learning of object categories," *TPAMI*, 2006.
[34] J. M. Allred *et al.*, "Stimulating stdp to exploit locality for lifelong learning without catastrophic forgetting," Purdue University West Lafayette United States, Tech. Rep., 2019.
[35] M. S. Shim *et al.*, "Biologically inspired reinforcement learning for mobile robot collision avoidance," in *IJCNN*, 2017.
[36] M. Avrami, "Granulation, phase change, and microstructure kinetics of phase change. III," *The Journal of Chemical Physics*, 1941.
[37] D. B. Strukov, "Endurance-write-speed tradeoffs in nonvolatile memories," *Applied Physics A: Materials Science and Processing*, 2016.
[38] J. Kennedy, "Particle swarm optimization," *Encyclopedia of machine learning*, 2010.
[39] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, 2015.
[40] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, 2012.
[41] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems (NeurIPS)*, 2012.
[42] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.
[43] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *arXiv*, 2014.
[44] A. Das *et al.*, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ecg," in *CHASE*, 2018.
[45] A. Balaji *et al.*, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *JOLPE*, 2018.
[46] G. B. Moody *et al.*, "Physionet: a web-based resource for the study of physiologic signals," *IEEE Engineering in Medicine and Biology Magazine*, 2001.
[47] P. U. Diehl *et al.*, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, 2015.
[48] T. Chou *et al.*, "CARLsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters," in *International Joint Conference on Neural Networks (IJCNN)*, 2018.
[49] A. Das *et al.*, "Unsupervised heart-rate estimation in wearables with Liquid states and a probabilistic readout," *Neural Networks*, 2018.
[50] H. J. Kashyap *et al.*, "A recurrent neural network based model of predictive smooth pursuit eye movement in primates," in *International Joint Conference on Neural Networks (IJCNN)*, 2018.
[51] Y. Dan *et al.*, "Spike timing-dependent plasticity of neural circuits," *Neuron*, vol. 44, pp. 23–30, 2004.