# In-Memory Computing in Emerging Memory Technologies for Machine Learning: An Overview

Kaushik Roy, Indranil Chakraborty, Mustafa Ali, Aayush Ankit and Amogh Agrawal
(All authors contributed equally)
kaushik@purdue.edu
School of Electrical and Computer Engineering, Purdue University
West Lafayette, Indiana

## ABSTRACT

The saturating scaling trends of CMOS technology have fuelled the exploration of emerging non-volatile memory (NVM) technologies as a promising alternative for accelerating data intensive Machine Learning (ML) workloads. To that effect, researchers have explored special-purpose accelerators based on NVM crossbar primitives. NVM crossbars have high storage density and can efficiently perform massively parallel in-situ Matrix Vector Multiplication (MVM) operations, the key computation in ML workloads, helping overcome the memory bottleneck faced by von Neumann architectures. Despite the promises, analog computing nature of NVM crossbars can lead to functional errors due to device and circuit non-idealities such as parasitic resistances and device non-linearities. Moreover, NVM crossbars need high cost peripheral circuitry to be integrated in large scale systems. Hence, there is a need to study different levels of the design stack to realize the potential of this technology.

In this paper, we present an overview of in-memory computing in NVM crossbars for ML workloads. We discuss the basic anatomy of NVM crossbars and highlight the challenges faced at the primitive level. Next, we present how the high storage density of NVM crossbars can enable spatially distributed architectures. Further, we present various modeling and evaluation tools which can effectively help us study the functionality as well as performance of NVM crossbar systems. Finally, we provide an outlook on the future research directions in this field.

## 1 INTRODUCTION

The proliferating application space and unprecedented success of Deep Neural Networks (DNNs) [1] has been accompanied by a steady growth in the computational complexity of model sizes. This necessitates the development of special-purpose accelerators to improve the energy efficiency and latency of running DNN workloads over traditional computing systems. One key aspect of such accelerators is performing efficient matrix-vector multiplication (MVM) operations, the primary computational kernel in DNNs, through intertwining of memory and processing elements to overcome the off-chip memory bottleneck. Logically, this concept can be extended to a new paradigm, 'in-memory computing', where the computations are performed within the memory array itself.

In-memory computing has been extensively explored in CMOS-based SRAMs [2, 3], both for logical and arithmetic operations. Although CMOS memories can be suitable for Boolean and low-precision arithmetic computations, their slowing scaling trends and limited density has motivated researchers to explore in-memory computing in other emerging non-volatile memory (NVM) technologies such as PCM [4], RRAM [5], Spintronics [6]. NVM devices can form a two-dimensional crossbar array. As NVM devices can potentially store multiple states, NVM crossbars are dense with bit-cell area roughly $\sim 4F^2$ ($16F^2$ with selectors [4]). NVM crossbars can perform parallelized in-situ MVM operations, resulting in higher energy efficiency and speed than digital accelerators by eliminating sequential memory accesses. Also, the high density of NVM crossbars can enable a large on-chip data storage allowing for weight stationary ML architectures, significantly relaxing the requirements for off-chip memory accesses. This immense promise has led to the exploration of analog in-memory computing systems [7, 8] that leverage the high density and massively parallel MVM operations offered by NVM crossbars.

Despite its promises, there are several challenges toward making in-memory computing with NVM technologies feasible for large scale systems. First, the inherent analog nature of computing can lead to functional errors due to device and circuit non-idealities. Such non-idealities are i) read non-idealities affecting the inference operations in neural networks and ii) write non-idealities affecting training. Read non-idealities [9, 10] can arise from metal line resistance, source resistance in input drivers and sink resistance in sensing circuits as well as non-linear characteristics of NVM and selector devices. Write non-idealities originate due to non-linear conductance update trajectories of NVM devices [11]. Besides these non-idealities, there are further technological challenges such as reliability as well as resistance change over time and temperature which can cause errors. In large-scale DNNs, the errors at crossbar output can accumulate across the layers, resulting in a degradation in application performance. Secondly, due to the discrepancy of DNN model sizes and physical sizes of NVM crossbars, partial outputs produced by multiple NVM crossbars need to be communicated and processed through analog to digital and digital to analog converters (ADCs and DACs). However, ADCs are shown to consume upto 80% of the total energy and 70% of the area of a MVM core based on NVM crossbars [7]. This drastically reduces the benefits of analog computing systems over digital accelerators. Hence, there is a need to explore hardware-software co-design techniques which can reduce the overhead of ADCs.

In this paper, we present an overview of in-memory computing systems for ML workloads, highlighting the devices, circuits and architectures that form the basis of such systems. First, we present various NVM technologies and analyze their suitability in the context of array design. Next, we delineate the various components that constitute the basic MVM computation fabric. Further, we explore spatial architectures which enable mapping of ML applications on NVM crossbars. Finally, we summarize the paper by a brief outlook on the future directions of this field.

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

**Table 1: Comparison of various NVM Technologies explored in the literature for developing resistive crossbars.**

| Property | PCM | RRAM | MTJ |
|---|---|---|---|
| **Multi-level Cell** | Yes | Yes | No |
| **Storage Density** | High | High | High |
| $R_{ON}/R_{OFF}$ | High | High | Low |
| **Write Energy** | 6 nJ | 2 nJ | <1 nJ |
| **Write Latency** | 150 ns | 100 ns | 10 ns |
| **Endurance** | $10^7$ cycles | $10^5$ cycles | $10^{15}$ cycles |

## 2 NON-VOLATILE MEMORY TECHNOLOGIES

Non volatile memory technologies can offer low leakage, high density, and energy-efficient analog computing. There have been several proposals for NVM devices spanning phase-change materials (PCM) [4], resistive memories (RRAMs)[5], and spintronics (STT-MRAM) [6]. In this section, we provide an overview of the basic operation and properties of each of these technologies, in light of NVM crossbars. Table 1 summarizes and compares the various properties and metrics of PCM, RRAM and MTJ device technologies.

### 2.1 Phase-Change Materials (PCM)

PCMs have the property to modulate the conductance based on the material phase. In amorphous state (crystalline state), they exhibit high-resistance (low-resistance). PCMs can be switched from one state to the other by heat, by applying a series of low- and high-amplitude voltage pulses. The relative volumes of the switching domain (amorphous/crystalline) allow multi-level conductance states to be stored [4]. PCMs offer high ON/OFF ratios, high density [12] and scalability, making them ideal candidates for crossbars [13]. However, several technology challenges, such as conductance drift over time, high write energy (~6 nJ) and latency (~150 ns), and low endurance (~ $10^7$ cycles) [4], can adversely affect functionality.

### 2.2 Resistive RAM (RRAM)

An RRAM device is based on a typical metal-insulator-metal structure. By applying a series of voltage pulses, a soft breakdown of the dielectric occurs, creating multi-level conductance levels [14]. RRAMs also offer high ON/OFF ratios, high density and have been demonstrated in crossbars of sizes up-to 128x128 [15]. However, they suffer from low endurance (~ $10^5$ cycles), and high write energy (~2 nJ) and latency (~100 ns).

### 2.3 Spintronics

Spintronic devices are based on the Magnetic Tunnel Junction (MTJ) structure. It consists of an MgO tunneling barrier sandwiched between two ferromagnetic (FM) layers. One of the FM layers has pinned magnetic orientation, while the other is free, which can be switched by applying a current through the device, causing spin-transfer torque [16]. Unlike PCMs/RRAMs, an MTJ has two stable conductance states, depending on the relative orientation of the FM layers (low-resistance parallel and high-resistance anti-parallel orientation). MTJs offer high endurance (~ $10^{15}$ cycles) and low write latency (~10 ns), but suffer from low ON/OFF ratio, which makes it prone to process variations in crossbars.

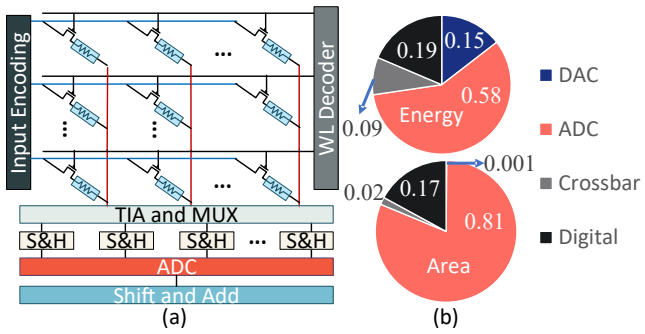


**Figure 1: NVM crossbar and its peripherals for large-scale integration.**

## 3 NVM CROSSBARS

Fig. 1a shows the NVM crossbar organization where NVM cells are connected in a matrix fashion where each cell is attached to its corresponding word-line (WL) and bit-line (BL). Fundamentally, MVM computations are performed in NVM crossbars by depending on Ohm's Law for multiplication, and Kirchhoff's Current Law (KCL) for accumulation. NVM cells usually have access devices, which can be a transistor or selector needed to eliminate sneak paths during write process[7]. In this section, we discuss the basic anatomy of NVM crossbar as a MVM primitive.

### 3.1 Crossbar Array and Peripherals

The input vector to an NVM crossbar can be encoded in either voltage amplitude, or signal pulse width [17]. In the more popular voltage encoding schemes, analog voltages representing the input vectors are applied to WLs, hence the current passing through the cell is the multiplication of the applied voltage ($V$) and the NVM conductance ($G$). Additionally, KCL sums the currents from each cell in a column so that the current at BL ($I_{BL}$) represents the multiply-and-accumulate (MAC) output.

NVM crossbars require peripheral circuits for accurate output reading and analog-to-digital interfacing for communication. As shown in Fig. 1, BLs are connected to trans-impedance amplifiers (TIA) to convert the output current $I_{BL}$ into voltage. Commonly-adopted TIAs consist of an operational amplifier with a negative-feedback resistor ($R_F$) to fix the voltage BL and produce an output voltage of $V_{TIA} \approx -I_{BL}R_F$. Afterwards, $V_{TIA}$ is converted to digital through an analog to digital converter (ADC) for communicating the MAC outputs to other units in the system. Several ADC architectures have been adopted in NVM crossbars including Successive Approximation Register (SAR) and Flash ADCs [18, 19]. SAR ADCs are commonly adopted in NVM crossbars [7, 8, 20] due to its relatively simple structure and low area/energy overhead compared to Flash ADCs. On the other hand, Flash ADCs provide higher conversion speed and suffer from limited precision. Note, the ADC precision required for ideal MVM computation in an NxN crossbar with K input precision and M weight precision is $\lceil \log_2((2^K-1)*(2^M-1)*N) \rceil$. However, due to crossbar device and circuit-level nonidealities (described in the next sub-section) and data-sparsity, recent proposals [20, 21] adopt lower-than-ideal ADC precision for better performance/energy. As shown in Fig. 1b, ADC dominates the crossbar area and energy [8].

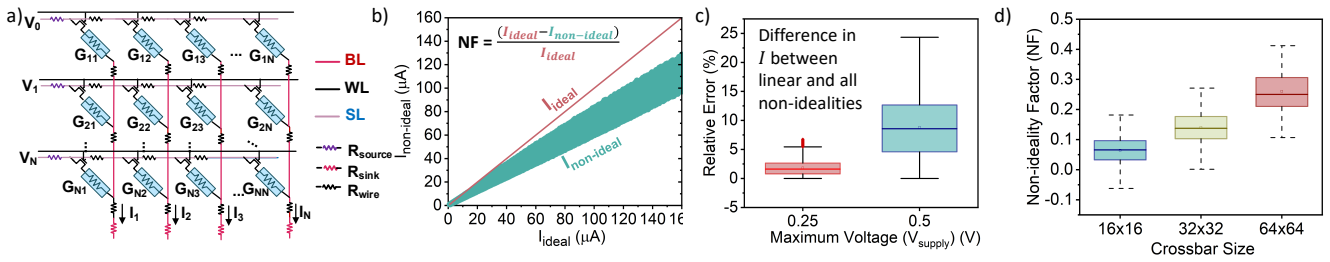


**Figure 2: a) Non-ideal Crossbar, b) Deviation of non-ideal current from ideal current, c) Error in current when all non-idealities v/s only linear non-idealities are considered, d) Impact of crossbar-size on NF.**

## 3.2 Non-idealities in NVM Crossbars

We now focus on functional errors arising from device and circuit non-idealities due to the analog nature of computing, shown in Fig. 4 (a). Device-circuit non-idealities can be broadly categorized into 2 types: 1) Read non-idealities, and 2) Write non-idealities.

*3.2.1* ***Read and Write Non-idealities.*** Generally, read non-idealities are functional errors in the MVM outputs that occur during the read operation of NVM crossbars. Read non-idealities can be categorized into: 1) Linear read non-idealities, 2) Non-linear read non-idealities.

**1. Linear Read Non-idealities:** Linear read non-idealities refer to parasitic resistances such as metal line, source, and sink resistances as shown in Fig. 4 (a). Metal line resistances originate from interconnect parasitics. Source resistance represents the output resistance of the input drivers while sink resistance results from the input resistance of the BL sensing circuits. Linear non-idealities lead to undesirable voltage drops causing erroneous MVM outputs.

**2. Non-Linear Read Non-idealities:** Non-linear read non-idealities originate from the non-linear current-voltage characteristics of both NVM devices and access transistors. Their characteristics depend on device technology, and result in deviation from the desired linear resistive behavior of the NVM devices [22]. Thus, MVM outputs can be significantly affected by such non-linearities since they alter the effective cell conductances, G. Additionally, non-linear characteristics of access devices such as selectors or transistors pose challenges toward MVM functionality.

**3. Write Non-idealities:** Write non-idealities impact conductance tuning operations. Their effect is crucial in case of on-device training where conductance update mechanism, desired to be linear and symmetric, exhibits non-linear behavior. Such effect has been modeled by researchers in [23]. In addition, the write mechanism in NVM crossbars can suffer from sneak path issues, which can lead to unreliable conductance update [24].

*3.2.2* ***Impact of non-idealities on MVM outputs.*** The aforementioned read and write non-idealities have detrimental impact on MVM outputs and conductance updates respectively.

**1. Read non-idealities:** Here, we study the combined impact of all read non-idealities through HSPICE simulations of 64x64 1T-1R RRAM crossbars. In Fig. 4 (b), we observe that different V and G combinations result in various non-ideal currents ($I_{non-ideal}$) for the same ideal current ($I_{ideal}$). Let us define a non-ideality factor (NF): $(I_{ideal} - I_{non-ideal})/I_{ideal}$. The difference between the impact of i) linear non-idealities and ii) both linear and non-linear non-idealities, is depicted in Fig. 4 (c). Moreover, design parameters such as crossbar size, ON resistance of the device, ON/OFF ratio of the technology have varying impacts on the output currents in presence of read non-idealities. The impact of crossbar size on NF is shown in Fig. 4 (d).

**2. Write non-idealities:** The impact of write non-idealities affect device conductance during on-line training which involves multiple conductance updates. Device write non-linearity and asymmetry result in varying errors in the final conductance values after multiple updates for different degrees of non-linearity [11].

# 4 CROSSBAR ARCHITECTURES

Typical machine learning workloads have 100s of million parameters, thereby requiring off-chip memory for model storage. Further, several applications (batch size 1) can have low data reuse [8]. Consequently, the execution of ML workloads on CMOS-based systems are dominated by off-chip memory accesses. To this effect, the in-memory computing benefits of resistive crossbars can be leveraged to design ML accelerators for overcoming the off-chip memory bottlenecks. However, expensive memory writes limit the applicability of a time-multiplexed architecture, where the crossbars are reused across layers by re-programming weight matrices and executing the corresponding MVM operations. Consequently, a spatial architecture where the DNN is partitioned such that the weights (stationary data) are pinned to crossbars located across multiple cores is more efficient as it leverages the benefits of high storage density while alleviating the costly writes.

## 4.1 Inference accelerator

In an inference workload, crossbars typically perform MVM operations with stationary matrices, i.e., the DNN weights. Thus, the spatial architecture can be mapped with DNN weights during configuration (writing weights to crossbars), and can be subsequently used for millions of inferences. In addition to the MVM operations performed by crossbars, DNNs require several other operations such as vector additions, vector multiplications, scalar and non-linear operations. Further, other non-stationary data such as inputs and partial sums need to be stored and moved. Consequently, designing an inference accelerator requires augmenting the resistive crossbar with CMOS based digital logic and memory units.

*4.1.1* ***Micro-architecture of a typical Crossbar-based accelerator:*** Typical ReRam-based accelerators are spatial, and organized in hierarchies: core, multi-core, node [7, 8, 25]. Figure 3(a) shows the logical organization of a core, which consists of an instruction execution pipeline, digital CMOS based functional unit, and analog crossbars based MVM unit. The instruction execution pipeline and specialized instruction set architecture (ISA) enable

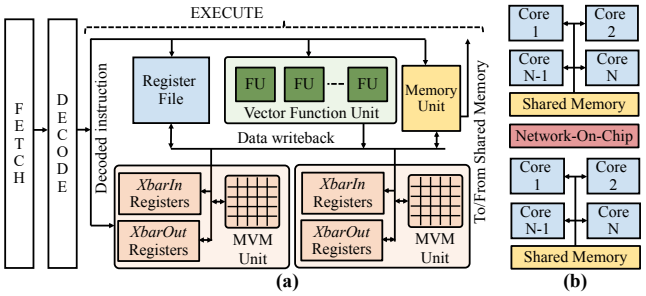


**Figure 3: Logical organization of crossbar-based architecture based on hybrid CMOS-NVM technology [8]**

representing multiple ML workloads using a small set of custom instructions. Such programmability is important to ensure that an accelerator can cater to the ever increasing variety of ML workloads. The choice of custom instructions provide a trade-off between efficiency and accelerator's generality. The functional unit is based on Single Instruction Multiple Data (SIMD) and executes linear and non-linear vector operations. ML workloads typically have high data parallelism, and execute wide vector operations, which motivates having wide vector instructions. On the other hand, hardware considerations motivate having narrow vector width to avoid offsetting the area efficiency of crossbars. Therefore, a time-multiplexed narrow SIMD unit balances the tradeoff between workloads favoring wide vector width and hardware favoring narrow vector width. The choice of SIMD width provides a trade-off between vector instruction efficiency and MVM efficiency. The MVM unit executes dot-product operations in the analog domain in crossbars. In addition to the crossbars, an MVM unit consists of DAC and ADC to interface with the digital data since, all other computations and communication occur in the digital domain. Typical MVM units use bit-slicing to represent high precision data (input and weights) required in ML [7]. Here, input bits are streamed using low-resolution DACs and weight bits are partitioned across low-precision NVM devices. DAC reuse across crossbars that share the inputs and ADC reuse across crossbar columns reduce the area overhead from peripherals to enable an efficient MVM unit design. The choice of crossbar size, DAC resolution, NVM-device's precision and ADC resolution provide control knobs to vary the energy-efficiency and area-efficiency of MVMU.

Figure 3(b) illustrates a multi-core, which consists of several cores connected together with a shared memory. The shared memory enables data movement between cores both within a multi-core as well as between different multi-cores. Subsequently, the choice of shared memory size and optimizations to reduce the communication between cores, reduce the data movement cost. A pool of multi-cores are connected together using an on-chip network to form a node. Further, multiple nodes can be connected using suitable chip-to-chip interconnect.

*4.1.2* ***Challenges:*** Despite their energy-benefits, further improvements in ReRam-based accelerators are impeded by costly ADCs required in MVMUs. Typical MVMUs used in ML accelerators use 8-10 bit ADCs to guarantee correct dot-product operation with no degradation from quantization during the analog to digital conversion. Such high resolution ADCs consume significantly higher energy and area compared to the resistive crossbar, thereby dominating the energy and area distribution of an MVMU [7, 8]. Recent research has explored micro-architecture optimization for reducing the ADC overhead by dynamically scaling the ADC precision requirements [26]. Here, the lower bits of input and weight data use lower ADC precision, whereas the higher bits of input and weight data use high ADC precision for their crossbar operations. Further research on performance of MVM operations with reduced ADC precision in error-resilient layers of the ML model, domain-specific ADC designs for ML, and ADC designs at scaled technology nodes will be instrumental towards improving ReRam-based accelerators.

## 4.2 Training accelerator

Training workloads pose additional requirements for designing crossbar-based systems namely 1) larger shared memory, and 2) frequent crossbar writes. During inference, the computed activations can be discarded once they are fed to the following layers. However, during training, these computed activations need to be stored, as they are also required during the weight update operations for computing the gradients. Therefore, a system for DNN training has much higher storage requirements for non-stationary data (or shared memory) than a system for DNN inference. Thus, the on-chip shared memory of crossbar-based inference accelerators is often insufficient, and these accelerators have to be accompanied with high bandwidth and high capacity external memory to enable a training system that is capable of handling large-scale DNNs. Consequently, a training system also requires the traditional memory hierarchy (with multiple levels), to efficiently hide the main memory latency. Second, while crossbar writes are infrequent in inference (configuration-time only), the crossbar is written at the end of every step/batch during the training flow [27].

Recent research have proposed a device-circuit technique called parallel write where weight gradient computations and updates can be performed *in-situ* in resistive crossbars [28]. While such an in-situ operation can mitigate the requirement of expensive crossbar writes in ML training, they are limited to low-precision operations. To this effect, researchers have explored a bit-slicing technique to incorporate parallel write to enable high-precision weight updates required in ML [27]. Further research on enhancing the scalability of parallel write for large-sized crossbars can enable the adoption of ReRam-based training accelerators.

# 5 MODELING AND EVALUATION TOOLS

The next step towards a full-fledged system is the frameworks that enable mapping of application algorithms.

## 5.1 Modeling of NVM Crossbars

The errors at the output of each MVM unit can accumulate and propagate across the layers of large-scale DNNs to cause significant degradation in application accuracy. In order to address these errors, there is a need to model the non-ideal behavior of MVM crossbars. Broadly, this can be of two types: 1) Analytical Modeling, 2) Statistical Modeling.

*5.1.1* ***Analytical Modeling:*** Analytical modeling involves close-form equations to express the output current as a function of input

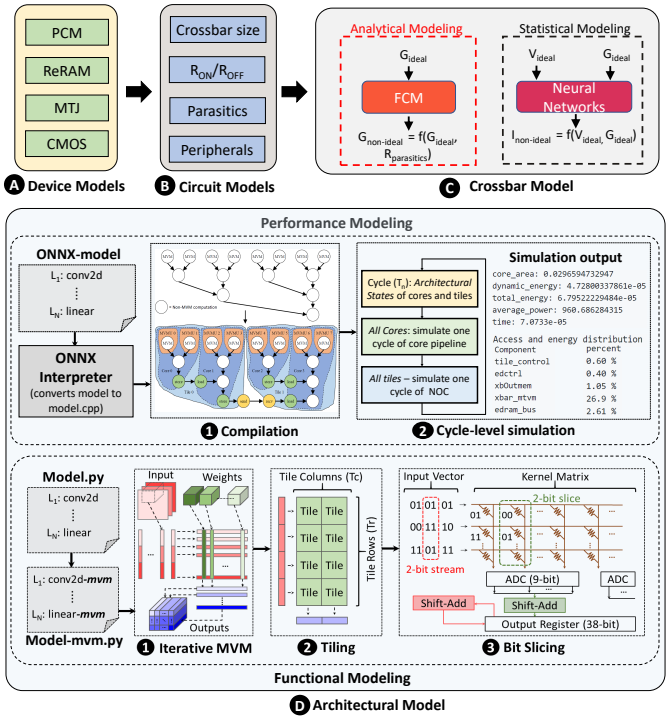


**Figure 4: Tool flow for Modelings across various levels of design stack**

voltages and conductances. This can either involve weight transformation techniques [9] to convert ideal weight matrices to non-ideal weight matrices or through lumped parastic models to form resistive divider equations [29].

*5.1.2* ***Statistical Modeling:*** While analytical models can capture impact of read non-idealities when input variables such as $V$ and $G$ are not co-dependent, there is a need to explore statistical models where input dependence of NVM devices cause inter-dependence of $V$ and $G$. One such model [10] leverages the generalizing property of neural networks to model the behavior of a non-ideal crossbar. Here, a representative dataset of various $V$ and $G$ combinations and corresponding non-ideal outputs is used to train a neural network. Such an approach can approximate SPICE results more closely compared to analytical models.

## 5.2 Architecture Modeling Tools

*5.2.1* ***Performance Simulator.*** The performance and energy analysis of large-scale DNNs that map across 100s of cores on a crossbar-based architecture requires a detailed architecture model. Such an architecture model is instrumental for obtaining measurements (energy and performance), as well as bottleneck analysis. Past work has developed cycle-level performance simulator which runs applications compiled to a domain-specific ISA and provides detailed traces of execution [8]. The simulator incorporates timing, and power models of all system components. The digital components were implemented in Verilog and the corresponding area, power, and timing were incorporated in the cycle-level simulator. The estimates for analog components (crossbar and peripherals) were obtained from circuits level simulations (Section 3). Subsequently, the access counts of individual components and the total cycles is used to estimate the dynamic and static energy components.

*5.2.2* ***Functional Simulator.*** **Inference:** In order to map large-scale DNNs on memristive crossbars requires integration of crossbar models with existing ML frameworks. There have been initial works on mapping non-ideal weights in ML frameworks [9], however, going forward its necessary to include architectural aspects such as *Tiling* and *Bit−Slicing*, described in Section. 4. Further, the iterative striding and convolution operation is required to model a DNN layer at the granularity of a MVM crossbar. In image classification tasks, the matrix in an MVM operation is formed by weights and vector is formed by a input pixel block. The MVM operation on this data produces a single output pixel. Next, the weight matrix is divided into crossbar sized chunks, called tiles. Finally, the bit-slicing of weights and bit-streaming of inputs mimic the bit-serial computing to represent high precision operations using NVM devices. The outputs from each tile goes through an ADC and ADC outputs from series of weight slices and input streams are shifted and added to represent the output. Such a detailed framework is necessary to accurately evaluate large-scale DNNs on memristive crossbar architectures. We observed that non-idealities reduce the accuracy by ∼4% and ∼13% for ImageNet and CIFAR-100, respectively for 16-bit fixed point representation.

**Training:** DNN training frameworks have also been explored by modifying ML platforms to incorporate the write characteristics of NVM devices. Most training frameworks based on NVM crossbars study the impact of device [30] and circuit [31] characteristics on on-device learning for ML workloads. These studies show that non-linear and asymmetric conductance updates as well as circuit parasitics can severely affect the training operations.

## 5.3 Mitigation of Functional Errors:

Further, techniques should be explored to compensate for the functional errors to preserve the software level application accuracy.

*5.3.1 Inference.* Mitigation algorithms for inference can be broadly categorized into: i) re-training of weights by non-ideal loss minimization [9, 32], ii) mapping algorithms [33–36] and iii) hardware compensation [9]. First, re-training of weights can be performed through iterative gradient descent on modified loss values obtained by using the non-ideal crossbar models in the forward pass of the network. Re-training achieve an accuracy improvement of 8%-27% for various DNN models with just 150 re-training iterations [9]. Further modification of gradient descent algorithm to account for the non-idealities in the backward pass has also been explored [32].

Alternatively, mapping algorithms have been explored to account for computational errors. Variation and defect aware mapping techniques [33, 35, 37] involve mapping weights to cells with less variations. Other techniques are based on linear algebraic techniques to reduce crossbar size [34] and mapping techniques to place sensitive weights near the voltage source to reduce parasitic impact.

Finally, techniques have been explored with regard to a compensation hardware which divides the output current by a non-ideal compensation factor [9] to obtain the ideal current. Such a technique can bring down degradation in accuracy to 1.8%-2.5%.

*5.3.2 Training.* We have mentioned above re-training DNNs with crossbars models is one of the most powerful techniques of addressing functional errors during inference. However, on-device training requires algorithmic techniques to circumvent challenges of limited ON/OFF ratio and dynamic range of input/output. Researchers have explored activation normalization techniques to address these issues [38]. On the other hand, noise management techniques are required to address the problem of differences in forward and backward pass inputs during training.

## 6 OUTLOOK

In this paper, we provided an overview of in-memory computing systems based on NVM technologies for ML workloads. We highlighted how the storage density of NVM crossbars allows large-scale DNNs to be entirely mapped onto a spatial architecture, thus reducing costly DRAM access. As a result, systems with NVM crossbars can potentially achieve significantly lower energy consumption and latency compared to state-of-art CMOS digital systems. We have also focused on several challenges at various levels of the design stack that need to addressed to truly realize the potential of in-memory computing systems based on NVM crossbars.

There are further challenges worth mentioning. First, limitations of NVM devices such as endurance, data retention challenges due to resistance drifting effects can have detrimental effects in NVM crossbars for ML workloads. Further, the temperature dependence of the NVM devices can introduce undesirable functional changes. Finally, device variability in both NVM devices and selectors is a major challenge that can produce erroneous MVM outputs. These inherent technological challenges needs careful primitive design to minimize their impact on large-scale DNNs.

We have also highlighted how ADCs are a major contributor to energy consumption in NVM crossbars. One avenue of research would be exploiting the inherent sparsity in the inputs of weights of DNNs to reduce the precision requirement of ADCs. Further, we can explore low SNR ADCs since the MVM computations are inherently approximate in nature. Overall, in-memory computing with emerging technologies is at its dawn today, with several challenges looming high in the path forward. With intelligent device research, careful primitive design as well as effective hardware/software co-design, the true power of this technology can be harnessed.

## ACKNOWLEDGEMENTS

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

## REFERENCES

[1] David Silver et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.
[2] Avishek Biswas and Anantha P Chandrakasan. Conv-ram: An energy-efficient sram with embedded convolution computation for low-power cnn-based machine learning applications. In *2018 IEEE ISSCC*, pages 488–490. IEEE, 2018.
[3] Akhilesh Jaiswal et al. 8t sram cell as a multibit dot-product engine for beyond von neumann computing. *IEEE TVLSI*, 2019.
[4] H-S P Wong et al. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.
[5] H-S P Wong et al. Metal–oxide rram. *Proceedings of the IEEE*, 100(6):1951–1970, 2012.
[6] Xuanyao Fong et al. Spin-transfer torque devices for logic and memory: Prospects and perspectives. *IEEE TCAD*, 35(1):1–22, 2015.
[7] Ali Shafiee et al. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News*, 44(3):14–26, 2016.
[8] Aayush Ankit et al. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Twenty-Fourth ASPLOS*, pages 715–731. ACM, 2019.
[9] Shubham Jain and Anand Raghunathan. Cxdnn: Hardware-software compensation methods for deep neural networks on resistive crossbar systems. *ACM TECS*, 18(6):113, 2019.
[10] Indranil Chakraborty et al. Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks. *arXiv preprint arXiv:2003.06902*, 2020.
[11] Xiaoyu Sun and Shimeng Yu. Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks. *IEEE JETCAS*, 9(3):570–579, 2019.
[12] Hendrik F Hamann et al. Ultra-high-density phase-change storage and memory. *Nature materials*, 5(5):383, 2006.
[13] S Kim et al. Nvm neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. In *2015 IEEE IEDM*, pages 17–1. IEEE, 2015.
[14] Sheng-Yu Wang et al. Multilevel resistive switching in ti/cu x o/pt memory devices. *Journal of Applied Physics*, 108(11):114110, 2010.
[15] Can Li et al. Analogue signal and image processing with large memristor crossbars. *Nature Electronics*, 1(1):52, 2018.
[16] John C Slonczewski. Current-driven excitation of magnetic multilayers. *Journal of Magnetism and Magnetic Materials*, 159(1-2):L1–L7, 1996.
[17] M. Bavandpour et al. Mixed-signal neuromorphic inference accelerators: Recent results and future prospects. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 20.4.1–20.4.4, Dec 2018.
[18] B. Razavi. A tale of two adcs: Pipelined versus sar. *IEEE Solid-State Circuits Magazine*, 7(3):38–46, Summer 2015.
[19] B. Razavi. The flash adc [a circuit for all seasons]. *IEEE Solid-State Circuits Magazine*, 9(3):9–13, 2017.
[20] Peng Yao et al. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577:641 – 646, 2020.
[21] X. Sun et al. Xnor-rram: A scalable and parallel resistive synaptic architecture for binary neural networks. In *2018 DATE*, pages 1423–1428, March 2018.
[22] Ximeng Guan et al. A spice compact model of metal oxide resistive switching memory with variations. *IEEE electron device letters*, 33(10):1405–1407, 2012.
[23] Pai-Yu Chen et al. Mitigating effects of non-ideal synaptic device characteristics for on-chip learning. In *IEEE/ACM ICCAD*, pages 194–199. IEEE Press, 2015.
[24] Shimeng Yu et al. Read/write schemes analysis for novel complementary resistive switches in passive crossbar memory arrays. *Nanotechnology*, 21(46):465202, 2010.
[25] Ping Chi et al. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. In *Proceedings of the 43rd International Symposium on Computer Architecture*, pages 27–39. IEEE Press, 2016.
[26] Anirban Nag et al. Newton: Gravitating towards the physical limits of crossbar acceleration. *IEEE Micro*, 38(5):41–49, 2018.
[27] Aayush Ankit et al. Panther: A programmable architecture for neural network training harnessing energy-efficient reram. *arXiv preprint arXiv:1912.11516*, 2019.
[28] Sapan Agarwal et al. Resistive memory device requirements for a neural algorithm accelerator. In *2016 IJCNN*, pages 929–938. IEEE, 2016.
[29] YeonJoo Jeong et al. Parasitic effect analysis in memristor-array-based neuromorphic systems. *IEEE TNANO*, 17(1):184–193, 2017.
[30] P. Chen et al. Neurosim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning. *IEEE TCAD*, 37(12):3067–3080, Dec 2018.
[31] Sourjya Roy et al. Txsim: Modeling training of deep neural networks on resistive crossbar systems. *arXiv preprint arXiv:2002.11151*, 2020.
[32] Indranil Chakraborty et al. Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars. *IEEE TETCI*, 2(5):335–344, 2018.
[33] Chenchen Liu et al. Rescuing memristor-based neuromorphic design with high defects. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2017.
[34] Beiye Liu et al. Reduction and ir-drop compensations techniques for reliable neuromorphic computing systems. In *IEEE/ACM ICCAD*, pages 63–70. IEEE, 2014.
[35] Beiye Liu et al. Vortex: variation-aware training for memristor x-bar. In *Proceedings of the 52nd Annual Design Automation Conference*, page 15. ACM, 2015.
[36] Amogh Agrawal et al. X-changr: Changing memristive crossbar mapping for mitigating line-resistance induced accuracy degradation in deep neural networks. *arXiv preprint arXiv:1907.00285*, 2019.
[37] Lerong Chen et al. Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar. In *DATE '2017*, pages 19–24. IEEE, 2017.
[38] Malte J Rasch et al. Training large-scale anns on simulated resistive crossbar arrays. *arXiv preprint arXiv:1906.02698*, 2019.