



HAL
open science

AIDX: Adaptive Inference Scheme to Mitigate State-Drift in Memristive VMM Accelerators

Tony Liu, Amirali Amirsoleimani, F. Alibart, Serge Ecoffey, Dominique Drouin, Roman Genov

► **To cite this version:**

Tony Liu, Amirali Amirsoleimani, F. Alibart, Serge Ecoffey, Dominique Drouin, et al.. AIDX: Adaptive Inference Scheme to Mitigate State-Drift in Memristive VMM Accelerators. IEEE Transactions on Circuits and Systems II: Express Briefs, 2021, 68 (4), pp.1128 - 1132. 10.1109/TCSII.2020.3026642 . hal-03095208

HAL Id: hal-03095208

<https://hal.science/hal-03095208v1>

Submitted on 2 May 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AIDX: Adaptive Inference Scheme to Mitigate State-Drift in Memristive VMM Accelerators

Tony Liu, Amirali Amirsoleimani, Fabien Alibart, Serge Ecoffey, Dominique Drouin, and Roman Genov

Abstract—An adaptive inference method for crossbar (AIDX) is presented based on an optimization scheme for adjusting the duration and amplitude of input voltage pulses. AIDX minimizes the long-term effects of memristance drift on artificial neural network accuracy. The sub-threshold behavior of memristor has been modeled and verified by comparing with fabricated device data. The proposed method has been evaluated by testing on different network structures and applications, e.g., image reconstruction and classification tasks. The results showed an average of 60% improvement in convolutional neural network (CNN) performance on CIFAR10 after 10000 inference operations as well as 78.6% error reduction in image reconstruction.

Index Terms—Memristor, Crossbar, Vector-Matrix Multiplication, Inference, State-Drift, Neural Network.

I. INTRODUCTION

RESISTIVE switching memory crossbars have emerged as potentially high-speed and low-power accelerators for vector-matrix multiplication (VMM) [1], [2]. However, non-idealities and defects in these platforms dramatically impact the neural network (NN) performance and accuracy. One of the significant and not extensively studied non-ideal phenomena is memristance drift [3] and it occurs in different types of resistive switching memory technologies in various ways. For instance, phase change memories (PCM) will experience increasing resistance due to drift, even when there is no voltage applied over the cell [4]. On the other hand, for memristors, state-drift from their programmed state happens as a result of many repeated VMM operations which leads to the computational accuracy degradation (Fig. 1). Previous studies [5]–[7] on memristance drift in memristor technology have mainly been focused on high-density memory where memristors are used solely for storage rather than computation. More recent reports on drift [8] for computational memristor crossbars include an inline calibration approach [9] which involves optimizing the calibration time of the memristor crossbar. By performing polynomial fitting on the computational error data, a 21.77% calibration efficiency is achieved. A closed-loop weight compensation based solution is presented in [10] which minimizes the effects of state-drift by increasing the computational service lifetime by $14.95\times$ and results in approximately 70% computational accuracy degradation within 1705 read operations. In this brief, we present an adaptive inference scheme (AIDX) as a flexible optimization procedure that automatically adapts to existing crossbar non-idealities and circuit parasitics and can be applied to any VMM-based task. According to experimentally verified simulations, AIDX

cuts accuracy loss due to the device state-drift in modern convolutional neural networks (CNN) by more than 60% as well as a 78.6% error reduction in image reconstruction.

II. PRELIMINARY

A. Impact of Memristance Drift on Crossbar MAC Operations

Memristance drift is defined as the unintended small changes in memristor conductance caused by a low-voltage read/inference operation. Ideally, for the ideal weight distribution (\mathbf{G}) the output current j -th column I_j is given by $I_j = \sum_i G_{ij} V_i$ (Fig. 1(a)). We can define the memristance drift caused by the k -th inference operation as δG_k and the conductance of the memristor at the $(k+1)$ -th iteration as:

$$G_{k+1} = G_k + \delta G_k \quad (1)$$

The total memristance drift due to the k -th operation is $\Delta G = \sum_i^k \delta G_i$. As such, the real output current of the j -th column at the k -th operation is $I'_j = \sum_i (G_{ij} + \Delta G_{ij}) V_i$. The current error $I - I'$ due to memristance drift can be quite problematic in larger crossbars because current scales with crossbar size. However, a differential mapping scheme can prevent the build-up of memristance drift error in very large arrays because the error in the positive column will scale at the same rate as the negative column. Fig. 1(b) illustrates the concept of small changes in NN weights accumulating into much larger errors in the output layer. Fig. 1(c) illustrates 3D structure of the network in Fig. 1(b). Fig. 1(d) shows sample heatmaps of simulated 32×32 array of memristors conductance changes due to memristance drift. The bottom row represents the bias weights of a NN and they are initially mapped to a high-conductance state which is why it is the only row with reduction in overall conductance. Fig. 1(e) examines the state-drift impact on MNIST classification task for multi layer perceptron network with various number of hidden layers and Fig. 1(f) illustrates the difference between above- and sub-threshold memristor switching.

B. Memristance drift modeling and analysis

Behaviour-based memristor models are typically used in memristor crossbar simulations due to their simplicity and light computational load. However, most behaviour-based models do not consider memristance drift and approximate the internal state change due to an applied sub-threshold voltage to be zero. To address this issue, we propose an extension to the popular VTEAM model [11] that accounts for the minute changes in internal state due to sub-threshold voltages. For

Tony Liu, Amirali Amirsoleimani, and Roman Genov are with the Department of Electrical and Computer Engineering, University of Toronto, 10 Kings College Road, Toronto, Ontario, Canada. Fabien Alibart, Serge Ecoffey and Dominique Drouin are with the Interdisciplinary Institute for Technological Innovation - 3IT, University of Sherbrooke, Qc, Canada. (A. Amirsoleimani corresponding author email: amirali.amirsoleimani@utoronto.ca).

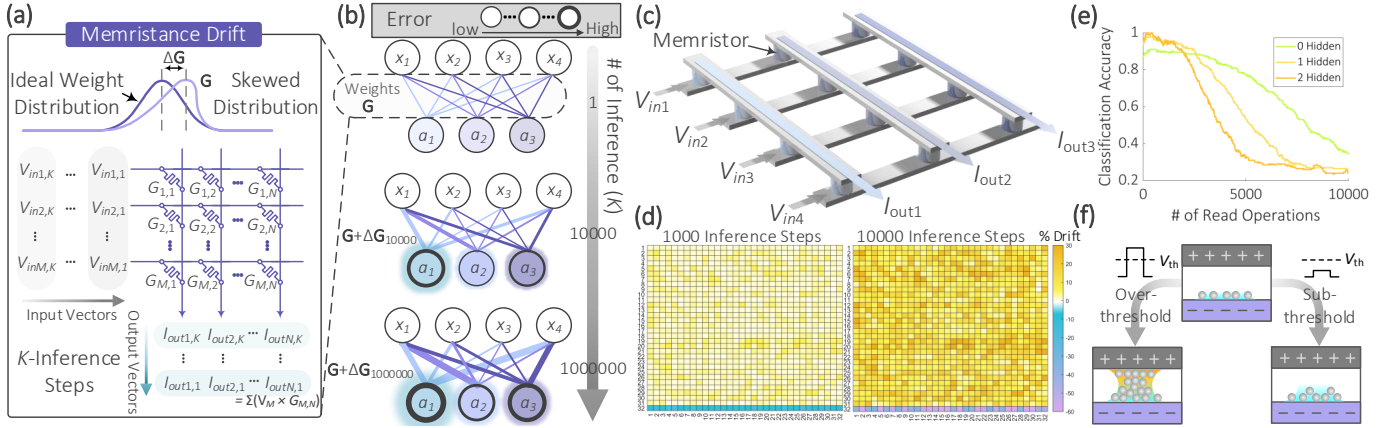


Fig. 1. (a) Neural network (NN) forward pass VMM performed on a memristor crossbar. (b) Degradation of neural network weights and output accuracy over time. (c) 3D memristor crossbar array structure. (d) Percentage change of memristors conductance in a randomly chosen subset of NN weights done on a simulated memristor crossbar. (e) Impact of simulated state-drift on MNIST classification accuracy with varying number of hidden layer. (f) Sub- and over-threshold behaviour of device transition oxide filament.

model consistency, we adopt a similar mathematical structure in the sub- and above-threshold region:

$$\frac{dw(t)}{dt} = \begin{cases} k_{s,off} \cdot \left(\frac{v(t)}{v_{off}}\right)^{\alpha_{s,off}} \cdot f_{s,off}(w), & \text{if } 0 \leq v < v_{off} \\ k_{s,on} \cdot \left(\frac{v(t)}{v_{on}}\right)^{\alpha_{s,on}} \cdot f_{s,on}(w), & \text{if } v_{on} < v < 0 \end{cases} \quad (2)$$

Here, v_{off} and v_{on} represent the RESET and SET voltage thresholds respectively. $w(t)$ is the internal state variable and is related to the resistance R of the memristor as $R(t) = R_{off}w(t) + R_{on}(1 - w(t))$. $k_{s,off}$ and $k_{s,on}$ are fitting parameters that represent the rate of ion migration at any given applied sub-threshold voltage. Similarly, $\alpha_{s,off}$ and $\alpha_{s,on}$ are parameters that characterize the exponential relationship between speed of ion migration and the applied voltage. $f_{s,on}(w)$ and $f_{s,off}(w)$ are window functions that bounds the state between 0 and 1. The time derivative of the resistance can be expressed as:

$$\frac{dR(t)}{dt} = R_{off} \frac{dw(t)}{dt} - R_{on} \frac{dw(t)}{dt} \quad (3)$$

R_{on} and R_{off} are low and high resistance state of device, respectively. Cycle-to-cycle and device-to-device variations in sub-threshold drift speed are modelled by adding 15% random Gaussian noise to k and α parameters. The probability density function (PDF) of k_{on} is shown in Eqn. (4) where k_{on} is the ideal, fitted parameter and x represents k_{on} with added Gaussian noise. The PDF of the other k and α parameters follow the same structure as Eqn. (4).

$$f_{kon}(x) = \frac{1}{\sqrt{0.3k_{on}\pi}} e^{-\frac{(x-k_{on})^2}{0.3k_{on}}} \quad (4)$$

To validate our proposed model, the VTEAM extension is applied to TiOx-based memristor device (Fig. 2(a)) data. The extended VTEAM k and α parameters were fit using simulated annealing algorithms and gradient descent with SET and RESET voltage thresholds of $-0.6V$ and $0.6V$. Fig. 2(b-c) illustrates that the extended VTEAM models sub-threshold memristor behaviour much more accurately than the original VTEAM. Fig. 2(d) shows a 3D plot of how memristor switching behavior and conductance changes with internal state w and applied voltage in sub-threshold region.

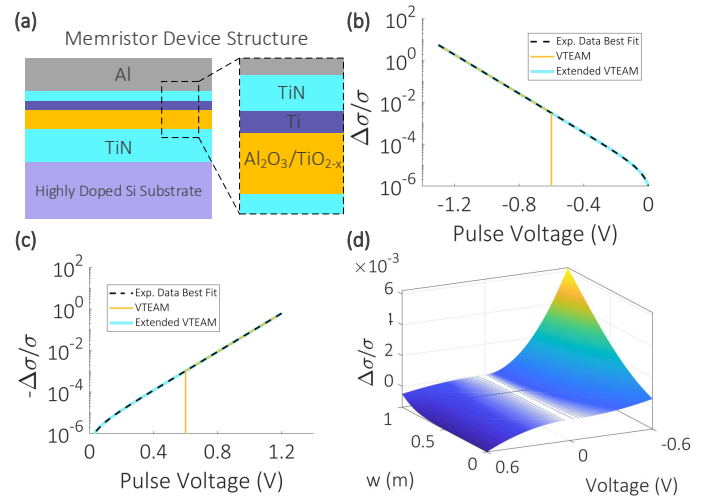


Fig. 2. (a) Memristor device structure used for fitting. (b) Comparison of I - V fitting of the extended VTEAM with experimental data and the original model in sub-threshold region for SET operation ($k_{s,on} = -8.445 \times 10^{-6}$, $\alpha_{s,on} = 6$). For clarity, we extracted a best fit curve to represent the experimental data from over the threshold and extrapolated the curve into sub-threshold region by keeping the gradual drift trend.

(c) Fitting comparison with experimental data for RESET operation ($k_{s,off} = 1.126 \times 10^{-7}$, $\alpha_{s,off} = 5$). (d) 3D characterization of the extended VTEAM for the same device with respect to the internal state variable w and voltage.

III. METHODOLOGY

A. Problem and Assumptions

By formulating the issue of memristance drift as an optimization problem, we can develop an optimization scheme to minimize accuracy degradation. With no memristance drift, the ideal mean squared error (MSE) is $E_0 = \sum_j (y_j - \sum_i G_{ij} V_i)^2$ and the real MSE at the k -th operation is $E_k = \sum_j (y_j - \sum_i (G_{ij} + \Delta G_{ij}) V_i)^2$. Where V_i is the voltage applied to i -th row and ΔG_{ij} is the total memristance drift of the ij -th memristor from its originally programmed value. We define the error due to memristance drift E_{Drift} as the difference in MSE between the initially programmed state (E_0) and the k -th inference operation (E_k). As an optimization problem, the goal is to minimize the increase of E_{Drift} with respect to time. The change in conductance due to memristance drift,

ΔG , can mainly be optimized by input to voltage amplitude mapping and relative inference voltage pulse width. Factors that cannot be easily changed such as specific memristor characteristics and overall crossbar structure will be ignored in the optimization procedure.

B. Optimization Methodology

We will frame the minimization of E_{Drift} as an unconstrained optimization problem where \mathbf{A} is the input to voltage amplitude mapping and \mathbf{D} is the relative voltage pulse width:

$$\min_{\mathbf{A}, \mathbf{D}} E_{Drift}(\mathbf{A}, \mathbf{D}) \quad (5)$$

\mathbf{D} is a vector that represents the length of a positive input read pulse relative to a negative read pulse. For instance, a given row can have a positive read pulse of $200ns$ while the negative read pulse is only $150ns$ long. Similarly, \mathbf{A} is a vector whose elements represent the relative inference voltage pulse amplitude ratio for positive to negative inputs (Fig. 3a). In summary, AIDX modifies the amplitude and duration of inference voltage pulses to minimize memristance drift for a given task. Even the minimum allowable voltage pulse amplitude and widths will still result in noticeable memristance drift after many inference operations. As such, AIDX is required to minimize aggregate memristance drift through balancing the total drift in the SET and RESET directions. We use the popular Broyden-Fletcher-Goldfarb-Shannon (BFGS) algorithm [12] for this optimization problem. The BFGS algorithm is a quasi-Newton method that relies on the gradient of the objective function to find the optimal solution. However, E_{Drift} is an unknown function that can only be evaluated, so the ∇E_{Drift} had to be approximated using finite-difference approach. Gradient-free optimization methods like Nelder-Mead simplex method [13] were also explored, but quasi-Newton algorithms were most effective for this problem.

C. Constraint Violations

Normally, the optimized voltage amplitude ratio \mathbf{A} and width ratio \mathbf{D} can be reasonably used. However, there are certain cases where elements of the optimal \mathbf{A} and \mathbf{D} are far too large or too small to be implemented practically, typically when the device characteristics and input distribution are heavily skewed. To address this issue, we will first frame the optimization problem through a different lens. Let's start with a simplified scenario of a single memristor with input data

modelled by the discrete random variable X with a probability density function (PDF) of $f(x)$. Defining the time derivative of the internal state w for a given x as

$$\frac{dw(x)}{dx} = g(x). \quad (6)$$

In our sub-threshold model, $g(x)$ is the same as Eqn. (2) as $v(t)$ replaced with $v(x)$ which represents the mapping function of input x to voltage amplitude $v(x)$. The average rate of memristance drift given input distribution X is as follows:

$$E\left[\frac{dw(x)}{dx}\right] = \sum_x g(x)f(x). \quad (7)$$

The optimization problem over E_{Drift} can also be reframed as minimizing $\left|E\left[\frac{dw(x)}{dt}\right]\right|$ over all memristors where the $g(x)$ parameters for each memristor is sampled according to Eqn. (4) to account for device-to-device variations. The AIDX scheme defined so far only affects $g(x)$, but has not yet made any adjustments related to $f(x)$. If we allow AIDX to first optimize over $f(x)$, the issue of impractical \mathbf{A} or \mathbf{D} can be circumvented. One of the only useful recoverable transformations of the input vector x is inversion through multiplying by -1 . By inverting a random proportion a of the input data, the input PDF is transformed into $f'(x)$:

$$f'(x) = (1 - 2a)f(x), 0 < a < 1 \quad (8)$$

Impractical \mathbf{A} or \mathbf{D} only occur when either $\left|E\left[\frac{dw(x)}{dt}\right]\right| \gg 0$ or $\left|E\left[\frac{dw(x)}{dt}\right]\right| \ll 0$ before applying AIDX. As such, if we can optimize $\left|E\left[\frac{dw'(x,a)}{dt}\right]\right|$ over a , $\left|E\left[\frac{dw(x)}{dt}\right]\right|$ will be brought close to 0 before optimizing \mathbf{A} and \mathbf{D} which will therefore prevent any constraint violations. Fig. 3(b) illustrates our approach to constraint violations.

D. General Solution Flow

As it can be seen in Fig. 3(c), during pre-processing, optimization is done in three separate scenarios to guarantee optimal fitting parameters. Once hardware constraint violations are resolved with input data inversion, the input circuits e.g. digital-to-analog converters (DACs) are adjusted to fit the optimized input to voltage signal mapping parameters. The majority of AIDX takes place during pre-processing which only needs to be done once for any given task. The only difference in the AIDX inference operation as compared to a normal inference operation is to recover the intended output current from an inverted output through multiplying by -1 . Fig. 3(d) summarizes the general pipeline of AIDX. Fig. 3(e) shows the evolution of memristor state due to memristance drift for AIDX and the baseline model and Fig. 3(f) is a heatmap of a portion of the memristor crossbar at 1000 and 10000 inference steps where the bottom row represents the bias. While memristance drift is a phenomenon that can cause memristors to switch in both the set and reset direction as seen in Fig. 3(e), almost all memristors within a crossbar will typically drift in only one direction for image-based applications. These inputs are almost entirely positive which causes an aggregate drift in the reset direction. Other reasons for

Algorithm 1: BFGS Algorithm

1. Obtain a direction \mathbf{p}_k through solving $B_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$
 2. Perform 1D Line Search to find step size α_k such that $\alpha_k = \operatorname{argmin} f(\mathbf{x}_k + \alpha \mathbf{p}_k)$
 3. $\mathbf{s}_k = \alpha_k \mathbf{p}_k$
 4. $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
 5. $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$
 6. $B_{k+1} = B_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{B_k \mathbf{s}_k \mathbf{s}_k^T B_k^T}{\mathbf{s}_k^T B_k \mathbf{s}_k}$
 7. Repeat 1-6 until x converges.
-

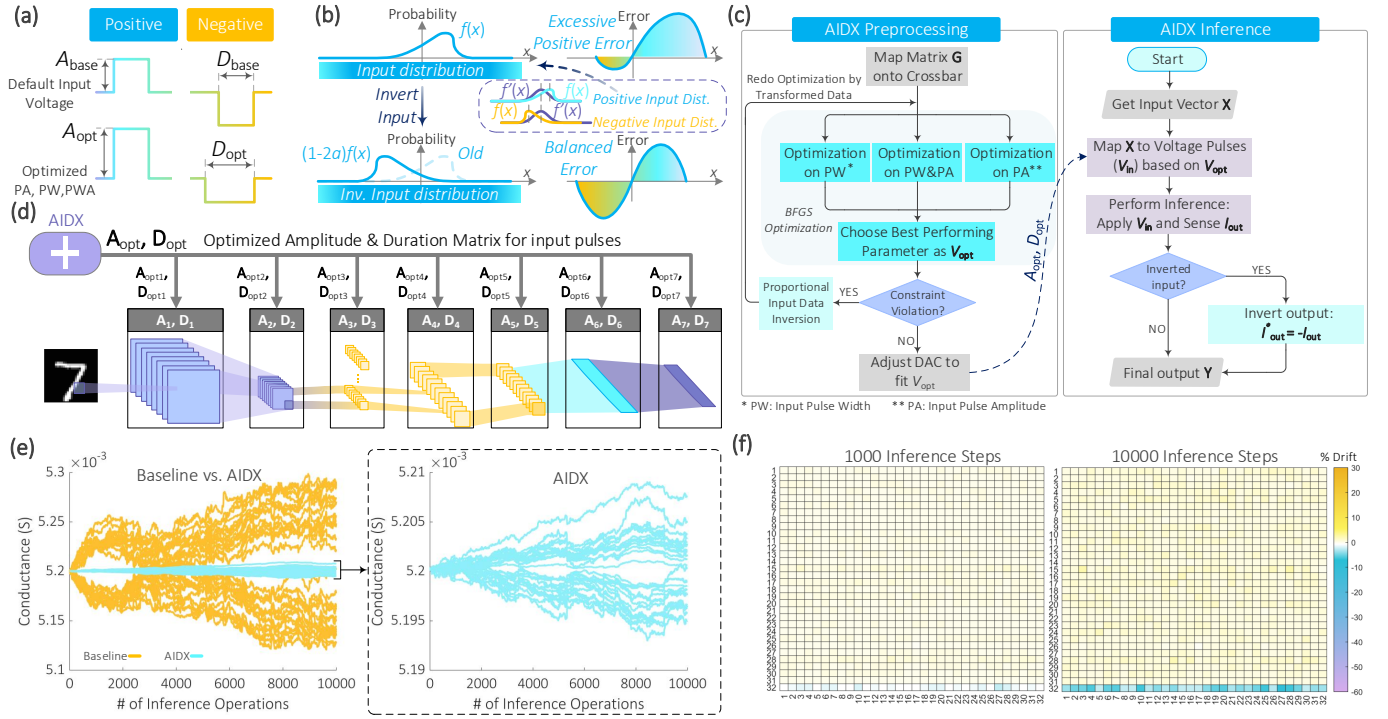


Fig. 3. (a) AIDX’s optimized parameters A and D mapped onto input voltage pulses. (b) Proportional input inversion to balance memristance drift error and prevent constraint violations. (c) AIDX Design flowchart. (d) AIDX optimized parameters for duration (D_{opt}) and amplitude (A_{opt}) are applied to each CNN layer’s input separately. (e) Simulated memristance drift with and without AIDX over time with 15% device-to-device variations. To better observe the total state-drift, all devices initial conductance are set to 0.0052 S and a positively skewed pre-generated random sequence of voltage pulses were applied to half the memristors and a negatively skewed pulses to the other half. (f) Percentage change in conductance of the same simulated memristors shown in Fig. 1(d) by utilizing AIDX.

a unidirectional aggregate drift include: the device switching speed is not the same in the set and reset direction and most non-biased memristors are being initialized close to the high resistive state where the drift speed is strongly skewed in the set direction (Fig 2(d)).

IV. RESULTS AND DISCUSSION

In this paper, all simulations are performed using our extended VTEAM memristor model [14] by including the effects of sub-threshold state-drift whose parameters are fit according to the experimental data shown in Fig. 2. We integrate this memristor model into our existing 1T1R memristor crossbar simulation to simulate both memristance drift and crossbar non-idealities like sneak paths and line resistance. A differential weight mapping scheme is used where each element is mapped onto a pair of memristors where one memristor represents positive values and the other represents negative values. In Fig. 4(a), AIDX is tested across 10 baseline tasks from the Proben1 benchmark datasets [14]. We trained a shallow 1-hidden layer NN for all of these tasks. While there are large variations in baseline performance across different tasks, it should be noted that all baseline tasks ended at around the same classification accuracy as random guessing due to some tasks having more classification categories than others. To verify our solution’s effectiveness for more practical applications, we adopted AIDX for a selection of CNN architectures on the CIFAR10 dataset. The CNN memristor crossbar mapping scheme used is similar to the one found in [15]. Fig. 4(b) compares the performance of 10 different CNN architectures between AIDX and the baseline model. As

compared to the shallow NNs used for the Proben1 datasets, the CNNs had an overall higher speed of accuracy degradation. The worse performance of CNNs is to be expected because of error propagation from one layer to the next amplifying the effect of memristance drift. The error in column j of the $l + 1 - th$ layer in a fully connected NN is:

$$E_{j,l+1} = \sum_i^n V_{i,l+1}(\sigma(E_{i,l}) + \Delta G_{ij,l+1}) \quad (9)$$

Here, n is the total length of the input vector and σ is the activation function of the $l - th$ layer. Due to the large number of parameters in modern CNNs, BFGS optimization in AIDX is performed sequentially layer by layer to reduce optimization time. Applying AIDX to the selected CNNs provided consistent improvements in classification accuracy on CIFAR10. The consistent improvement in AIDX performance across varying sizes and designs of CNNs demonstrates the proposed method flexibility across different crossbar sizes and structure. In addition to classification tasks, we wanted to demonstrate AIDX’s effectiveness in a different type of memristor crossbar application. Fig. 4(c) shows the results of image reconstruction with the MNIST dataset. For this task, a 1-hidden layer auto-encoder with 32 hidden units was trained off-chip which corresponds to a $24.5\times$ compression factor. With AIDX, the average mean squared error has improved by 78.6% over the baseline after 10000 inference operations.

V. OVERHEAD ANALYSIS AND COMPARISON

Different state-drift mitigation techniques have been compared with two different AIDX configurations optimized for

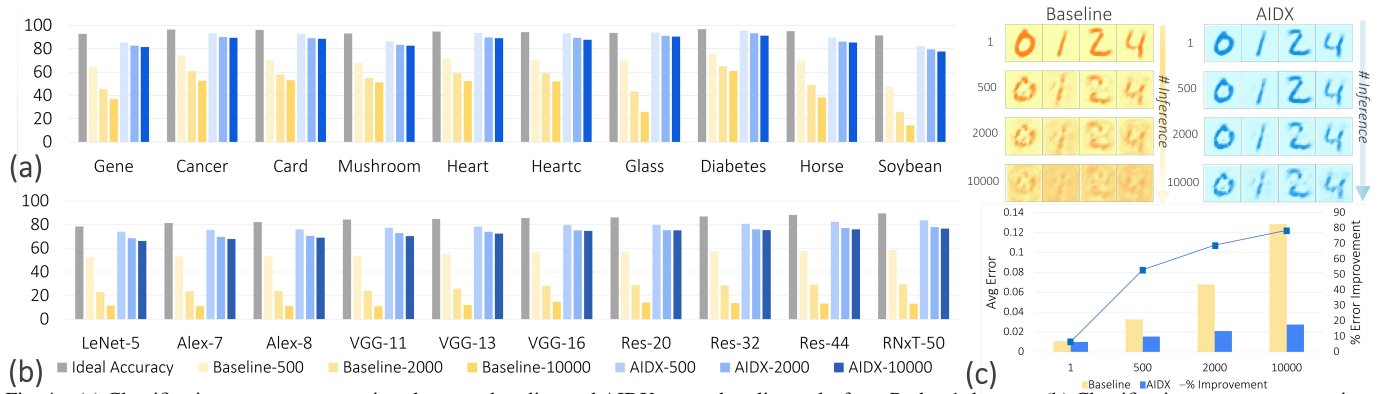


Fig. 4. (a) Classification accuracy comparison between baseline and AIDX across baseline tasks from Proben1 datasets. (b) Classification accuracy comparison across different CNN architectures on CIFAR-10 image classification dataset. (c) Sample reconstructed MNIST images and average image reconstruction error from baseline and AIDX-enhanced auto encoders.

TABLE I
COMPARATIVE ANALYSIS OF AIDX.

Methods	[5]	[6]	AIDX-A	AIDX-P
Power overhead (%)	-	+1.61	+3.27	+1.19
Area overhead (%)	0	+2.34	0	0
Performance life-time	1.22×	14.85×	37.62×	31.41×
Scalability vs Baseline	Worse	Worse	Better	Better
Accuracy improvement (%)	8.6	37.3	65.7	57.4
Include non-idealities	No	No	Yes	Yes

accuracy (AIDX-A) and power efficiency (AIDX-P) to implement a MLP network in Table 1. AIDX-A is the baseline AIDX method discussed in previous sections while AIDX-P adds a L2 regularization terms for \mathbf{A} and \mathbf{D} as follows: $\min_{\mathbf{A}, \mathbf{D}} (E_{\text{Drift}}(\mathbf{A}, \mathbf{D}) + \lambda_1 \sum A^2 + \lambda_2 \sum D^2)$. Where λ_1 and λ_2 are regularization constants and regularizing the voltage amplitude and width ratios allows AIDX-P to reduce the passive crossbar power consumption. For the sake of consistency, we use the same estimates of peripheral power consumption as [10]. Crossbar power consumption in AIDX is computed as the average power consumed across the memristors in one inference operation. Area overhead is defined as the percentage increase in on-chip area required for the memristance drift solution due to peripherals, external circuit, and other items. Accuracy improvement is the increase in classification accuracy provided by a solution over the baseline model in a 1-hidden layer MLP at the end of the baseline models defined lifetime. Performance lifetime is defined as the amount of time required for a system to degrade to 70% classification accuracy on the MNIST dataset. We chose this metric as an axis of comparison primarily because it is used in [10] and is easily adaptable to the Interrupt and Benchmark method used in [9]. Scalability is a measure of how well a memristance drift solutions performance and overhead scales with crossbar size and additional layers in NN applications. Time overhead is not shown in Table 1 because there is negligible time overhead introduced by all solutions presented as compared to their respective baseline models.

VI. CONCLUSION

In this paper, we propose a new inference scheme based on voltage signal optimization called AIDX to reduce the impact of memristance drift on memristor crossbar MAC operations. By optimizing the voltage pulse width and amplitude input

mapping, AIDX is flexible and effective across a different range of tasks including classification and image reconstruction. AIDX minimizes the computational error due to memristance drift. AIDX provides up to a 60% and 78.60% increase in classification accuracy on the CIFAR-10 datasets and image reconstruction of MNIST dataset, respectively. In addition, we have proposed an extension to the popular VTEAM model to more precisely simulate memristor behaviour below the switching voltage thresholds.

ACKNOWLEDGMENT

This work is supported by NSERC HIDATA project and ERC-CoG IONOS n773228.

REFERENCES

- [1] P. Yao et al., Fully hardware-implemented memristor convolutional neural network, *Nature*, vol. 577, no. 7792, pp. 641646, 2020.
- [2] C. Li et al., Efficient and self-adaptive in-situ learning in multilayer memristor neural networks, *Nature comm.*, vol. 9, no. 1, pp. 18, 2018.
- [3] T. Chang, et al., Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor, *ACS Nano*, vol. 5, no. 9, pp. 76697676, 2011.
- [4] S. Oh, et al., "The Impact of Resistance Drift of Phase Change Memory (PCM) Synaptic Devices on Artificial Neural Network Performance," in *IEEE Electron Device Letters*, vol. 40, no. 8, pp. 1325-1328, 2019.
- [5] S. S.-. Sheu et al., "A 4Mb embedded SLC resistive-RAM macro with 7.2 ns read-write random-access time and 160ns MLC-access capability", 2011 *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, pp. 200-202, 2011.
- [6] Y. Chen, et al., "A nondestructive self-reference scheme for spin-transfer torque random access memory (STT-RAM)", 2010 *Design Autom. & Test in Europe Conf. & Exh. (DATE)*, pp. 148-153, 2010.
- [7] D. Niu, et al., "Low power memristor-based RERAM design with error correcting code", 2012 *17th Asia and South Pacific Design Autom. Conf. (ASP-DAC)*, pp. 79-84, 2012.
- [8] V. Joshi, et al., "Accurate deep neural network inference using computational phase-change memory" *Nature Comm.*, vol. 11, no. 1, pp. 1-13, 2020.
- [9] B. Li, et al., "Memristor-based approximated computation," 2013 *Int. Symp. on Low Power Elec. and Design (ISLPEd)*, pp. 242-247, 2013.
- [10] B. Yan, et al., "A closed-loop design to enhance weight stability of memristor based neural network chips," 2017 *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 541-548, 2017.
- [11] S. Kvatinsky, et al., "VTEAM: A General Model for Voltage-Controlled Memristors," in *IEEE Trans. on Circuits and Sys. II: Exp. Briefs*, vol. 62, no. 8, pp. 786-790, 2015.
- [12] R. Fletcher, "Practical methods of optimization. J. Wiley & Sons, 2013.
- [13] L. S. Nelson, *Nelder-Mead Simplex Method*, Wiley StatsRef: Statistics Reference Online, 2014.
- [14] L. Prechelt, "PROBEN 1-a set of benchmarks and benchmarking rules for neural network training algorithms.", 1994.
- [15] F. Zhang, et al, "Mitigate Parasitic Resistance in Resistive Crossbar-based Convolutional Neural Networks." *ACM Journal on Emerg. Tech. in Computing Sys. (JETC)*, vol. 16, no. 3, pp. 1-20, 2020.