Exploiting the State Dependency of Conductance Variations in Memristive Devices for Accurate In-Memory Computing

Athanasios Vasilopoulos, *Graduate Student Member, IEEE*, Julian Büchel, *Member, IEEE*, Benedikt Kersting, Corey Lammie, *Member, IEEE*, Kevin Brew, Samuel Choi, Timothy Philip, *Member, IEEE*, Nicole Saulnier, Vijay Narayanan, *Senior Member, IEEE*, Manuel Le Gallo, *Member, IEEE*, and Abu Sebastian, *Fellow, IEEE*

***Abstract*— Analog in-memory computing (AIMC) using memristive devices is considered a promising Non-von Neumann approach for deep learning (DL) inference tasks. However, inaccuracies in the programming of devices, that are attributed to conductance variations, pose a key challenge toward achieving sufficient compute precision for DL inference. Fortunately, conduction variations in memristive devices, such as phase-change memory (PCM) devices, exhibit a strong state dependence. This state dependence can be exploited in synaptic unit cells that comprise more than one memristive device, to encode positive or negative weights. In such multi-memristive unit cells, we propose a method that optimally maps the weights to the device conductance values, by maximizing the number of devices at the stable SET and RESET states. We demonstrate that this method reduces the matrix-vector multiplication (MVM) error and is more resilient to non-ideal device retention characteristics. With this approach, we increase the mean experimental inference accuracy of a network trained for MNIST classification by 0.71% on two PCM-based AIMC cores, and the hardware-realistic simulated top-1 accuracy of a network trained for ImageNet classification by 0.28%, while significantly reducing variability across multiple experiment instances.**

Manuscript received 9 September 2023; accepted 27 September 2023. Date of publication 11 October 2023; date of current version 28 November 2023. This work was supported in part by the IBM Research AI Hardware Center; in part by the European Union's Horizon Europe Research and Innovation Program under Grant 101046878 and Grant 101070634; and in part by the Swiss State Secretariat for Education, Research and Innovation (SERI) under Grant 22.00029 and Grant 23.00205. The review of this article was arranged by Editor J. Tang. *(Corresponding author: Athanasios Vasilopoulos.)*

Athanasios Vasilopoulos, Julian Büchel, Benedikt Kersting, Corey Lammie, Manuel Le Gallo, and Abu Sebastian are with IBM Research-Europe, 8803 Rüschlikon, Switzerland (e-mail: atv@zurich.ibm.com; anu@zurich.ibm.com).

Kevin Brew, Samuel Choi, Timothy Philip, and Nicole Saulnier are with IBM Research-Albany, Albany, NY 12203 USA.

Vijay Narayanan is with IBM T. J. Watson Research Center, IBM Research–Yorktown Heights, Yorktown Heights, NY 10598 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TED.2023.3321014.

Digital Object Identifier 10.1109/TED.2023.3321014

***Index Terms*— Analog in-memory computing (AIMC), deep learning (DL) inference, device programming, phase-change memory (PCM).**

I. Introduction

ANALOG in-memory computing (AIMC) has been identified as a viable alternative to the conventional von Neumann computing paradigm for a wide range of applications [1], [2], [3]. The use of synaptic unit-cells comprising memristive devices, organized in a crossbar topology, can be used to perform matrix-vector multiply (MVM) operations in $\mathcal{O}(1)$ time complexity. This is done by encoding weight matrix elements as device conductance values and vector elements as amplitudes or duration of voltage pulses. However, memristive devices typically exhibit significant conductance variations which affect the effective precision of the encoded weights. We use the term conductance variations to capture both the imprecision associated with achieving a target analog conductance value, as well as the subsequent temporal variations of the conductance values. These conductance variations have a detrimental effect on the accuracy of MVM and downstream tasks, such as deep learning (DL) inference.

It is well known that certain conductance states of memristive devices exhibit less variations than others. For example, the SET state of phase-change memory (PCM) devices has substantially less variations than an intermediate state [4]. This is mostly attributed to the presence of crystalline percolation pathways and random telegraph noise in the intermediate states [5], [6]. In resistive random access memory (RRAM), SET states have reduced conductance variations due to the higher number of defects present in the conductive filament, making the path for current conduction well-defined compared with intermediate states [7].

In this article, we propose a device programming approach that exploits the state dependence of conductance variations in scenarios where a synaptic unit-cell comprises of multiple

memristive devices. In such cases, it is possible to optimally encode the synaptic weights in such a way that the conductance states that exhibit minimal conductance variations are maximally used. This simple approach is remarkably effective in improving the MVM accuracy and DL inference accuracy. The rest of the article is structured as follows. In Section II, we present some background material. In Section III, we introduce our proposed programming method, entitled *Max SET Fill (MSF)*, and motivate its conception using experimental data. In Section IV, we provide a comprehensive analysis of the improvement in MVM accuracy achieved via MSF, and in Section V, we present DL inference demonstrations where the synaptic weights are programed using MSF. Finally, in Section VI, the article is concluded.

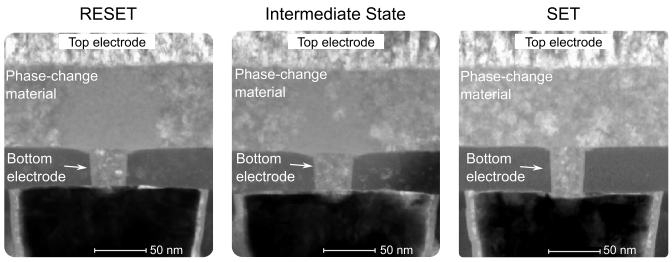

Fig. 1. Low-angle annular darkfield (LAADF) scanning transmission electron microscope (STEM) images of PCM devices in different states. The image of the RESET PCM device shows a large amorphous region, that gets reduced as the device is programed to intermediate, more conductive states. A device in SET state has no distinguishable amorphous region.

II. Background and Related Work

A. Conductance Variations and Mitigation Strategies

In PCM devices, the conductance variations are mostly attributed to the 1/f noise and the intrinsic structural relaxation of the amorphous phase [8]. Much of the recent research efforts on tackling these conductance variations have focused on device-level innovations, such as projected PCM. Such a device comprises a non-insulating material segment parallel to the phase change material segment [9]. However, the advantages of projected PCM are less pronounced in conventional device geometries such as the mushroom-type devices [10].

B. Multi-Memristive Unit-Cells

As conductance values cannot be negative, most unit-cells comprise at least two memristive devices in a differential configuration, where positive and negative weight components are encoded using different devices on separate bitlines (BLs). However, it has been shown that it is advantageous to use multiple devices to encode even the positive or negative weight components [11], [12], [13]. This has led to the adoption of Diff-N unit-cells, which use N devices per weight polarity, comprising a total of $2N$ devices. Furthermore, prior studies suggest that varying the significance of the N devices could lead to improvements in accuracy, by extending the conductance range of the unit-cell, while still allowing precise fine-tuning of its conductance [14]. Although Diff-N unit-cells increase the crossbar area, the impact on energy consumption can be limited if device power consumption is much lower than that of peripheral circuits, which is typically the case in current AIMC chips [15], [16], [17]. However, such unit-cells designs introduce complexity in the programming process, during which synaptic weights are encoded onto the unit-cells. Hence, modifications to well-established programming methods are required to achieve maximum precision.

C. Programming Methods

Programming methods can be usually split into two phases, the first assigning conductance values to devices, referred to as *weight mapping*, and the second, modulating the conductance state of each device to their assigned target conductance value, referred to as *weight programming*. Weight programming is usually performed using an iterative read-write verify (RW-verify) process [18], [19], [20], [21], as single-shot programming cannot be utilized to reliably reach a target conductance value. However, it is possible to use single-shot programming to place a device in either its highest, that is, SET, or lowest, that is, RESET, conductance state. Despite the prominence of the RW-verify paradigm, optimization-based weight programming methods have recently demonstrated increased MVM and downstream DL inference accuracy [22].

The methods associated with the weight mapping phase can be characterized as either static or dynamic, based on whether they use read-out device conductance information from hardware to determine the mapping. Most methods in the literature use static weight mapping methods, in which weight values are scaled to a representative conductance range of the devices [19], [20]. In [11], two such methods are introduced for Diff-N unit-cells, equal-fill (EQF) and max-fill (MF). The former assigns the same conductance to all N devices, whereas the latter starts *filling* each device up to a maximum reliably programmable conductance value, $G_{\max}$, before moving on to the next. $G_{\max}$ has to be selected such that, for most devices in the array, $G_{\max} \leq G_{\text{SET}}$, else it would not be possible to achieve $G_{\max}$ in a significant number of devices. However, this means that most of the devices when programed to $G_{\max}$ will not be in their respective SET states. This drawback associated with static weight mapping methods can be addressed by dynamic mapping methods such as the method reported in [23], entitled weight mapping algorithm + (WMA+). Here, the devices are sequentially initialized with single-shot pulses of varying amplitudes, based on the measured conductance of the already initialized devices. Then, the RW-verify scheme is employed to program one device to adjust the unit-cell conductance to reach a target conductance value. Even though this method considers the read-out device conductance during mapping, it uses a characterized mean programming curve to generate the single-shot pulses, making initialization unpredictable and the method susceptible to convergence failure.

III. Max SET Fill

In this section, we will describe the proposed *MSF* programming method, and motivate the need for it using experimental data from PCM devices, as the representative memristive technology. For the experiments, we employed two PCM -based AIMC cores, each comprising a 256 $\times$ 256 crossbar with Diff-

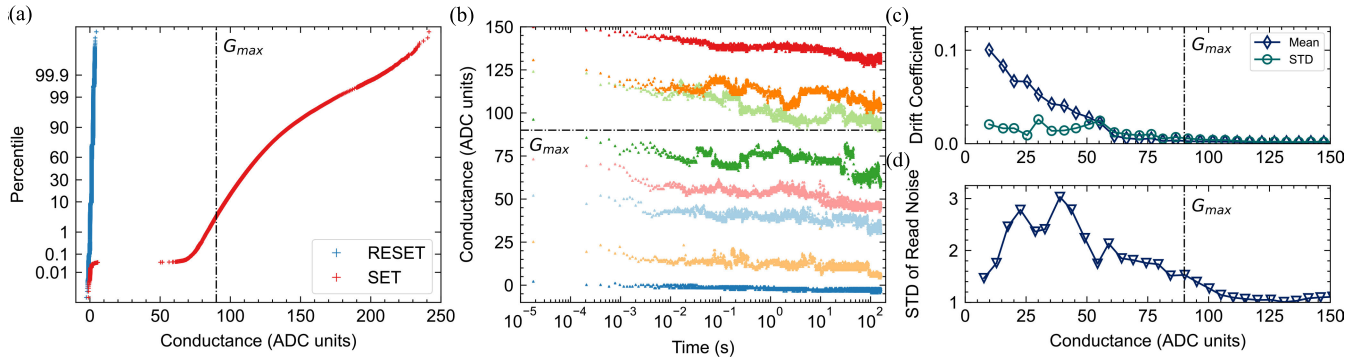

Fig. 2. (a) SET and RESET distributions of a crossbar array with 262k PCM devices. Since the SET state distribution has a large variance, $G_{\max}$ must be selected as a low percentile of the distribution, here fifth, to ensure that most devices can reach it through RW-verify. (b) Temporal evolution of conductance of one sample PCM device, for different initial phase configurations. SET and RESET states are less variable than intermediate states, including the state corresponding to $G_{\max}$. (c) Drift coefficient characterization for 10k devices, as a function of the conductance. High conductance states, and particularly SET states, have less drift variability and drift at a lower rate. (d) Short-term conductance fluctuation, referred to as read noise, characterization for 10k devices as a function of the conductance. SET states exhibit the lowest read noise. **Note:** One ADC unit corresponds to approximately 0.115 μS.

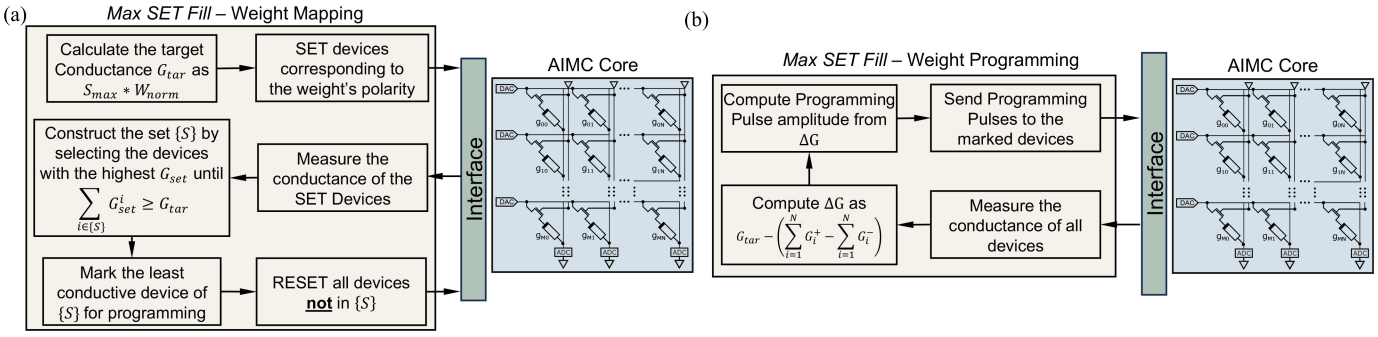

Fig. 3. (a) Flowchart of MSF's weight mapping algorithm. MSF dynamically maps each weight to a unit cell by using the read-out SET conductance of the devices. (b) RW-verify variant used by MSF to program the conductance of the devices. At most one device per unit-cell is being programed to converge to G_{tar}.

2 equal significance unit-cells [17]. The PCM devices are of mushroom-type and have doped $Ge_2Sb_2Te_5$ as the phase-change material.

The conductance of PCM devices can be varied by changing the relative volume of the material in crystalline and amorphous phases, which exhibit high and low conductance, respectively (Fig. 1). PCM devices exhibit temporal conductance drift, in addition to device-to-device and cycle-to-cycle variability [24]. In Fig. 2(a), we show experimentally measured distributions of PCM SET and RESET states across all the PCM devices in one of the AIMC cores. We also show the temporal evolution of conductance of one representative device for different initial conductance in Fig. 2(b). By characterizing the evolution of conductance for 10k devices at different initial phase configurations, we calculated the drift coefficient, which corresponds to the drift rate of a device [Fig. 2(c)], and the read noise, which corresponds to short-term conductance fluctuations [Fig. 2(d)], as functions of the conductance. The SET and RESET states have substantially less read noise, and the SET states less drift, than the intermediate states. Furthermore recent works have shown that SET and RESET states demonstrate preferential behavior in further non-ideal phenomena, like the bipolar current voltage asymmetry [25]. Hence, it would be highly beneficial to maximize their use when encoding the synaptic weights. However, it can be seen that when using static weight mapping methods, $G_{\max}$ ends up being an intermediate state conductance for >95% of devices in the array [see Fig. 2(a)], preventing higher utilization of the more stable SET states. MSF is able to solve this problem using an effective dynamic weight mapping algorithm, that maximizes the use of SET and RESET states.

A flowchart of MSF's weight mapping algorithm is presented in Fig. 3(a). Initially, we use the maximum unit-cell conductance ($S_{\max}$), usually dependent on the weight distribution and the saturation current of the analog to digital converter (ADC), to transform the normalized weight values to the target unit-cell conductance (G_{tar}). Note that $S_{\max}$ is used for the conversion of the weight value to total unit-cell conductance, and not to SD conductance, as the static weight mapping methods do using $G_{\max}$. Next, the N devices corresponding to each weight's polarity are programed to their respective SET states, and their conductance is measured. Then, we define the set $\{S\}$ that contains all devices that will be placed in a non-RESET state. We construct $\{S\}$ by sequentially appending to it the device with the highest SET conductance, until the summed conductance of the devices in $\{S\}$ exceeds G_{tar} or all devices are placed in $\{S\}$. Devices not in $\{S\}$ are RESET, and the least conductive device of $\{S\}$ is marked for further programming such that the unit-cell conductance is reduced

TABLE I

WEIGHT MAPPING COMPARISON OF A UNIT WEIGHT WITH A VALUE OF $W_{NORM} = 0.8$ IN A DIFF-2 UNIT CELL. THE SET CONDUCTANCE OF THE DEVICES ARE $G^1_{SET} = 85$, $G^2_{SET} = 110$. WE CONSIDER $G_{MAX} = 90$, $S_{MAX} = 180$ FOR THE G_{TAR} CONVERSION

Method	G_{tar}	G_1	G_2
SD	72	72	-
EQF	144	72	72
WMA+	144	$Z^\dagger$	Y^*
MF	144	90 ($> G^1_{set}$)	54
MSF	144	34	110 ($= G^2_{set}$)

* Y equals to the conductance after an SSP to reach $G_{tar} - G^1_{set}$.
† $Z = G^1_{set} - (G_{tar} - G^1_{set} - Y)$.

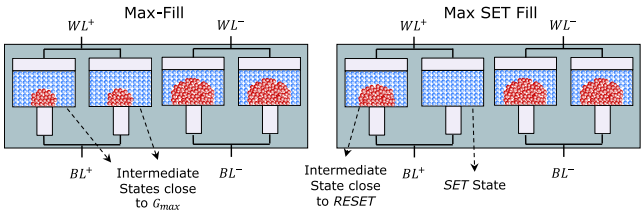

Fig. 4. Visual depiction of the weight mapping example of Table I. MSF attempts to maximize the number of devices in the SET and RESET states, with at most one device in an intermediate state. Static weight mapping methods that map weight magnitudes from 0 to G_{max}, including MF, result in placing multiple devices in noisy intermediate states.

to the target value. In our experiments, we use the RW-verify method for the programming of the marked device [Fig. 3(b)], but any weight programming method can be employed. During the RW-verify procedure, all devices of the unit-cell are read to calculate the conductance update, as an elementary drift compensation mechanism, since the short-term drift of the SET devices will be compensated by increasing the target conductance of the device being programed. If following the SET initialization, the summed conductance of all N devices is smaller than the target conductance value, the unit-cell cannot accommodate the assigned weight. Hence, it is left in the full SET state to minimize the conductance error. In Table I, we overview how methods from the literature would map an indicative weight, compared to MSF. As Fig. 4 shows, MSF is the only method that benefits from the stability of the SET state, which results in the unit-cell being less affected by conductance variations than for methods using static weight mapping, such as MF.

The key advantage of MSF is that it optimally maps a weight to each unit-cell, by selecting the smallest number of devices needed to accommodate it, through the use of true SET states. This guarantees that at most one device per unit-cell will be at an intermediate state, thus reducing the overall conductance variability of the unit-cell. Furthermore, since single-shot pulses can be used for SET and RESET programming, only one device needs to be programed with a more time consuming iterative RW-verify scheme, hence reducing the total programming cost compared with other methods. Overall, the time overhead and memory requirements of MSF are bound by the selected weight programming scheme, as MSF's overhead in the weight mapping phase is not substantial.

IV. MATRIX VECTOR MULTIPLICATION PRECISION

In this section, we will compare the aforementioned programming approaches in terms of the MVM precision. First, for each method, we program the same weight array $\mathbf{W}$, and perform numerous MVM operations with the same batch of input vectors, $\mathbf{x}_i$ for i from 1 to B. The weights and inputs are drawn from a sparse uniform distribution. Then, for each MVM, we compute the error ϵ_{MVM} as follows:

$$\epsilon_{\text{MVM}} = \frac{\|\mathbf{y}_i - \tilde{\mathbf{y}}_i\|_2}{\|\mathbf{y}_i\|_2} \tag{1}$$

where $\mathbf{y}_i = \mathbf{W}\mathbf{x}_i$ is the floating-point result of the operation and $\tilde{\mathbf{y}}_i$ denotes the measured ON-chip result. The MVM error results are shown in Fig. 5(a). It is observed that any multi-device programming method is more accurate than SD programming, demonstrating that multi-memristive unit-cells increase accuracy. Using two devices, even at the same target state (EQF), improves performance due to an averaging effect of the device noise, also observed in [11] and [12]. The results also indicate that accuracy can be further increased if one takes in consideration the device characteristics. WMA+, which dynamically maps the weight based on device reads, performs better than EQF, while MF enhances accuracy further by giving higher conductance states priority. MSF outperforms all these methods noticeably, due to its use of stable SET and RESET states. All methods achieve better MVM accuracy compared to a digital system performing the MVM with 8-bit input-output and 3-bit weights. MSF improves the MVM accuracy such that it is closer to 4-bit weights.

Further insights can be gained if we examine how accurately each approach programs the target weight array on the crossbar. An estimate of the programed weights, $\hat{W}$, can be inferred through MVM operations [26] by solving the following equation:

$$\hat{W} = \arg\min_{\hat{W}} \sum_{i=1}^{B} \|\hat{W}\mathbf{x}_i - \tilde{\mathbf{y}}_i\|_2. \tag{2}$$

Fig. 5(b) depicts the weight error, defined as the standard deviation of $\mathbf{W} - \hat{W}$, as a function of the weight values. SD and EQF's weight error characteristics have an identical shape, because they map the devices to the same conductance states, with their magnitude differing roughly by a factor of $1/\sqrt{2}$, further indicating that adding devices to a unit-cell averages the device noise. WMA+'s accuracy in the higher weight range degrades, due to the method's tendency to undershoot high weight values [23]. MF begins outperforming EQF just before the weight range's midpoint, which corresponds to G_{max}, that is, the method's "fill" limit. This demonstrates that programming one device to a stable state is preferable to distributing the weight between two devices in noisier states [Fig. 2(d)]. The methods converge at the limit of the weight range, where both methods map weights in a similar fashion. MSF 's weight error is smaller than MF's when the corresponding normalized weight magnitude is larger than 0.5. Additionally, a noticeable drop and overall lower error is observed in the high weight range ($\gg$0.5). MSF is the only method that does not demonstrate a monotonic increase of

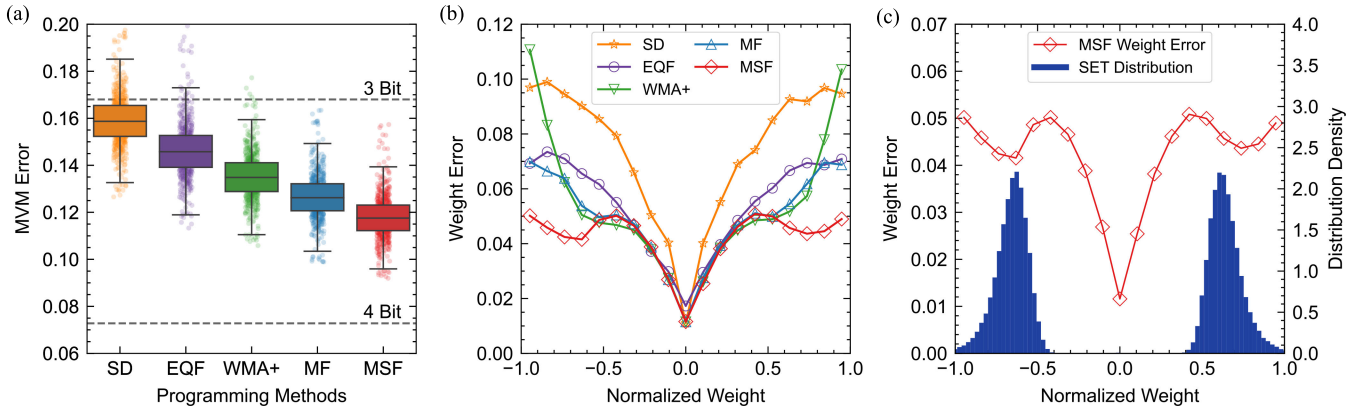

Fig. 5. (a) MVM error (ϵ_{MVM}) comparison of all methods, as determined by (1). MSF outperforms the rest of the methods, due to its use of true SET states. All methods achieve accuracy between that of digital systems performing MVM with 3 and 4 bit weights. (b) Measured weight error after programming as a function of the normalized weight values. (c) Depiction of the weight error of MSF overlaid with the SET distribution normalized to the corresponding weight range. The peak of the SET distribution aligns with the dip of the weight error in the high weight range, showing that the use of SET states indeed lowers the weight error.

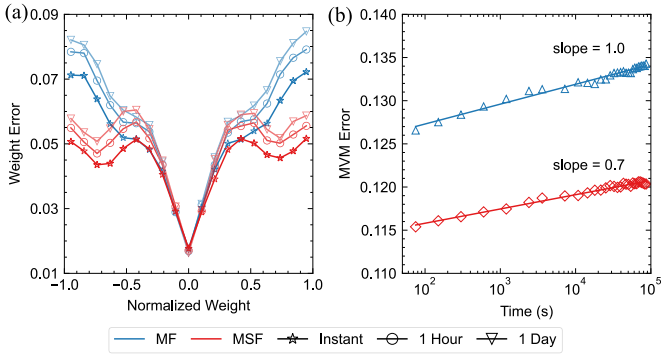

Fig. 6. (a) Evolution of the weight error at three distinct points of time during a 24-h period. MSF's weight error evolves at a slower rate as SET states drift at a slower rate than intermediate states. (b) Mean MVM error increase through the 24-h period. MSF exhibits better accuracy retention.

weight error in the intermediate weight range and it is the result of exploiting the stable SET states. Overlaying the SET distribution of the devices with MSF's weight error [Fig. 5(c)] shows that the point at which the weight error starts reducing coincides with the start of the SET distribution, and the local minimum matches the distribution's peak.

As PCM exhibits conductance drift, examining whether MSF's superior accuracy is retained over time is crucial for downstream tasks. We only compare MF and MSF, as the previous results suggest that they perform the best. We perform MVM throughout a 24-h period and record the evolution of weight and MVM error (Fig. 6). MSF exhibits greater accuracy retention over that time, as SET states drift less than intermediate states [Fig. 2(c)].

After establishing the accuracy of our method for a Diff-2 equal significance unit-cell, we investigate its performance in larger and varying significance unit-cells. To facilitate these experiments with our hardware, we emulate varying significance Diff-N unit-cells, by grouping N unit-cells of neighboring BLs. Furthermore, by assigning a significance factor α to each BL, we are able to select a significance factor for each device independently [Fig. 7(a)]. For these experiments, we calculate the conductance update during the iterative RW-verify procedure of MSF by reading only the device being programed instead of the whole unit-cell as done previously. With the latter approach, we found that devices of large significance drifting at a rapid rate often cause the device of lower significance being iteratively programed to not converge, making it unsuitable for this use-case. We first examine the accuracy of the methods for a Diff-2 unit-cell with equal ($\alpha_0 = \alpha_1 = 1$) and 2-exponent ($\alpha_0 = 1$, $\alpha_1 = 2$) significance. In Fig. 7(b), we measure that the equal significance configuration outperforms the 2-exponent. Furthermore, MSF outperforms MF in both cases. The precision of the 2-exponent configuration is reduced, as the MVM error is dominated by the conductance fluctuations of the most significant device. The assignment of different significance to the devices seems to hinder precision, as it counteracts the averaging effect observed in the previous experiments.

Expanding the unit-cells to Diff-4, we get the chance to explore more combinations of significance factors. In addition to equal and 2-exponent ($\alpha_i = 2^i$) significance, we explore three more configurations with a number of devices acting as the MSB ($\alpha = 2$) and LSB ($\alpha = 1$). Amongst all compared configurations, equal significance was found again to be the best approach [Fig. 7(c)]. The 3MSB 1LSB configuration performs the best from the varying significance configurations. Finally, MSF outperforms MF in all configurations explored here.

V. DL Inference

In this section, we examine the impact of MSF in DL inference tasks. First, we compare the classification accuracy of a network with the LeNet-5 architecture [Fig. 8(a)] using the MNIST dataset [27]. All MVM were performed on-chip [17] [Fig. 8(b)], using Diff-2 unit-cells, while the rest of the operations were performed in software at floating-point precision. Fig. 8(c) indicates that MSF lowers the MVM error on every layer, which leads to 0.71% improvement in the mean inference accuracy. It can also be seen that the variance across multiple inference runs is also significantly reduced over MF [Fig. 8(d)].

To determine the efficacy of our method for larger networks, we evaluate the top-1 classification accuracy on ImageNet

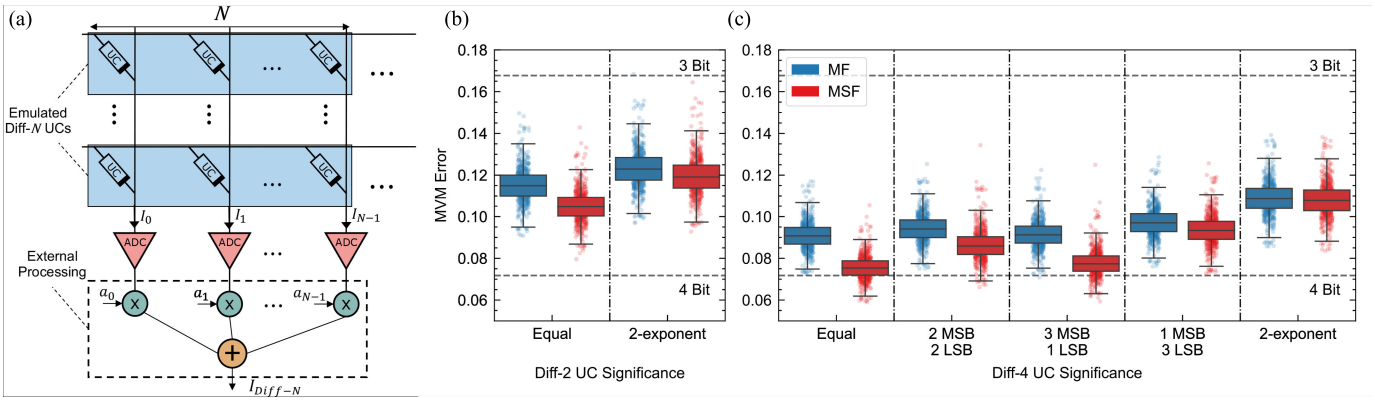

Fig. 7. (a) Emulation of a Diff-*N* varying significance unit-cell (UC) using *N* neighboring BLs of [17]. The significance factors are applied external to the crossbar. (b) MVM accuracy of a Diff-2 UC of equal and 2-exponent significance. MSF with an equal significance cell is the best performing combination. Note that the resulting MVM error is lower than for the experiments shown in Fig. 5 due to using two ADCs instead of one to measure the same effective current. (c) MVM accuracy of a Diff-4 UC with different significance configurations. MSF with equal significance UC is the best performing combination, and the most accurate varying significance configuration is the 3MSB and 1LSB UC.

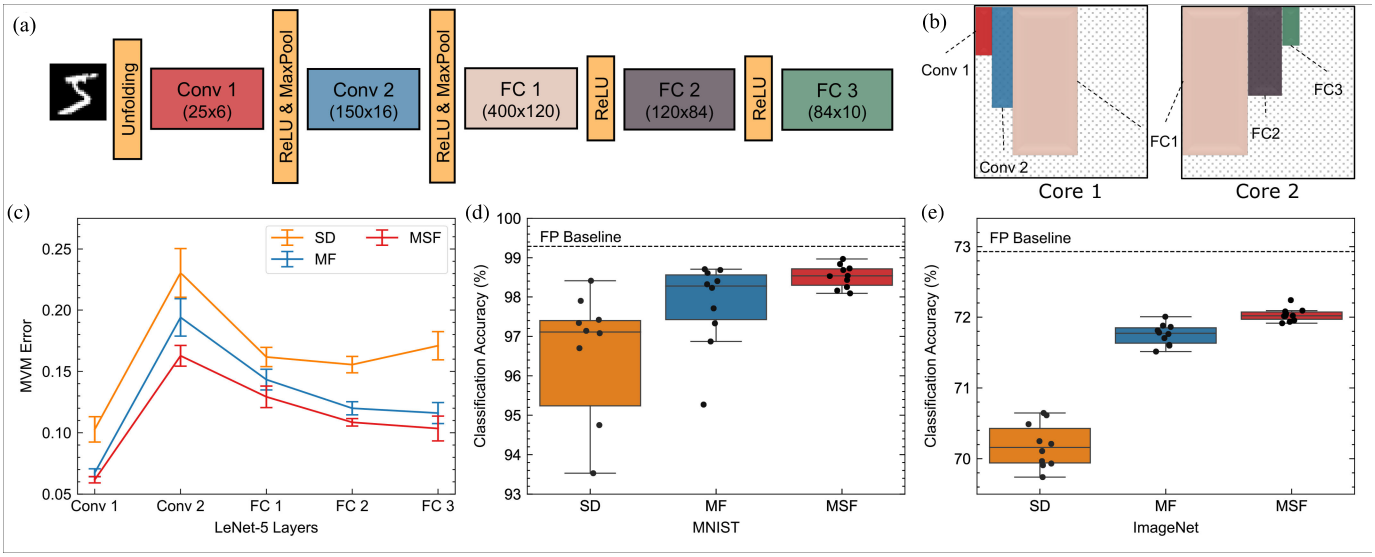

Fig. 8. (a) LeNet-5 architecture for image classification using the MNIST dataset. (b) LeNet-5 layers mapped onto two AIMC cores of the experimental platform [17]. (c) Per-layer MVM error when performing inference on the MNIST dataset, for SD, MF, and MSF. MSF lowers the MVM error on all layers. (d) MSF's lower MVM error leads to higher mean inference accuracy and reduced variance (ten inference runs). (e) Hardware-realistic simulation results of the classification accuracy on the ImageNet dataset (ten inference runs). MSF is able to improve accuracy and reduce variance in large networks as well.

using the ResNet-34 architecture [28]. As this network is too large to fit on our current hardware, we perform simulations using statistical models extracted from the chip measurements presented above. Our simulations account for the crossbar input–output quantization of our hardware and the weight noise of each method, as shown in Fig. 5(b). The hardware-realistic simulation results in Fig. 8(e) show that MSF outperforms MF by 0.28%, and reduces the variance, despite the fact that larger networks are more resilient to device noise [29].

VI. Conclusion

Mitigating the effect of conductance variations is important to achieve sufficient compute precision using AIMC hardware. In this article, we presented a novel programming method for AIMC cores with multi-memristive unit-cells that exploit the state dependency of conductance variations to improve computational precision. This is achieved by maximizing the number of devices in SET and RESET states when programming the unit-cells. Using PCM -based AIMC cores, we experimentally demonstrated that MSF achieves lower MVM error and better retention over time, reducing the accuracy gap to 4-bit compute precision, when two devices are used to encode each weight. Furthermore, we showed that our approach can be expanded to *N*-device unit-cells, while retaining its accuracy advantage. Finally, we demonstrated significant improvements in neural network inference accuracy for downstream MNIST and ImageNet classification tasks. It is noted that MSF can be improved with the use of programming methods that outperform the iterative RW-verify [22] scheme. Even though we used PCM device technology to demonstrate the efficacy of MSF, this approach is equally applicable to other memristive devices exhibiting similar state-dependent conductance variations [7]. Moreover, MSF could initiate

new research directions that aim at improving the stability of specific conductance states as opposed to tackling the more challenging task of reducing the overall conductance variations.

REFERENCES

[1] A. Sebastian, M. L. Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnol.*, vol. 15, no. 7, pp. 529–544, Jul. 2020.

[2] M. Lanza et al., "Memristive technologies for data storage, computation, encryption, and radio-frequency communication," *Science*, vol. 376, no. 6597, Jun. 2022, Art. no. eabj9979.

[3] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits Syst. Mag.*, vol. 21, no. 3, pp. 31–56, 3rd Quart., 2021.

[4] S. R. Nandakumar et al., "Precision of synaptic weights programmed in phase-change memory devices for deep learning inference," in *IEDM Tech. Dig.*, Dec. 2020, pp. 29.4.1–29.4.4.

[5] J. Liu, "Microscopic origin of electron transport properties and ultrascalability of amorphous phase change material germanium telluride," *IEEE Trans. Electron Devices*, vol. 64, no. 5, pp. 2207–2215, May 2017.

[6] D. Fugazza, D. Ielmini, S. Lavizzari, and A. L. Lacaita, "Distributed-Poole–Frenkel modeling of anomalous resistance scaling and fluctuations in phase-change memory (PCM) devices," in *IEDM Tech. Dig.*, Dec. 2009, pp. 1–4.

[7] A. Prakash and H. Hwang, "Multilevel cell storage and resistance variability in resistive random access memory," *Phys. Sci. Rev.*, vol. 1, no. 6, Jun. 2016, Art. no. 20160010.

[8] M. Le Gallo and A. Sebastian, "An overview of phase-change memory device physics," *J. Phys. D, Appl. Phys.*, vol. 53, no. 21, Mar. 2020, Art. no. 213002.

[9] W. W. Koelmans, A. Sebastian, V. P. Jonnalagadda, D. Krebs, L. Dellmann, and E. Eleftheriou, "Projected phase-change memory devices," *Nature Commun.*, vol. 6, no. 1, p. 8181, Sep. 2015.

[10] S. Ghazi Sarwat et al., "Projected mushroom type phase-change memory," *Adv. Funct. Mater.*, vol. 31, no. 49, Sep. 2021, Art. no. 2106547.

[11] M. Le Gallo et al., "Precision of bit slicing with in-memory computing based on analog phase-change memory crossbars," *Neuromorphic Comput. Eng.*, vol. 2, no. 1, Feb. 2022, Art. no. 014009.

[12] G. Pedretti, E. Ambrosi, and D. Ielmini, "Conductance variations and their impact on the precision of in-memory computing with resistive switching memory (RRAM)," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2021, pp. 1–8.

[13] I. Boybat et al., "Neuromorphic computing with multi-memristive synapses," *Nature Commun.*, vol. 9, no. 1, p. 2514, Jun. 2018.

[14] C. Mackin et al., "Optimised weight programming for analogue memory-based deep neural networks," *Nature Commun.*, vol. 13, no. 1, p. 3765, Jun. 2022.

[15] W. Wan et al., "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, Aug. 2022.

[16] S. Yin, X. Sun, S. Yu, and J.-S. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90-nm CMOS," *IEEE Trans. Electron Devices*, vol. 67, no. 10, pp. 4185–4192, Oct. 2020.

[17] R. Khaddam-Aljameh et al., "HERMES-core—A 1.59-TOPS/mm^2 PCM on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs," *IEEE J. Solid-State Circuits*, vol. 57, no. 4, pp. 1027–1038, Apr. 2022.

[18] N. Papandreou et al., "Programming algorithms for multilevel phase-change memory," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2011, pp. 329–332.

[19] Y. Luo, X. Han, Z. Ye, H. Barnaby, J.-S. Seo, and S. Yu, "Array-level programming of 3-bit per cell resistive memory and its application for deep neural network inference," *IEEE Trans. Electron Devices*, vol. 67, no. 11, pp. 4621–4625, Nov. 2020.

[20] V. Milo et al., "Accurate program/verify schemes of resistive switching memory (RRAM) for in-memory neural network circuits," *IEEE Trans. Electron Devices*, vol. 68, no. 8, pp. 3832–3837, Aug. 2021.

[21] B. Q. Le et al., "RADAR: A fast and energy-efficient programming technique for multiple bits-per-cell RRAM arrays," *IEEE Trans. Electron Devices*, vol. 68, no. 9, pp. 4397–4403, Sep. 2021.

[22] J. Büchel et al., "Gradient descent-based programming of analog in-memory computing cores," in *IEDM Tech. Dig.*, Dec. 2022, pp. 33.1.1–33.1.4.

[23] M. Martemucci et al., "Accurate weight mapping in a multi-memristive synaptic unit," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[24] A. Pirovano et al., "Reliability study of phase-change nonvolatile memories," *IEEE Trans. Device Mater. Rel.*, vol. 4, no. 3, pp. 422–427, Sep. 2004.

[25] S. G. Sarwat et al., "Mechanism and impact of bipolar current voltage asymmetry in computational phase-change memory," *Adv. Mater.*, vol. 35, no. 37, May 2022, Art. no. 2201238.

[26] M. L. Gallo et al., "A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference," *Nature Electron.*, vol. 6, no. 9, pp. 680–693, Aug. 2023.

[27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Dec. 1998.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[29] M. J. Rasch et al., "Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators," *Nature Commun.*, vol. 14, no. 1, p. 5282, Aug. 2023.