Chapter

Mitigating State-Drift in Memristor Crossbar Arrays for Vector Matrix Multiplication

Amirali Amirsoleimani, Tony Liu, Fabien Alibart, Serge Eccofey, Yao-Feng Chang, Dominique Drouin and Roman Genov

Abstract

In this Chapter, we review the recent progress on resistance drift mitigation techniques for resistive switching memory devices (specifically memristors) and its impact on the accuracy in deep neural network applications. In the first section of the chapter, we investigate the importance of soft errors and their detrimental impact on memristor-based vector–matrix multiplication (VMM) platforms performance specially the memristance state-drift induced by long-term recurring inference operations with sub-threshold stress voltage. Also, we briefly review some currently developed state-drift mitigation methods. In the next section of the chapter, we will discuss an adaptive inference technique with low hardware overhead to mitigate the memristance drift in memristive VMM platform by using optimization techniques to adjust the inference voltage characteristic associated with different network layers. Also, we present simulation results and performance improvements achieved by applying the proposed inference technique by considering non-idealities for various deep network applications on memristor crossbar arrays. This chapter suggests that a simple low overhead inference technique can revive the functionality, enhance the performance of memristor-based VMM arrays and significantly increases their lifetime which can be a very important factor toward making this technology as a main stream player in future in-memory computing platforms.

Keywords: Memristor Crossbar, State-drift, Vector–matrix Multiplication, Inference

1. Introduction

Designing specialized hardware accelerators has been a topic of interest recently due to rapid growth of machine learning and artificial intelligence [1–5]. Despite the recent advancements in developing machine learning complementary-metal-oxide (CMOS) based chips to efficiently implement vector–matrix multiplication (VMM) operations, these systems are limited by the off-chip memory bottleneck [6]. To this end, co-locating memory and processing has been considered as a solution by using non-volatile resistive switching memory technologies [7–14]. One such device technology is memristor and it has risen as a promising high speed, low power computational alternative to traditional CMOS hardware [15–18]. Memristor can be placed in a crossbar structure to perform highly parallel multiply-accumulate (MAC) operations efficiently using Ohm's Law [19–22]. Through heavy parallelization using Kirchoff's laws, a memristor crossbar is able to do MAC operations at $O(1)$ speed [23]. Crossbars are most commonly used to perform VMM by mapping a n by m matrix the memristors' conductance range, applying an input voltage vector to the rows, and then reading the current from the appropriate columns [24]. In addition to neuromorphic applications [25–28] of memristor crossbar, through VMM memristor crossbar has shown to be capable of performing a number of tasks ranging from image processing [29], physical unclonable functions (PUFs) [30–32], optimization problems [33–35], sparse coding [36], and solving partial differential equations [37]. Also, there have been many researches focusing on implementation of deep neural network (DNN) accelerators using memristor crossbars which focuses on different device, algorithm and system level contributions [18–20, 38]. While there has been significant progress in memristor crossbar-based computational devices, there are still many major challenges in robustness and computational accuracy that hinder the technology [20, 39, 40]. There have been several researches focus on mitigating the impact of non-idealities on memristor crossbar systems performance and these reliability improvement techniques are mainly proposed for process variations [41, 42], hard faults [43, 44], signal distortion issue [45], and memristance drift [46–48]. These techniques can be mainly categorized into (i) retraining to compensate the error (ii) mapping techniques (iii) closed-loop training (iv) error-correction coding.

Here, we focus on the memristance drift effect and the mitigation techniques to avoid the impact of this issue over the memrisitive DNN systems. Typically, in memristor crossbars, there are two major operations to perform: write and read operations. In the write operation, a voltage above the switching voltage of the memristor [49] is applied to a memristor repeatedly until the resistance of the memristor is sufficiently close to the target resistance. During the read operation, a voltage lower than the switching voltage is applied to the memristor and the current from the memristor is measured. The read operation is used extensively in the inference operation of many Ex-situ and In-situ algorithms including artificial neural networks (ANN). Ideally, the resistance of the memristor should not change at all, but in practice, there is often a very small change in the memristor state after a read operation. This phenomenon is known as memristance drift [50, 51]. Over many read operations, these small changes in resistance of the memristors in a crossbar will add up to have a significant impact on computational accuracy. Memristance drift occurs in different resistive switching memory technologies and it is not similar in terms of the behavior. In phase change memory (PCM) devices, the memristance drift occurs even when there is no voltage applied over the memory cell and the amorphous state (high resistance state) of the device is changing over time [52]. Subsequently, this issue will be more severe a the high-resistive amorphous state increases and this will impact dramatically the PCM-based system's performance in presence of high cycle-to-cycle and device-to-device variations. In memristor technology, as discussed before, the repetitive VMM operations result a memristance drift phenomenon and it becomes worse as the number of inference operations increases. Existing solutions to this problem include periodical weight reprogramming and feedback designs have limitations with high computational overhead and limited long-term effectiveness. For instance, HfOx RRAM testing results using a dynamic BL-bias circuit for preventing memristor state disturbance during read operations [53]. Error correction code (ECC) is used in [51] to reduce write latency by up to 70%. However, these techniques are not sufficient in themselves to enhances the performance of computational memristor crossbars in which 2–3 bits of memristance drift can cause significant decreases in performance. Recently, a few more effective memristance-drift mitigation techniques have been proposed [54–56]. This chapter will summarize the approach and results of a closed-loop feedback system technique [54] and an inline calibration approach [55] before taking a more in-depth look into an adaptive inference technique (AIDX) [56] that optimizes the inference voltage pulse amplitude and width. In addition, the power and chip area overhead of these three techniques are briefly compared.

2. Memristance drift and its modeling

In general, there are memristor models can be separated into physics-based and behavior-based simulations depending on the characteristics of their modeling and their general purpose. Physics-based models typically attempt to simulate memristors at a molecular-level by considering the material characteristics of the active memristor layers and mathematical modeling of the ion drifting between these materials. While physical models accurately model memristance drift, they are generally computationally expensive and limited in scope to the detailed analysis of singular memristor behavior. As such, memristor crossbar arrays are modeled using behavior-based models. Behavior-based memristor models are much simpler than physics-based models and use experimental fitting parameters to match the behavior of different types of memristors. Current–voltage plots are one of the most common methods of quickly visualizing memristor short-term behavior and many behavior-based models like VTEAM [57] are built to agree with these plots. Over the course of a voltage sweep, there is not enough time for the memristor's state to change noticeably under the threshold voltage and as such, the long term consequences of memristance drift aren't captured in these models. Many popular behavior-based models, such as VTEAM [57] and TEAM [58], utilize current and voltage thresholds to partition memristor behavior under high and low voltage/current scenarios. Generally, these threshold models approximate the subthreshold state change as zero and thus do not consider the long-term effects of memristance drift. Other behavior-models, such as the nonlinear ion drift [59] and Simmons Tunnel Barrier model [60], do not utilize a threshold and instead model memristor high and low voltage memristor behavior using the same equations and fitting parameters. Without a threshold, these models lack the flexibility to accurately model the minute changes of memristance drift without sacrificing the accuracy of its higher-voltage switching modeling. Recently, there have been attempts to extend popular behavior-based models to more accurately simulate memristance drift. For instance, [56] added subthreshold modeling equation and fitting parameters to extend the VTEAM model. However, the modeling of memristance drift in behavior-based models is still currently in its infancy due to the lack of experimental data on long-term memristor behavior when exposed to low voltage pulses.

2.1 Impact of state-drift on crossbar VMM

Memristance drift in crossbar arrays can be summarized as the buildup of small unintended changes in memristor state over many low-voltage read operations [56]. During a crossbar VMM operation, the ideal output current I_j of the j-th column can be modeled simply as:

$$I_j = \sum_i G_{ij} V_i \tag{1}$$

Here V_i represents the voltage applied to the i-th row of the crossbar and G_{ij} is the conductance of the memristors in the i-th row and j-th column. For a given VMM operation, state drift can be represented as a small change in conductance δG. The non-ideal output current I'_j is then given as:

$$I'_j = \sum_i (G_{ij} + \Delta G_{ij}) V_i \tag{2}$$

ΔG_{ij} represents the accumulation of memristance drift caused by all previous VMM operations and skews the distribution of weights within the crossbar (**Figure 1a**). It can be shown from Eqs. (1) and (2) that the difference between I'_j and I_j scales with the read voltage of each row V and the number of rows N. Over a long period, the buildup of drift in the crossbar weights caused by memristance drift will lead to significant error (**Figure 1b**). The speed of memristance drift will vary depending on application and memristor characteristics. Even after a short period of 1 second, a 0.1 V signal could cause a memristor to deviate around 2% from its initial state [61]. A thorough analysis of memristance drift speed with respect to initial state and drift direction is done in [54]. In the SET direction, [54] found that memristors' resistance decreased by 77.07%, 62.07%, 56.28% and 8.81% after 100 read operations with initial resistances of 200kΩ, 100kΩ, 80kΩ, and 15kΩ, respectively. The speed of drift in the RESET direction was much slower at only ~0%, 1.17 × 10–4%, 0.018% and 16.43% increase in resistance for the same initial states. From the analysis in [54], memristance drift speed is shown to be greatly impacted by initial state and the direction of switching. A heatmap of conductance highlights the impact of initial state on the buildup of memristance drift over time

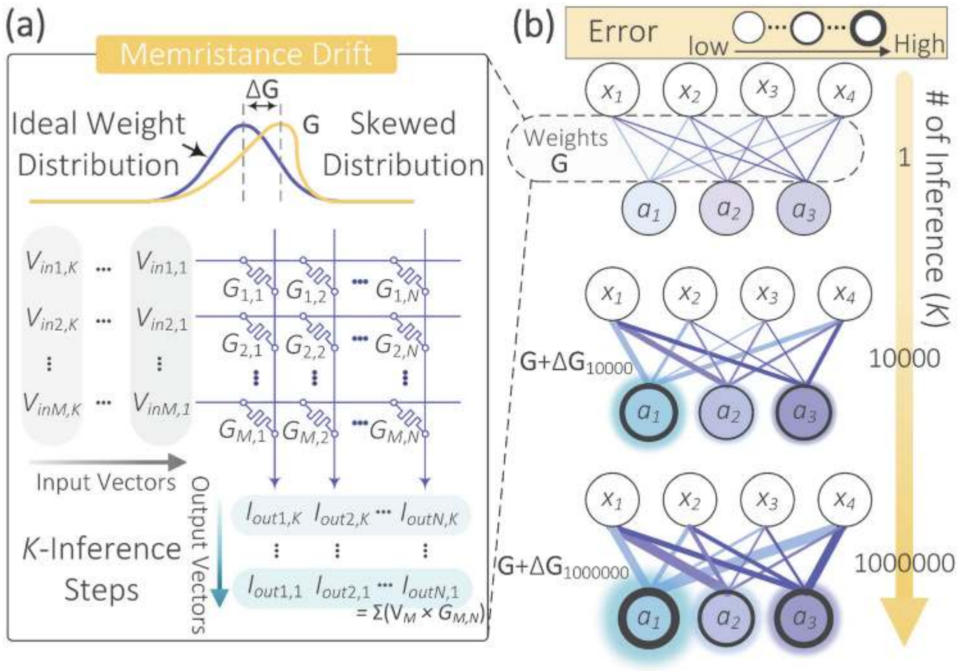
(a)
Memristance Drift
ΔG
Ideal Weight Distribution
G
Skewed Distribution
$V_{in1,K}$ ... $V_{in1,1}$
$V_{in2,K}$ ... $V_{in2,1}$
$V_{inM,K}$ ... $V_{inM,1}$
$G_{1,1}$ $G_{1,2}$ ... $G_{1,N}$
$G_{2,1}$ $G_{2,2}$ ... $G_{2,N}$
$G_{M,1}$ $G_{M,2}$ ... $G_{M,N}$
Input Vectors
Output Vectors
K-Inference Steps
$I_{out1,K}$ $I_{out2,K}$ ... $I_{outN,K}$
$I_{out1,1}$ $I_{out2,1}$ ... $I_{outN,1}$
$= \Sigma(V_M \times G_{M,N})$
(b)
Error low High
# of Inference (K)
x_1 x_2 x_3 x_4
Weights G
a_1 a_2 a_3
1
$G+\Delta G_{10000}$
10000
$G+\Delta G_{1000000}$
1000000

Figure 1.
(a) Neural network (NN) weight matrix mapped onto crossbar with conductance matrix G. Each subsequent VMM operation will cause a slight skew in the distribution of the memristors' conductance. (b) Initially, the error due to memristance drift is negligible. Over many inference operations, the skew in the conductance distribution builds up resulting in significant error in the NN output. Figure reprinted by [56].

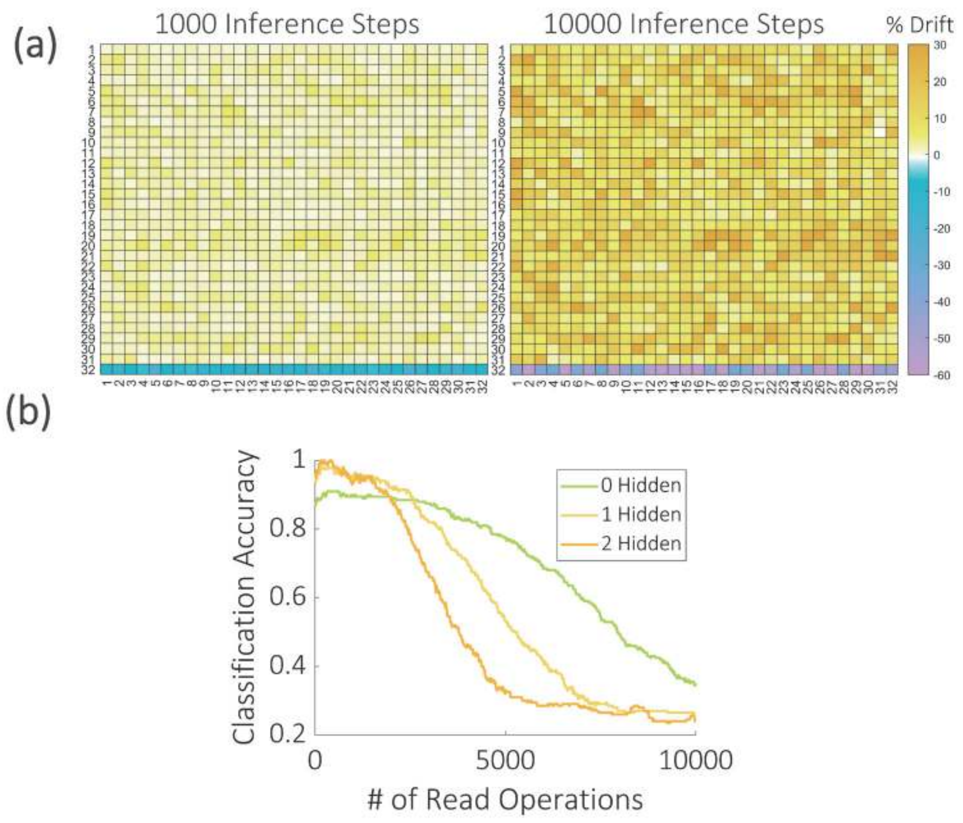

Figure 2.
(a) Impact of varying number of hidden layers on memristance drift induced accuracy degradation on the MNIST dataset [62]. (b) Heatmap of typical memristor crossbar weight mapping with the bottom row as the bias and the rest of the heatmap representing the weight matrix. Over time, the memristance drift direction between the bias and weights diverges due to differences in bias and weight initialization. Figure reprinted by [56].

(**Figure 2a**). Here, the bottom row represents the bias of a neural network and are initialized near the high conductance state while the rest of the memristors representing the weights are initialized close to the low conductance state. Due to this conductance initialization, all memristors except the bottom row in the above figure experience an aggregate memristance drift in the positive direction while the bottom row experiences memristance drift in the negative direction. There have been multiple studies that show that memristance drift causes significant performance degradation on various applications after long term use. In [56], the negative impact across ten baseline ML tasks in the Proben1 [63] datasets, memristance drift caused an average classification accuracy decrease of 26%, 42%, and 51% after 500, 2000, and 10000 inference operations, respectively. [56] also analyzed the effects of memristance drift on convolutional neural networks with the CIFAR10 image classification dataset [64]. Ten different CNN architectures were tested with a relatively consistent accuracy degradation of 29%, 59%, and 72% after 500, 2000, and 10000 inference operations respectively. Memristor drift ranges from upwards of 10% deviatation from its programmed value to upwards of 30% deviation at 10000 inference steps as shown in (**Figure 2a**). To clarify, the memristance drift speed remains the same for each network in (**Figure 2b**) regardless of the number of hidden layers. However, each additional hidden layer accumulates memristance increasing amounts of memristance drift-related error from the previous layer causing deeper neural network's accuracy to degrade more quickly than networks with less hidden layers (**Figure 2b**). In [55], the classification accuracy on the

MNIST handwritten digits dataset [62] decreases by approximately 2.5–4% across four independent trials due to the cycle-to-cycle variation of memristance drift. In [54], the classification accuracy degradation on MNIST was tested with memristance drift in the SET and RESET directions separately. In the SET direction, the classification accuracy dropped from 91.91–60% after only 100 inference operations. In the RESET direction, accuracy degradation was slower where accuracy dropped to 60% after approximately 300 inference operations.

3. Memristance drift mitigation overview

3.1 ICE: inline calibration

Given the significant negative impact of memristance drift on crossbar performance, there has been a few works that have proposed meaningful solutions to the memristance drift problem. When a memristor crossbar's performance drops below acceptable levels due to memristance drift, the memristors will be rewritten to their intended states. For a given application, this recalibration is usually done at periodically ensure high crossbar accuracy over long time intervals. Since rewriting a memristor to a specific state can be up to 100 times slower than the speed of an inference operation [65], frequent crossbar calibrations could significantly bottleneck crossbar throughput. In [55], the authors propose an inline calibration method that utilizes "interrupt-and-benchmark (I&B)" operations to track the crossbar computational error in order to predict the time period before the next calibration operation. In addition, the time period between two I&B operations is dynamically optimized as to minimize the time overhead of the inline calibration method on crossbar performance. As defined in the paper, I&B operations interrupt regular crossbar operation in order to evaluate crossbar computational error on a set of benchmark data [55]. Between each recalibration of the memristor crossbar, there exists a theoretical maximum number of inference operations $sup\ n_r$ before the next recalibration must be done due to performance degradation. The optimization goal of ICE is to make the actual number of inference operations n_r between two calibration operations approach $sup\ n_r$. This is not a trivial task because $sup\ n_r$ can vary significantly between many calibration operations due to changes in memristance drift speed from cycle-to-cycle variations and other factors. ICE approximates $sup\ n_r$ by applying polynomial fitting to the crossbar error data during I&B operations.

To reduce the time overhead of the inline calibration method, ICE seeks to minimize the number of I&B operations k between each recalibration of the crossbar. ICE uses its polynomial fitting function to guess the benchmark computation error of the next I&B operation. If the absolute difference between the guessed crossbar error and the actual I&B error is below some threshold, the time until the next I&B operation will be doubled up to some maximum time interval. Otherwise, ICE will reset the time interval between successive I&B operations to its default value. For testing, ICE adopts a TiOx-based memristor device model from [61]. This model extends the bulk model TiO2 with a focus on behavior-based process variation analysis of memristors. The results presented in [55] are measured in terms of efficiency with computational efficiency defined as:

$$\gamma = \frac{n_r}{sup\ n_r} \tag{3}$$

To quantify the time overhead caused by I&B operations, the parameter Δ is defined as:

$$\Delta = \frac{kn_{IB}}{n_r} \tag{4}$$

Here, k is the average number of I&B operations between crossbar recalibrations and n_{IB} is the number of inference operations required per I&B operation. These criteria were evaluated across four baseline tasks (HMAX, KMeans, Sobel, and MNIST [62]) and compared to a baseline of constant time period crossbar recalibrations. With second degree polynomial fitting, ICE achieves an average calibration efficiency γ of 91.18% which is a 21.77% improvement over the baseline. This improvement in calibration efficiency only came at the cost of a time overhead Δ of 0.439%. No information on the voltage range or power consumption of ICE is presented in [55]. However, the author's mentioned that future works could include a detailed power analysis of ICE using a SPICE-based memristor model.

3.2 Closed-loop feedback circuit

Traditionally, the data flow of a memristor crossbar-based neural network follows a linear pipeline in a conventional open-loop system. While these open-loop systems serve as a simple and efficient pipeline for VMM operations, they are not able to effectively manage the effects of memristance drift. In [54], a closed-loop circuit is proposed to mitigate memristance drift by adaptively adjusting the direction of current in each memristor using a feedback controller. Mean square error (MSE) is used to measure the degradation caused by memristance drift. Specifically, the difference in MSE (ΔMSE) between the ideal crossbar and the current state is used as a metric to inform the feedback controller. For each inference operation, a weight compensation algorithm is run to minimize memristance drift speed. The second feature of the closed-loop design introduced in [54] is the usage of an "arrogant principle" which assumes that the prediction made by the crossbar system is always correct. This principle allows the system to use its output as the label to determine the direction of compensation in the weight compensation algorithm. The effectiveness of this assumption hinges on the ideal accuracy of the crossbar-mapped neural network. With a high initial accuracy, the "expectation of recognition accuracy probability with respect to time" will be close to its upper bound, thus keeping the rate of degradation of the feedback controller low. Naively, this closed-loop design requires an additional compensation pulse for each inference operation which would halve the throughput of the crossbar system. To address this issue, [54] combines the compensation pulse with regular inference operations by manipulating the k-th inference operation into compensating for the memristance drift caused by the (k-1)-th inference. The feedback controller is used to determine the recall direction of the (k-1)-th and then adjusts the direction of the k-th inference pulse accordingly. By integrating the k-th compensation pulse into the (k + 1)-th inference pulse, there is no need to sacrifice crossbar throughput to implement this proposed system. For testing, [54] adopts the dynamic model of the TaOx memristor from [66] with a low resistive state and high resistive state of 1 kΩ and 1MΩ, respectively. The performance of single and two-layer neural networks were tested on the MNIST dataset. The effectiveness of the closed loop design was measured as the number of inference operations before the crossbar accuracy drops below 70%. As compared to a baseline crossbar system, the proposed closed-loop design is shown to increase the number of recall operations by 1897 operations for the single-layer network and 1590 operations for the two-layer network. This increase corresponds to a 13.84× lifetime extension for the single layer network and a 13.95× lifetime extension for the two-layer network. For power estimation, [54] uses a square recall voltage pulse of 0.3 V for 100 ns. Boban had an estimated power consumption of 1.1196 mW for a single layer neural network which increases to 6.7367 mW for a two-layer neural network implementation.

4. Adaptive inference scheme for memristance drift mitigation

In [56], the authors proposed an adaptive inference scheme called AIDX that optimizes the amplitude and duration of inference voltage pulses in order to minimize the speed of memristance drift. AIDX formulates memristance drift as an optimization problem and seeks to minimize the memristance drift error defined as the increase in mean squared error (MSE) from the initial programmed crossbar after a set number of inference operations. The initial MSE can be modeled as:

$$E_0 = \sum_j \left(y_j - \sum_i G_{ij} V_i \right)^2 \tag{5}$$

Similarly, the MSE after k inference operations is given as:

$$E_k = \sum_j \left(y_j - \sum_i \left(G_{ij} + \Delta G_{ij}^k \right) V_i \right)^2 \tag{6}$$

where ΔG_{ij}^k is the accumulated memristance drift in the memristor i-th row and j-th column from k inference operations. The additional error due to memristance drift after k operations is calculated as $E_{Drift} = E_k - E_0$. The naive approach of simply choosing the minimum allowable voltage amplitude and duration may seem logical because speed of memristance drift scales with amplitude and duration. However, this naïve approach would still result in significant memristance drift because of the vast differences in state drift speed in the SET and RESET drift [54]. As such, AIDX focuses on balancing the aggregate drift in the SET and RESET directions for a given application by optimizing the ratio of SET to RESET voltage pulse amplitude $\boldsymbol{A}$ and duration $\boldsymbol{D}$ (**Figure 3a**). The minimization of memristance drift with respect to voltage pulse amplitude and duration is formalized as follows:

$$\mathrm{min}_{\mathrm{A,D}} \mathrm{E}_{\mathrm{Drift}}(\mathbf{A}, \mathbf{D}) \tag{7}$$

Here, E_{Drift} is a function of **A** and **D** that are vectors which represent the ratio of SET to RESET voltage pulse amplitude and duration of each row of the memristor crossbar respectively. The Broyden-Fletcher-Goldfarb-Shannon (BFGS) algorithm [67] is used to tackle this optimization problem. Since the gradient of the memristance drift error cannot be evaluated directly, the gradients used in the BFGS algorithm were numerically approximated using function evaluations. Under some circumstances, it is possible for the optimized values of **A** and **D** to be too large or small to be properly implemented on a crossbar. Extreme values of **A** and **D** are often caused by skewed memristor characteristics where memristance drift speed in one direction is much faster than the other direction or when the data distribution is heavily skewed toward one recall direction. To address this issue, AIDX randomly inverting the direction of inference for an input vector **x** with probability a in order to compensate for imbalanced memristor characteristics and a skewed data distribution (**Figure 3b**). The probability of input inversion is optimized to minimize E_{Drift} before the optimization of **A** and **D** vectors to ensure a

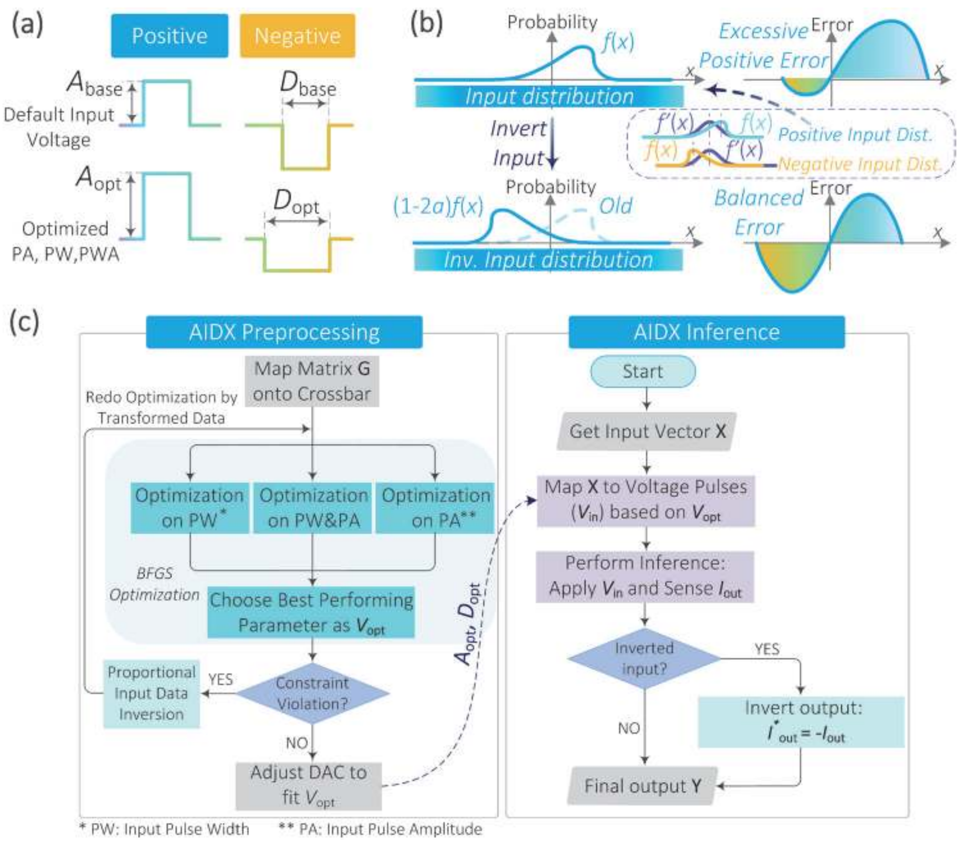

Figure 3.
(a) Visual representation on how AIDX's parameters changes the relative amplitude and width of positive and negative voltage pulses before and after optimization. (b) Illustration of skewed input distributions can cause an imbalance of memristance drift error in a particular direction. By applying input inversion, the input distribution is reflected such that the memristance drift error in each direction is balanced. (c) Flowchart describing the AIDX procedure for preprocessing and inference. Figure reprinted by [56].

relatively balanced memristance drift speed in the SET and RESET directions as to prevent extreme final values of **A** and **D.**

The general usage of AIDX in both preprocessing and inference is described in (**Figure 3c**). To ensure optimal performance, optimization is done in three scenarios: Optimizing over pulse amplitude ratio, optimizing over pulse duration ration, and optimization over both parameters simultaneously. The best set of parameters is then chosen according to lowest evaluated E_{Drift}. If the parameters **A** and **D** are too extreme, the optimization of probability of input inversion a is performed. Applying AIDX during inference is almost identical to a normal crossbar VMM operation except that if the input was inverted, the output vector must be reinverted to recover the intended output. When applying AIDX to deep neural networks, it can be inefficient to optimize over all layers simultaneously due to the large number of parameters. Instead, AIDX applies the BFGS algorithm to each layer separately in forward pass order in order to reduce optimization time (**Figure 4a**). AIDX used a simulated extended VTEAM model to fit real TiOx-based memristor device data. The following non-idealities were considered for testing: 15% memristor programming error, 15% random gaussian noise added to the high/low conductance states and alpha/k parameters of the extended VTEAM model for device-to-device variation. In addition, 200 ohms of source resistance and 20 ohms of line resistance were considered as well as sneak paths were also considered. AIDX was applied to a memristor system that

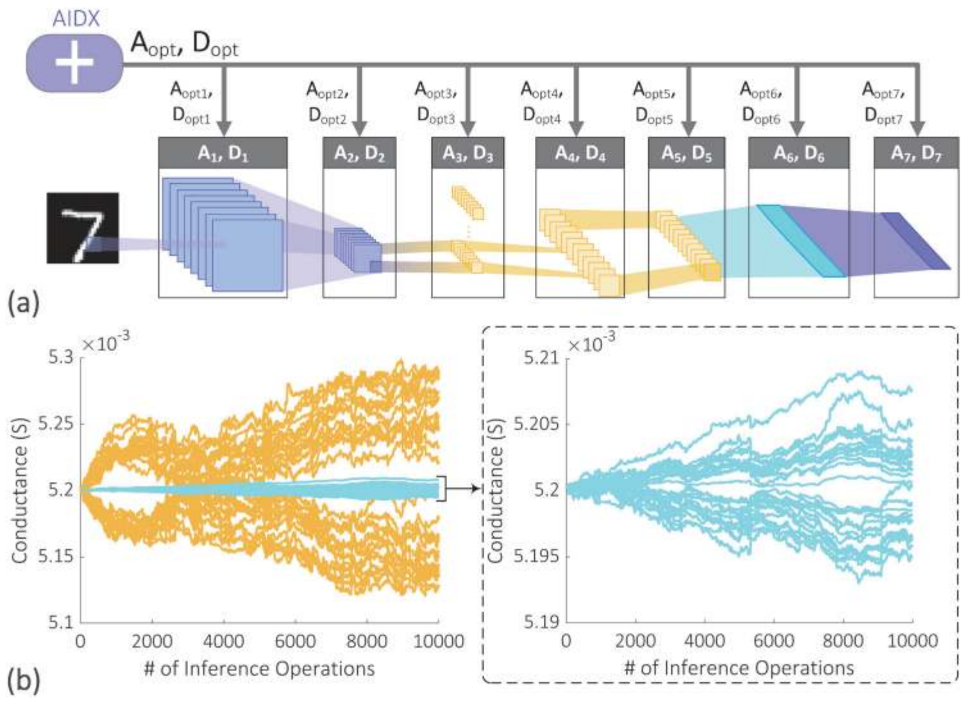

Figure 4.
(a) AIDX optimizes and applies separate voltage amplitude and duration ratios for each layer separately. (b) Basic test on memristance drift with all devices with and without AIDX. With all memristors initially set to 0.0052 S, half of the memristors receive pre-generated random sequences of positive and an identical sequence of negative pulses are applied to the other half of the crossbar.

operates with a voltage range of −0.2 V to 0.2 V. Depending on the pulse amplitude optimization, AIDX's inference pulse can vary within this range. On average, there is 2% reduction in passive power consumption within the crossbar compared to the baseline system when using AIDX. A simple test of AIDX effectiveness is done by applying random positive and negative voltage pulses to memristors with and without AIDX. After 10000 inference operations, the baseline memristors deviated a max of 1.9% from its initial value while AIDX had a max deviation of only 0.17% (**Figure 4b**). When tested on the 10 benchmark tasks from the Proben1 dataset [63], the average classification accuracy degradation with AIDX is approximately 4%, 7%,

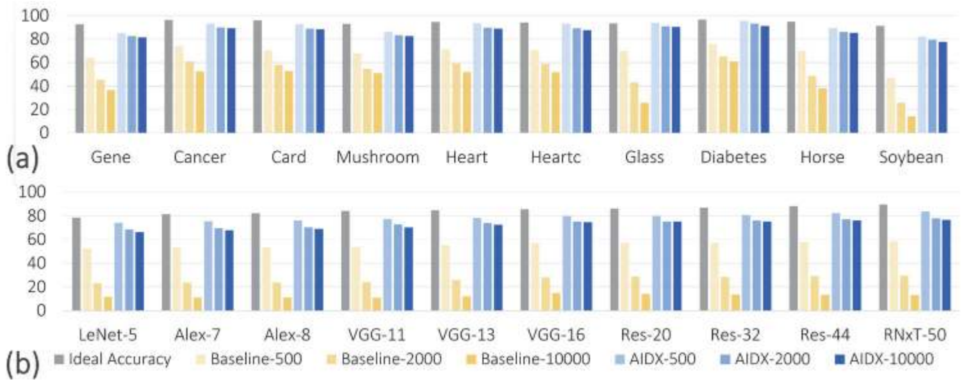

Figure 5.
(a) Classification accuracy of AIDX and baseline neural network on ten tasks from the Proben1 dataset after 500, 2000, and 10000 inference operations. (b) Classification accuracy of AIDX and baseline on CIFAR10 dataset with various CNN architectures after 500, 2000, and 10000 inference operations. Figure reprinted by [56].

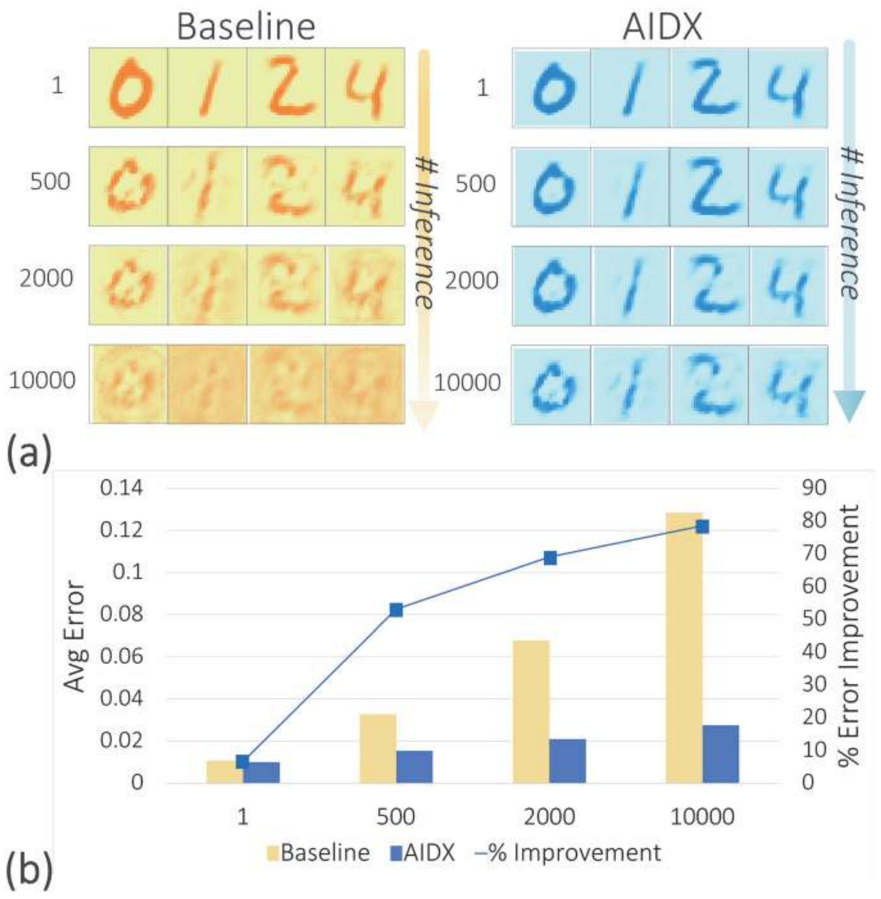

Figure 6.
(a) Reconstruction of sample images from MNIST dataset after 1, 500, 2000, and 10000 inference operations. (b) The average mean squared error in image reconstruction between AIDX and baseline autoencoder after set time steps. The percentage error improvement of AIDX over the baseline is also shown. Figure reprinted by [56].

and 8% after 500, 2000, and 10000 inference operations (**Figure 5a**). On average, AIDX reduced accuracy degradation by 42% as compared to the baseline test after 10000 inference operations. When testing AIDX on 10 CNN architectures using the CIFAR10 dataset [18], the classification accuracy decrease by an average of 4%, 7%, and 8% after 500, 2000, and 10000 inference operations (**Figure 5b**). This accuracy degradation corresponds to a 22%, 35%, and 43% improvement over the baseline respectively. In addition, AIDX was also applied to image reconstruction by training a simple 3-layer autoencoder on the MNIST dataset [62]. The average mean squared error of the baseline auto-encoder was 0.033, 0.068, and 0.129 after 500, 2000, and 10000 inference operations respectively. With AIDX, the average mean squared error drops to 0.015, 0.021, and 0.028 after 500, 2000, and 10000 inference operations which is an improvement of 53.0%, 69.0%, and 78.6% over the baseline (**Figure 6**).

5. Overhead analysis

The three methods for mitigating memristance drift discussed in this chapter all induce small overheads in terms of power consumption and chip area. Time overhead is not discussed in this section because there is negligible change in crossbar throughput by all three mitigation methods. Power overhead is defined in this section as the additional power consumption induced in the memristor crossbar and peripheral circuits due to proposed memristance drift solutions. For the sake of consistency, the estimates of peripheral power consumption of [54] are used for comparison. While power consumption is not disclosed in [55], the power overhead of [56] is 1.19% while [54] has a power overhead of 1.61%. Area overhead is defined consistently with [56] as the additional on-chip area required for memristance drift mitigation method because of peripherals, external circuit, and other items. Since both [55, 56] do not include any additional on-chip circuitry, these two methods do not have any chip area overhead while the closed loop circuits proposed in [54] require an additional 2.34% chip area. On the other hand, both [55, 56] require solving an optimization problem before implementing their mitigation technique. However, considering that the optimization procedure would only needed to be performed once for an application, these solutions still promise great scalability for long-term memristor crossbar usage.

6. Conclusions

In summary, this chapter first discusses memristor crossbar modeling and how there is a current lack of attention in modeling subthreshold memristor behavior. The next section overviews how the speed of memristance drift is impacted by recall voltage and amplitude, memristor characteristics, crossbar size, and number of inference operations since the last write operation. In addition, memristance drift is shown to cause severe accuracy degradation across multiple datasets and tasks such as MNIST and CIFAR10. The second half of this chapter is dedicated to overviewing three different approaches for memristance drift mitigation. First, an inline calibration approach [55] and a closed-loop feedback system is summarized. Then, there is a more in-depth look into an adaptive inference scheme that optimized the ratio of SET to RESET voltage pulse amplitude and width to minimize memristance drift speed. The final section of the chapter briefly compared the power and chip area overhead of these three memristance drift mitigation techniques. Hopefully, this chapter can bring more much-needed attention to the study of memristance drift and the development of drift mitigation techniques.

Author details

Amirali Amirsoleimani[1]*, Tony Liu[2], Fabien Alibart[3], Serge Eccofey[3], Yao-Feng Chang[4], Dominique Drouin[3] and Roman Genov[2]

1 Department of Electrical Engineering and Computer Science, Lassonde School of Engineering, York University, Toronto, Ontario, Canada

2 Intelligent Sensory Microsystem Laboratory, University of Toronto, Toronto, Ontario, Canada

3 Interdisciplinary Institute for Technological Innovation – 3IT, University of Sherbrooke, Sherbrooke, Quebec, Canada

4 Microelectronics Research Center, The University of Texas at Austin, Austin, TX, USA

*Address all correspondence to: amirsol@yorku.ca

IntechOpen

© 2021 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

References

[1] Jouppi NP, Young C, Patil N, Patterson D, Agrawal G, Bajwa R, Bates S, Bhatia S, Boden N, Borchers A, Boyle R. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th annual international symposium on computer architecture 2017 Jun 24 (pp. 1-12).

[2] Chung E, Fowers J, Ovtcharov K, Papamichael M, Caulfield A, Massengill T, Liu M, Lo D, Alkalay S, Haselman M, Abeydeera M. Serving dnns in real time at datacenter scale with project brainwave. IEEE Micro. 2018 Apr 20;38(2):8-20.

[3] Chen YH, Krishna T, Emer JS, Sze V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE journal of solid-state circuits. 2016 Nov 8;52(1):127-138.

[4] Lee J, Kim C, Kang S, Shin D, Kim S, Yoo HJ. UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In2018 IEEE International Solid-State Circuits Conference-(ISSCC) 2018 Feb 11 (pp. 218-220). IEEE.

[5] Moons B, Uytterhoeven R, Dehaene W, Verhelst M. 14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi. In2017 IEEE International Solid-State Circuits Conference (ISSCC) 2017 Feb 5 (pp. 246-247). IEEE.

[6] Wulf WA, McKee SA. Hitting the memory wall: Implications of the obvious. ACM SIGARCH computer architecture news. 1995 Mar 1;23(1): 20-24.

[7] Yu S. Neuro-inspired computing with emerging nonvolatile memorys. Proceedings of the IEEE. 2018 Feb;106 (2):260-285.

[8] Chakraborty I, Jaiswal A, Saha AK, Gupta SK, Roy K. Pathways to efficient neuromorphic computing with non-volatile memory technologies. Applied Physics Reviews. 2020 Jun 3;7(2):021308.

[9] Ambrogio S, Narayanan P, Tsai H, Shelby RM, Boybat I, Di Nolfo C, Sidler S, Giordano M, Bodini M, Farinha NC, Killeen B. Equivalent-accuracy accelerated neural-network training using analogue memory. Nature. 2018 Jun;558(7708):60-67.

[10] Hu M, Graves CE, Li C, Li Y, Ge N, Montgomery E, Davila N, Jiang H, Williams RS, Yang JJ, Xia Q. Memristor-based analog computation and neural network classification with a dot product engine. Advanced Materials. 2018 Mar;30(9):1705914.

[11] Cai F, Correll JM, Lee SH, Lim Y, Bothra V, Zhang Z, Flynn MP, Lu WD. A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. Nature Electronics. 2019 Jul;2(7):290-299.

[12] Li C, Belkin D, Li Y, Yan P, Hu M, Ge N, Jiang H, Montgomery E, Lin P, Wang Z, Song W. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. Nature communications. 2018 Jun 19;9(1):1-8.

[13] Ramasubramanian SG, Venkatesan R, Sharad M, Roy K, Raghunathan A. SPINDLE: SPINtronic deep learning engine for large-scale neuromorphic computing. In2014 IEEE/ ACM International Symposium on Low Power Electronics and Design (ISLPED) 2014 Aug 11 (pp. 15-20). IEEE.

[14] Ankit A, Hajj IE, Chalamalasetti SR, Ndu G, Foltin M, Williams RS, Faraboschi P, Hwu WM, Strachan JP,

Roy K, Milojicic DS. 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 Apr 4 (pp. 715-731).

[15] Hu M, Graves CE, Li C, Li Y, Ge N, Montgomery E, Davila N, Jiang H, Williams RS, Yang JJ, Xia Q. Memristor-based analog computation and neural network classification with a dot product engine. Advanced Materials. 2018 Mar;30(9):1705914.

[16] Ielmini D, Wong HS. In-memory computing with resistive switching devices. Nature Electronics. 2018 Jun;1 (6):333-343.

[17] Zidan MA, Strachan JP, Lu WD. The future of electronics based on memristive systems. Nature electronics. 2018 Jan;1(1):22-29.

[18] Sebastian A, Le Gallo M, Khaddam-Aljameh R, Eleftheriou E. Memory devices and applications for in-memory computing. Nature nanotechnology. 2020 Jul;15(7):529-544.

[19] Tsai H, Ambrogio S, Narayanan P, Shelby RM, Burr GW. Recent progress in analog memory-based accelerators for deep learning. Journal of Physics D: Applied Physics. 2018 Jun 21;51(28): 283001.

[20] Amirsoleimani A, Alibart F, Yon V, Xu J, Pazhouhandeh MR, Ecoffey S, Beilliard Y, Genov R, Drouin D. In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal–Oxide–Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives. Advanced Intelligent Systems. 2020 Nov;2(11): 2000115.

[21] Shafiee A, Nag A, Muralimanohar N, Balasubramonian R, Strachan JP, Hu M, Williams RS, Srikumar V. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. ACM SIGARCH Computer Architecture News. 2016 Jun 18;44(3):14-26.

[22] Chi P, Li S, Xu C, Zhang T, Zhao J, Liu Y, Wang Y, Xie Y. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. ACM SIGARCH Computer Architecture News. 2016 Jun 18;44(3):27-39.

[23] Yao P, Wu H, Gao B, Tang J, Zhang Q, Zhang W, Yang JJ, Qian H. Fully hardware-implemented memristor convolutional neural network. Nature. 2020 Jan;577(7792):641-646.

[24] Yakopcic C, Alom MZ, Taha TM. Extremely parallel memristor crossbar architecture for convolutional neural network implementation. In2017 International Joint Conference on Neural Networks (IJCNN) 2017 May 14 (pp. 1696-1703). IEEE.

[25] Rahimi Azghadi M, Chen YC, Eshraghian JK, Chen J, Lin CY, Amirsoleimani A, Mehonic A, Kenyon AJ, Fowler B, Lee JC, Chang YF. Complementary Metal-Oxide Semiconductor and Memristive Hardware for Neuromorphic Computing. Advanced Intelligent Systems. 2020 May;2(5):1900189.

[26] Jo SH, Chang T, Ebong I, Bhadviya BB, Mazumder P, Lu W. Nanoscale memristor device as synapse in neuromorphic systems. Nano letters. 2010 Apr 14;10(4): 1297-1301.

[27] Kim KH, Gaba S, Wheeler D, Cruz-Albrecht JM, Hussain T, Srinivasa N, Lu W. A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications. Nano letters. 2012 Jan 11;12 (1):389-395.

[28] Prezioso M, Merrikh-Bayat F, Hoskins BD, Adam GC, Likharev KK, Strukov DB. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature. 2015 May;521(7550):61-64.

[29] Li C, Hu M, Li Y, Jiang H, Ge N, Montgomery E, Zhang J, Song W, Dávila N, Graves CE, Li Z. Analogue signal and image processing with large memristor crossbars. Nature electronics. 2018 Jan;1(1):52-59.

[30] Gao L, Chen PY, Liu R, Yu S. Physical unclonable function exploiting sneak paths in resistive cross-point array. IEEE Transactions on Electron Devices. 2016 Jun 21;63(8):3109-3115.

[31] Nili H, Adam GC, Hoskins B, Prezioso M, Kim J, Mahmoodi MR, Bayat FM, Kavehei O, Strukov DB. Hardware-intrinsic security primitives enabled by analogue state and nonlinear conductance variations in integrated memristors. Nature Electronics. 2018 Mar;1(3):197-202.

[32] Jiang H, Li C, Zhang R, Yan P, Lin P, Li Y, Yang JJ, Holcomb D, Xia Q. A provable key destruction scheme based on memristive crossbar arrays. Nature Electronics. 2018 Oct;1(10):548-554.

[33] Cai F, Kumar S, Van Vaerenbergh T, Liu R, Li C, Yu S, Xia Q, Yang JJ, Beausoleil R, Lu W, Strachan JP. Harnessing intrinsic noise in memristor Hopfield neural networks for combinatorial optimization. arXiv preprint arXiv:1903.11194. 2019 Mar 26.

[34] Bojnordi MN, Ipek E. Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In2016 IEEE International Symposium on High Performance Computer Architecture (HPCA) 2016 Mar 12 (pp. 1-13). IEEE.

[35] Liu S, Wang Y, Fardad M, Varshney PK. A memristor-based optimization framework for artificial intelligence applications. IEEE Circuits and Systems Magazine. 2018 Feb 9;18 (1):29-44.

[36] Sheridan PM, Cai F, Du C, Ma W, Zhang Z, Lu WD. Sparse coding with memristor networks. Nature nanotechnology. 2017 Aug;12(8):784.

[37] Zidan MA, Jeong Y, Lee J, Chen B, Huang S, Kushner MJ, Lu WD. A general memristor-based partial differential equation solver. Nature Electronics. 2018 Jul;1(7):411-420.

[38] Chen A, Datta S, Hu XS, Niemier MT, Rosing TŠ, Yang JJ. A survey on architecture advances enabled by emerging beyond-CMOS technologies. IEEE Design & Test. 2019 Feb 28;36(3):46-68.

[39] Jain S, Ankit A, Chakraborty I, Gokmen T, Rasch M, Haensch W, Roy K, Raghunathan A. Neural network accelerator design with resistive crossbars: Opportunities and challenges. IBM Journal of Research and Development. 2019 Oct 11;63(6):10-11.

[40] Mittal S. A survey of ReRAM-based architectures for processing-in-memory and neural networks. Machine learning and knowledge extraction. 2019 Mar;1 (1):75-114.

[41] Cheng M, Xia L, Zhu Z, Cai Y, Xie Y, Wang Y, Yang H. Time: A training-in-memory architecture for memristor-based deep neural networks. In2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC) 2017 Jun 18 (pp. 1-6). IEEE.

[42] Chen L, Li J, Chen Y, Deng Q, Shen J, Liang X, Jiang L. Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar. InDesign, Automation & Test in Europe Conference & Exhibition (DATE), 2017 2017 Mar 27 (pp. 19-24). IEEE.

[43] Xia L, Liu M, Ning X, Chakrabarty K, Wang Y. Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems. In Proceedings of the 54th Annual Design Automation Conference 2017 2017 Jun 18 (pp. 1-6).

[44] Liu C, Hu M, Strachan JP, Li H. Rescuing memristor-based neuromorphic design with high defects. In2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC) 2017 Jun 18 (pp. 1-6). IEEE.

[45] Liu X, Mao M, Liu B, Li H, Chen Y, Li B, Wang Y, Jiang H, Barnell M, Wu Q, Yang J. RENO: A high-efficient reconfigurable neuromorphic computing accelerator design. In Proceedings of the 52nd Annual Design Automation Conference 2015 Jun 7 (pp. 1-6).

[46] Li B, Shan Y, Hu M, Wang Y, Chen Y, Yang H. Memristor-based approximated computation. In: International Symposium on Low Power Electronics and Design (ISLPED); 2013; Beijing, China. p. 242-247. DOI: 10.1109/ISLPED.2013.6629302.

[47] Yan B, Yang J, Wu Q, Chen Y, Li H. A closed-loop design to enhance weight stability of memristor based neural network chips. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD); 2017; Irvine, CA, USA. p. 541-548. DOI: 10.1109/ICCAD.2017.8203824.

[48] Li B, Shan Y, Hu M, Wang Y, Chen Y, Yang H. Memristor-based approximated computation. In: International Symposium on Low Power Electronics and Design (ISLPED); 2013; Beijing, China. p. 242-247. DOI: 10.1109/ISLPED.2013.6629302.

[49] Alibart F, Zamanidoost E, Strukov DB. Pattern classification by memristive crossbar circuits using ex situ and in situ training. Nature communications. 2013 Jun 25;4(1):1-7.

[50] Chen Y, Li H, Wang X, Zhu W, Xu W, Zhang T. A nondestructive self-reference scheme for spin-transfer torque random access memory (STT-RAM). In 2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010) 2010 Mar 8 (pp. 148-153). IEEE.

[51] Niu D, Xiao Y, Xie Y. Low power memristor-based ReRAM design with error correcting code. In 17th Asia and South Pacific Design Automation Conference 2012 Jan 12 (pp. 79-84). IEEE.

[52] Oh S, Huang Z, Shi Y, Kuzum D. The impact of resistance drift of phase change memory (PCM) synaptic devices on artificial neural network performance. IEEE Electron Device Letters. 2019 Jul 2;40(8):1325-1328.

[53] Hsieh CC, Chang YF, Jeon Y, Roy A, Shahrjerdi D, Banerjee SK. Short-Term Relaxation in HfOx/CeOx Resistive Random Access Memory with Selector. IEEE Electron Device Letters. 2017 Jun 1;38(7):871-874.

[54] Yan B, Yang J, Wu Q, Chen Y, Li H. A closed-loop design to enhance weight stability of memristor based neural network chips. In2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2017 Nov 13 (pp. 541-548). IEEE.

[55] Li B, Wang Y, Chen Y, Li H, Yang H. ICE: Inline calibration for memristor crossbar-based computing engine. In: 2014 Design, Automation & Test in Europe Conference & Exhibition; 2014; Dresden, Germany. p. 1-4. DOI: 10.7873/DATE.2014.197.

[56] Liu T, Amirsoleimani A, Alibart F, Ecoffey S, Drouin D, Genov R. AIDX: Adaptive Inference Scheme to Mitigate State-Drift in Memristive VMM Accelerators. IEEE Transactions on Circuits and Systems II: Express Briefs. 2021;68:4:1128-1132. DOI: 10.1109/TCSII.2020.3026642.

[57] Kvatinsky S, Ramadan M, Friedman EG, Kolodny A, Weiser UC. TEAM: ThrEshold Adaptive Memristor Model. IEEE Transactions on Circuits and Systems I: Regular Papers. 2013;60: 1:211-221. DOI: 10.1109/TCSI.2012.2215714.

[58] Kvatinsky S, Ramadan M, Friedman EG, Kolodny A. VTEAM: A General Model for Voltage-Controlled Memristors. IEEE Transactions on Circuits and Systems II: Express Briefs. 2015;62:8:786-790. DOI: 10.1109/TCSII.2015.2433536.

[59] Biolek Z, Biolek D, Biolkova V. SPICE Model of Memristor with Nonlinear Dopant Drift. Radioengineering. 2009.

[60] Pickett MD, Strukov DB, Borghetti JL, Yang JJ, Snider GS, Stewart DR, Williams RS. Switching dynamics in titanium dioxide memristive devices. Journal of Applied Physics. 2009; 106:7:1–6. DOI: 10.1063/1.3236506

[61] Pino RE, Li H, Chen Y, Hu M, Liu B. Statistical memristor modeling and case study in neuromorphic computing. In: DAC Design Automation Conference 2012; 2012; San Francisco, CA, USA. p. 585-590. DOI: 10.1145/2228360.2228466.

[62] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998; 86:11: 2278-2324. DOI: 10.1109/5.726791.

[63] Prechelt L. PROBEN 1-a set of benchmarks and benchmarking rules for neural network training algorithms. 1994.

[64] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images. 2009.

[65] Li B, Shan Y, Hu M, Wang Y, Chen Y, Yang H. Memristor-based approximated computation. In: International Symposium on Low Power Electronics and Design (ISLPED); 2013; Beijing, China. p. 242-247. DOI: 10.1109/ISLPED.2013.6629302.

[66] Strachan J, Torrezan A, Miao F, Pickett M, Yang J, Yi W, Medeiros-Ribeiro G, Williams S. State Dynamics and Modeling of Tantalum Oxide Memristors. IEEE Transactions on Electron Devices. 2013; 60:7:2194-2202. DOI: 10.1109/TED.2013.2264476

[67] Fletcher R. Practical methods of optimization. John Wiley & Sons; 2013. DOI: 10.1002/9781118723203.