On the Reliability of RRAM-Based Neural Networks

Aziza, Hassen; Zambelli, Cristian ; Hamdioui, Said; Diware, Sumit; Bishnoi, Rajendra; Gebregiorgis, Anteneh

DOI
10.1109/VLSI-SoC57769.2023.10321859

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC)

On the Reliability of RRAM-Based Neural Networks

Hassen AZIZA* Cristian Zambelli† Said Hamdioui‡ Sumit Diware‡ Rajendra Bishnoi‡ Anteneh Gebregiorgis‡
*IM2NP, UMR CNRS 7334, Aix-Marseille Université, Marseille, France. Email: hassen.aziza@univ-amu.fr
†Dip. Ingegneria, Università degli Studi di Ferrara, Ferrara, Italy. Email: cristian.zambelli@unife.it
‡Computer Engineering Lab, Delft University of Technology, Delft, The Netherlands.
Email: {S.Hamdioui, S.S.Diware, R.K.Bishnoi, A.B.Gebregiorgis}@tudelft.nl

2023 IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC) | 979-8-3503-2599-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/VLSI-SoC57769.2023.10321859

***Abstract*—Emerging device technologies such as Resistive RAMs (RRAMs) are under investigation by many researchers and semiconductor companies; not only to realize e.g., embedded non-volatile memories, but also to enable energy-efficient computing making use of new data processing paradigms such as computation-in-memory. However, such devices suffer from various non-idealities and reliability failure mechanisms (e.g., variability, endurance, and retention); these negatively impact the memory robustness and the computation accuracy. This paper discusses the non-idealities and reliability failure mechanisms for RRAM devices, provides an overview on the most popular ones. In addition, it reports detailed anlysis of some of these based on data measurements. Finally, it presents two different mitigation schemes for RRAM based accelerators; one is based on RRAM non-ideality aware quantization and conductance control for neural network accuracy enhancement while the second is based on reliability-aware biased training technique.**

***Index Terms*—RRAM, reliability, neural network, in-memory computing**

I. Introduction

Emerging IoT-edge applications are extremely demanding in terms of storage, computing power, and energy efficiency to enable the deployment of AI (Artificial Intelligence). On the other hand, both today's computer architectures and device technologies are facing major challenges making them incapable of delivering the required functionalities and features at economical affordable cost [1], [2]. For computing systems to continue delivering sustainable benefits for the foreseeable future in society, where energy budgets are tight, alternative computing architectures (such as analog computation-in-memory) that leverage novel post-CMOS device technologies (such as Resistive RAMs) are being explored, while incorporating radically new concepts such as brain-inspired concepts [3], [4]. Although these alternative architectures seem to be extremely promising [5]–[7], there is a fundamental issue that without solving it, such alternative computing paradigms will be worthless; i.e., dealing with post-CMOS device non-idealities and reliability failure mechanisms from which such storage/computing devices (inherently) suffer [8], [9].

Research in the reliability failure mechanism of RRAM devices, their impact on RRAM-based memories and computing accelerators, and the way to mitigate these are still in the early stages. Many reliability challenges of RRAM device technology were reported in the community. Some of them are quite popular (and are also applicable for other non-volatile technologies) such as endurance, retention, parasitics, and R/W disturb [10], [11], while others are unique to RRAM and/or to the way they manifest themselves such as variability [12], and intermittent faults [13]. Nevertheless, it is quite difficult to know if all the reliability mechanisms of RRAM are also known, as the technology is still evolving. To address these challenges, mitigation techniques are being explored both for RRAM-based memories as well as for RRAM-based accelerators (computing). Schemes related to reliability improvement for memory systems include redundancy, programming signals control, error correction techniques [14], on-line testing and repairing schemes [15], and error detection and mitigation schemes [16]. On the other hand, robustness improvement and reliability failures mitigation schemes targeting RRAM accelerators are an ongoing process that involves a combination of hardware and software measures. Moreover, the choice of the optimal reliability mitigation strategy is dictated by the specific requirements and criticality of the targeted application. Several software and hardware solutions have been proposed to mitigate the impact of non-idealities on the computation accuracy of Compute In-Memory (CIM) blocks [17]–[24]. Some of the software-based solutions focus on finding an optimal mapping where less relevant values (e.g., LSB) are mapped to the non-ideal memristor devices [17], [21], [23], while others focus on retraining techniques to partially regain the non-ideality induced accuracy reduction [18], [22], [24]. Similarly, the hardware-based solutions utilize redundancy schemes to tackle the impact of non-idealities [19], [20]. Indeed, RRAM devices suffer from many issues potentially impairing the operation of the applications relying on massive CIM acceleration. The most challenging reliability threats are represented by the conductance relaxation in time (i.e., drift), the cycle-to-cycle (C2C), and the device-to-device (D2D) variabilities [25]–[27]. Although mitigations are proposed, most of them are inefficient and impose several hardware overheads.

This work was supported in part by the EU H2020 grant "DAIS" that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No. 101007273 and by the Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 - Next Generation EU (NGEU).

Thus, understanding the nature of memristor non-idealities and providing efficient solutions is essential to realize reliable and efficient CIM operations.

This paper provides an overview of RRAM device reliability failure mechanisms and shows the impact and potential mitigation schemes of some of these both on memory and computing elements based on RRAM devices. Section 2 classifies these reliability mechanisms. Section 3 discusses the impact of variability and intermittent faults in RRAM based on measured data, and shows how these can influence some of the KPI (key performance indicators) of RRAM-based systems. Section 4 discusses how to deal with drift and variability when RRAMs are used as a key element in crossbar neural networks. Section 5 presents a memristor reliability-aware biased training technique to mitigate the impact of non-ideality on the network inference accuracy, while Section 6 concludes the paper and gives some future directions.

II. Classification of RRAM reliability failure mechanisms

This section provides first a classification of the RRAM reliability metrics, which are considered of interest when using RRAM devices as key elements for the realization of memory systems (storage) or accelerators (computing). Thereafter, each class will be discussed separately.

A. Classification of key reliability metrics

RRAM devices may suffer from reliability concerns either due to immature manufacturing processes and/or due to the inherent nature and properties of the device. This may impact the technology parameters of the device (e.g., the length and/or the width of the filament), which in turn impacts the electrical parameters of the device such as the value of the resistance of the device in ON state (or LRS for Low Resistance State) and OFF state (or HRS for High Resistance State). The way these non-idealities impact the functionality depends on the way the RRAM devices are used. For example, an RRAM device used to store the two binary values 0 and 1 will not much suffer from variability. However, when these devices are used for analog computing in a neural network context (where devices are used as synaptic weights), such variability has a large impact on the accuracy of computing as such computing is based on weighted sums of currents.

Next, we give a classification of the main RRAM device reliability metrics into *time-zero* reliability metrics and *time-dependant* reliability metrics [28]; the overview is presented in Fig. 1.

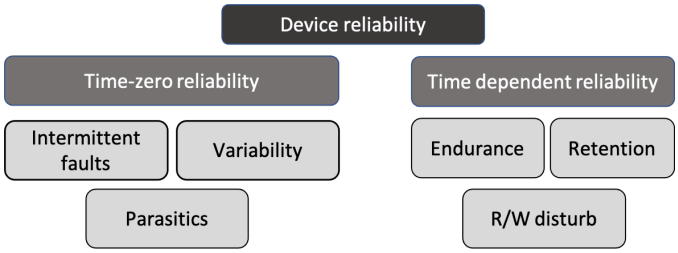


Fig. 1. Classification of RRAM reliability failure mechanisms.

Time-zero reliability mechanisms: Time-zero reliability mechanisms are mechanisms that occur at time $t = 0$. These are caused by imperfections in the fabrication process. Due to these varying conditions during fabrication, the specifications of the chip differ from the intended ones. The main time-zero mechanisms are:

- Variability [12]: refers to the variations that can occur in RRAM electrical characteristics (due to process manufacturing variability), including RRAM conductance levels. It includes Device-to-Device (D2D) and Cycle-to-Cycle (C2C) variability [25]. Variability directly impacts the accuracy of RRAM-based analog computing.
- Intermittent faults [13]: refers to random changes in the resistance of an RRAM cell that occur when write operations are applied to fresh "formed" devices. E.g., the RRAM can get unpredictably stuck to a low ohmic state after SET or a high ohmic state after RESET; the device recovers to normal behavior in the following cycles. The intermittent behavior may last one or a couple of cycles.
- Wire parasitics [29]: due to the finite parasitic resistance and capacitance of the interconnect wires, signals suffer from delay mismatch and voltage degradation, which can lead to erroneous outputs [29]. For instance, in logic operations, the reference and input signals reaching the sensing circuits (e.g., sense amplifier) suffer from delay mismatch caused by different critical paths. Additionally, the wordline degrades along the path reaching farther columns and degrading the associated current output.

Time dependent reliability mechanisms: Time-dependent mechanisms include those that occur during the operational lifetime (i.e., $t > 0$) of the device; they may be intrinsic (caused by mechanisms that occur inside the device itself) or extrinsic (caused by (external) environmental factors). The main time-dependent mechanisms are:

- Endurance [30]: RRAMs suffer from limited endurance due to the destructive nature of the programming operations (i.e., SET followed by a RESET operation or vise versa). The number of times an RRAM cell can be cycled before it fails is known as endurance. The endurance of RRAM cells is typically measured in millions or even billions of cycles, which is compatible with synaptic weight update operations occurring during the learning process of many neuromorphic applications.
- Retention [30]: the ability of the RRAM device to maintain its resistance state over time. The retention requirement for RRAM in neuromorphic applications is typically in the range of years to decades to emulate the persistent synaptic strengths found in biological synapses. In other words, the resistance states should be stable over a long period to avoid information loss.
- R/W disturb [31]: is a phenomenon that arises when reading a specific RRAM cell multiple times in succession. The addressed cell can experience unintended resistance changes due to the stress induced by the read

operation. Read disturb is an important reliability issue as it limits performances of read-intensive applications generally associated with deep neural networks (DNN) where synaptic weights are constantly and simultaneously read during inference. Write (W) disturb occurs when a neighboring memory cell is affected or altered during a write operation in a specific memory cell, leading to inadvertent resistance changes.

The above reliability mechanisms may impact the functionality of RRAM-based memory or RRAM-based accelerator differently. Moreover, some parameters can have a beneficial effect on storage but a negative effect on computing. For instance, the nonlinearity of RRAM devices [25] is advantageous for memory applications where a strict threshold between LRS and HRS states is needed but critical for computing applications. Indeed, to reach a high learning accuracy in neural networks based on the backpropagation learning rule, a linear weight update behavior of the analog RRAMs is a critical requirement [32]. Otherwise, it can result in inconsistent conductance changes that can make synaptic weight tuning difficult, affecting the convergence of training algorithms.

III. Analysis of RRAM Reliability Failure Mechanisms Based on Measurement Data

Two major reliability metrics will be analyzed in this section: variability and intermittent faults. These two metrics lead to a direct narrowing of the RRAM resistance window, hence, reducing significantly the conductance modulation capability of the technology. On the one hand and as already mentioned, variability prevents a strict control of RRAM conductance, hence, posing a significant hurdle for RRAM widespread adoption. On the other hand, in addition to variability, intermittent errors lead to important and sudden fluctuations in the RRAM conductance levels.

A. Variability: C2C and D2D

Variability refers to the inherent differences in the electrical characteristics of RRAM devices, even when fabricated under the same manufacturing conditions. It includes D2D and C2C variability. Variability can manifest as variations in the key parameters of RRAM devices such as switching voltages and resistance levels, thus, preventing reliable and predictable device behavior [33]. For instance, neuromorphic computing systems rely on training algorithms to adjust the synaptic weights and optimize the network performances. However, the variability in RRAM devices can affect the convergence of these algorithms, turning the learning stage challenging, although techniques such as adaptive learning algorithms and training strategies that account for device variability can help address this limitation [34]. Temporal and spatial resistance variations in both LRS and HRS states are so erratic that RRAM technology has been employed as an entropy source in True Random Number Generators (TRNG) [35], [36]. Hence, the successful implementation of a reliable conductance modulation scheme in a computing context mainly depends on the ability to control the impact of variability on the different conductance levels. At the cell level, the conductance modulation can be transposed to a segmentation of the I-V plane by different I-V characteristics. For simplicity, only 8 I-V characteristics are presented in Fig. 2. Each characteristic has a slope of G_X equal to $1/R_X$, where X represents the number of resistance states ranging from *0* to *n*. To take into account the variability of the n HRS/LRS resistance states, the conductance margin is highlighted in Fig. 2 by a shaded strip encompassing each I-V characteristic. Also, e can see that a reliable conductance modulation strategy does not just depend on the programming operation, but also depends on an accurate read mechanism as a read operation is conducted by applying a low bit line voltage bias V_{READ} across the RRAM cell and sensing the resulting currents. The latter, referred to as I_{Read_0} to I_{Read_n-1}, reflecs the RRAM conductance values.

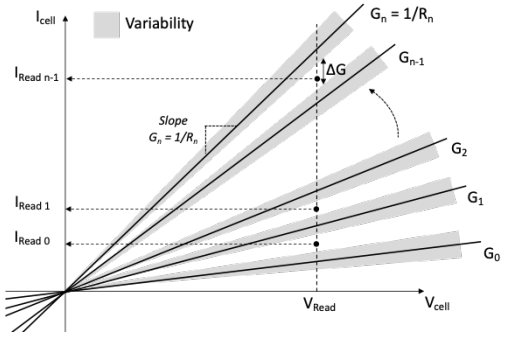


Fig. 2. I-V plane segmentation after a programming operation. Variability is illustrated by a shaded area encompassing each I-V characteristic. Adapted with permission from [37] under Creative Commons License (CC BY 4.0).

The impact of C2C and D2D variability is experimentally revealed at the I-V characteristic level after a RESET/SET operation applied to each cell of an elementary memory array [38] to catch C2C variability (Fig. 3a) and after a RESET/SET operation applied to an isolated cell of the same array to catch D2D variability (Fig. 3b). Although only two resistance levels are considered, HRS and LRS resistance values are both affected by C2C and D2D variability, with a more marked impact in the HRS state, particularly when D2D variability is concerned.

B. Intermittent errors

RRAM technology suffers from intermittent [13] and hard errors [30]. Both errors directly impact the RRAM resistance window. Hard errors are a permanent corruption of a memory cell resulting from physical defects usually activated after

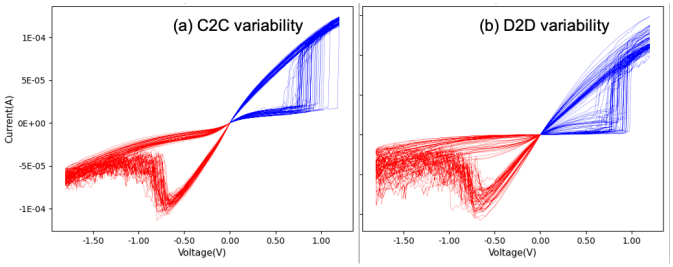


Fig. 3. RRAM I-V characteristics highlighting (a) C2C and (b) D2D variability.

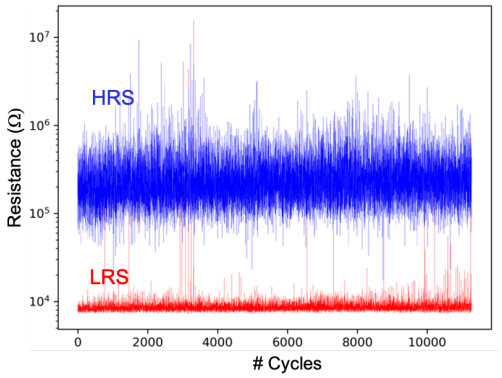


Fig. 4. HRS/LRS resistances in log scale over 11,269 cycles.

cycling or retention tests, while intermittent errors are related to a random, recoverable upsetting of the information stored in a memory cell [39]. This drawback of the technology results from the stochastic dynamics of resistive switching (i.e., oxygen vacancies/ions motion). Fig. 4 presents the evolution of HRS and LRS extracted at 0.1 V versus the number of programming cycles. Each programming cycle ranges from 0 to 11,269. When analyzing Fig. 4, many HRS and LRS level overlaps are visible. These overlaps happen either because one cell stops meeting its nominal characteristics or because one cell is stuck at a resistance state for one or several cycles before recovering. However, no specific pattern has been detected, demonstrating the unpredictable nature of the RRAM technology.

These overlaps can be explained from a physical standpoint. If during a SET operation, too many bonds between some of the metal and oxygen ions in the oxide layer break, too many oxygen ions are attracted to the top electrode hence leaving oxygen vacancies acting as a conductive filament (CF). Therefore, the voltage applied across the cell to induce an effective RESET operation might not be high enough. In such a case, switching the cell may demand several programming cycles to get those oxygen ions to drift back. Conversely, if during a RESET operation, too many ions are displaced, the voltage needed to induce an efficient SET operation in the next programming cycle may not be sufficient, leading the device to be stuck at HRS for several cycles. To mitigate intermittent errors impact on memristive networks, it is necessary to design specific circuits able to monitor the effectiveness of each RESET/SET operation [40].

IV. RRAM-based Neural Networks Reliability

DNNs are one of the most essential tools in AI for a plethora of applications, such as object detection, natural language, and text processing [41]. Reaching a high accuracy comes at the cost of increased computational complexity and model size, jeopardizing their implementation on traditional Von Neumann architectures due to the large energy consumption required in the communication between processing and off-chip memory elements [42]. To overcome this bottleneck, the CIM paradigm allows the processing of the data in-situ, proving to be an energy-efficient solution for DNN hardware acceleration. RRAM devices have emerged as the ideal candidate for CIM because of their non-volatile characteristics, conductance tuning for DNN synaptic weights representation, and low power consumption [43]. RRAM devices (usually connected to a select transistor as in 1T-1R structures) can be organized in a crossbar array topology (see Fig. 5a), that can perform analog matrix-vector multiplication (MVM) in one computational step [44], thanks to the inherent parallelism. This feature is of help in the hardware acceleration of a staple operation in DNNs such as the Multiply and Cumulate (MAC). Additionally, RRAM crossbars can be tiled and interconnected with ADCs, DACs, and DSPs to implement a complete DNN (see Fig. 5b). In this section, we present the most leveraged spanning from device optimizations to the advanced circuit and algorithmic strategy [45]. The reminder of this section presents different techniques to deal with impact of RRAM reliability failure mecahnisms on DNNs mapped to an RRAM-based CIM crossbar.

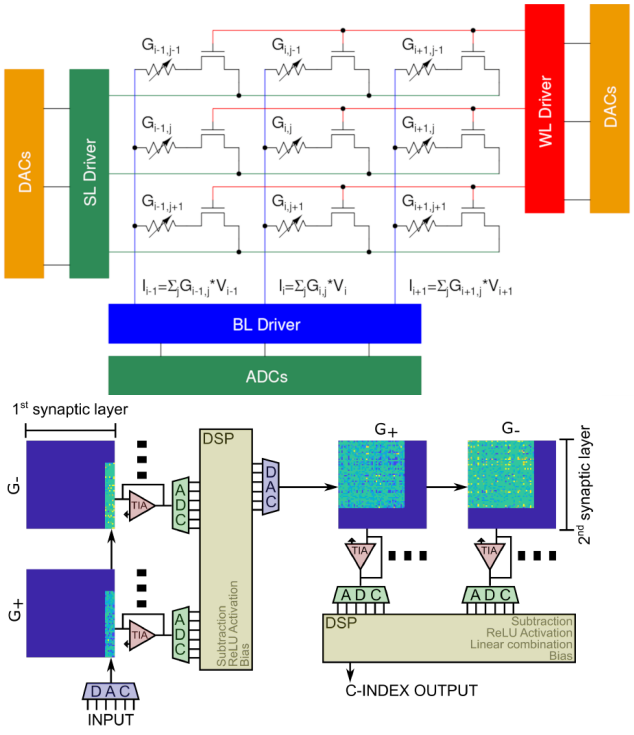


Fig. 5. RRAM 1T-1R crossbar array architecture for MVM operations with additional peripheral circuitry (top). RRAM-based implementation of a DNN (DeepSurv) using tiled crossbars. The additional circuitry such as ADCs, DACs, and DSPs required for the operations outside the MVMs are indicated as well (bottom). Adapted with permission from [46] under Creative Commons License (CC BY 4.0).

A. Accurate conductance control for weights mapping

RRAM devices are used to map the synaptic weights of a DNN architecture. MLC program and verify techniques have been demonstrated as the best device-level solution to precisely tune the conductance of RRAM cells, with the main benefit of overcoming the C2C and D2D variability effects [26]. In our work [47], we assessed the precision of two MLC algorithms in RRAM-based CIM architectures by comparing a program and verifying algorithm based on the modulation

of top electrode voltage (Incremental Step Pulse with Verify Algorithm – ISPVA) and a refined version based on the gate voltage modulation of the select transistor (Incremental Gate Voltage Verify Algorithm – IGVVA). The second algorithm demonstrated better conductance control due to the higher accuracy in the device current control arising from the tight relation between the gate voltage applied to the transistor and the compliance current used in RRAM operations [48]. The IGVVA was tested using two voltage steps in the program and verify approach, namely 100 mV (IGVVA-100) and 10 mV (IGVVA-10). To compare the programming accuracy of these MLC algorithms, we programmed 5 levels into a 4-kbit RRAM array and measured the conductance distributions owing to the results of Fig. 6a. The accuracy of the different MLC algorithms has then been studied for encoding synaptic weights in a neural network by extending the programming to 9 different conductance levels from 8 μS to 225 μS and devising them to implement the 4-kbit synaptic weights of a 2-layer fully connected neural network (FC-NN) investigated in [26]. The network consists of an input layer including 197 neurons, a hidden layer with 20 neurons, and an output layer with 10 neurons. To maximize the inference accuracy of the neural network, we implemented the FC-NN synaptic weights using the 9 conductance levels obtained with the MLC algorithms combined with the differential scheme illustrated in [47], namely by encoding the weight as the difference of two conductance G^+ and G^- [49]. Fig. 6b shows that the DNN accuracy is heavily impacted by the conductance precision in terms of the number of levels of the synaptic weight.

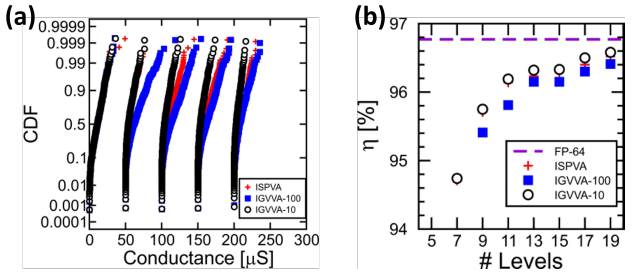


Fig. 6. (a) Conductance CDFs of HRS and 4 LRS levels measured after the application of the ISPVA, IGVVA-100, and IGVVA-10 MLC algorithms. IGVVA-10 CDFs display the lowest D2D variability. (b) Calculated inference accuracy (η) of the 2-layer FC-NN as a function of the number of synaptic weight levels programmed by IGVVA-100, ISPVA, and IGVVA-10 CDFs. The higher number of levels combined with IGVVA-10 programming leads η close to FP-64 accuracy. Reprinted with permission from [47] ©2021 IEEE.

B. Incremental Network Quantization

Despite reaching a considerable number of levels to represent synaptic weights with RRAM, we are still far from the radix used by CPUs or GPUs (i.e., 32/64 bits). The numerical precision of the synaptic weights can be reduced without compromising on the network accuracy through a quantization algorithm. We considered the use scenario of a DeepSurv network used for medical applications [46], but can be generalized to any DNN. As a preliminary step, we trained the DeepSurv network on a GPU with 32-bit floating-point precision using Tensorflow. Then, we implemented an iterative training algorithm described in [50] as Incremental Network Quantization (INQ). The key is to build an RRAM-aware training operation through the decision of the quantization steps number that are to be followed. In our study case, we found an acceptable trade-off in using 4 incremental quantization steps: 50%, 75%, 87.5%, and 100%. These percentage values allow for deriving the number of weights in each DeepSurv layer that will be rounded to the nearest quantization level at the end of each training epoch of the network while leaving the remaining weights free to continue with the training non-quantized. The advantage of this strategy lies in the compensation of the quantization-induced non-idealities [50]. Fig. 7(a) depicts an example of the INQ procedure application. Different INQ priority patterns can be exercised: *i)* weights with the greatest absolute value; *ii)* weights with the lowest absolute value; *iii)* weights featuring the lowest quantization error. The quantization error refers to the value calculated as the absolute difference between the value of the weight at the end of the training and the value of the closest quantization level. Fig. 7b shows an example of the efficiency of the three different policies reached within 100 experiments considering the C-index [51] DeepSurv accuracy metric.

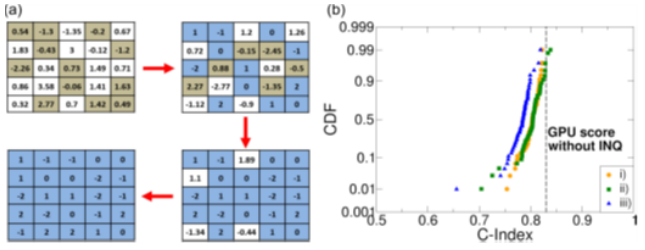


Fig. 7. (a) Different application steps of the INQ algorithm. (b) CDF of the DeepSurv C-index retrieved in Monte Carlo simulations with different weight-picking strategies compared with the C-index without quantization in training and working with full floating-point precision. Reprinted with permission from [46] under Creative Commons License (CC BY 4.0).

C. Variation-aware training

The training of a DNN is a process that is extremely sensitive to the fluctuations of the synaptic weights [41]. Even slight variations can move the optimal work of the network leading to severe accuracy loss during inference. A solution to improve network resilience against this issue is to deliberately inject some noise into the synaptic weights during the training, exploiting a technique called variability-aware training (VAT) [52]. In RRAM-based DNN design, such noise is linked to the D2D and C2C variability as both represent a major reliability threat. Fig. 8a demonstrates the inference accuracy of a Convolutional Neural Network (CNN) mapped on RRAM crossbars without noise injection (VAT0, in red), and with the injection of Gaussian noise having a standard deviation of 8% over the maximum (VAT8, in violet). Under ideal circumstances (i.e., no inference variability), the network can reach the ideal software accuracy of 98% in both cases, while, as the variability of weights during inference increases, VAT8 performs significantly better, gaining roughly 10% in accuracy. Fig. 8b shows that for variability spanning from 2% to 16% during the inference, VAT gradually increases the

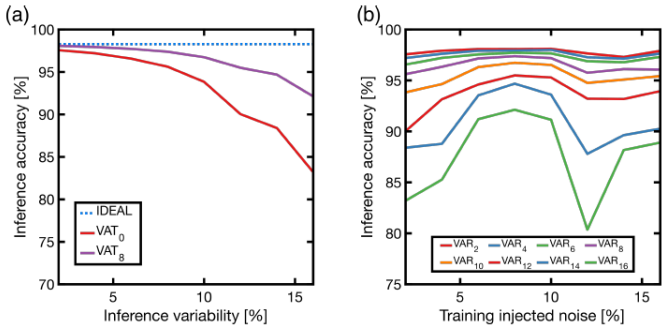


Fig. 8. The impact of the variability of the synaptic weights during the inference on the model trained without VAT (in red), and with 8% VAT (in violet) (a). Inference accuracy of DNN with different amounts of synaptic weights variability. The most accurate results are obtained with an 8% VAT model (b). Reprinted with permission from [53] ©2022 IEEE.

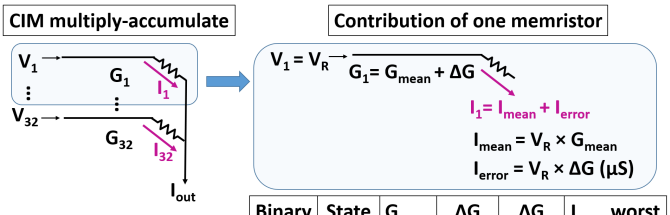
overall accuracy until a maximum effect around 8% of injected noise. Another trade-off is exposed with VAT: injecting higher noise leads to an accuracy decrease of the CNN, but a minimum amount is required to improve the overall network performance.

V. Memristor Reliability-aware Biased Training

As already mentioned, conductance variation leads to an undesired change in the neural network weights stored in a memristor, resulting in low computing accuracy. Another alternative way to address this issue is the *mapping-aware* biased training methodology; we will discuss the approach based on the work in [54]. We first identify memristor conductance states with low variation impact (favorable states) to derive a favorability constraint that only allows weight values that map to these favorable states. Then, a mapping-aware biased training is adopted to determine the weights that are important for CIM hardware accuracy and impose the favorability constraint on them. Finally, the post-training important weights are mapped to the favorable states, leading to high inference accuracy on CIM hardware.

A. Favorable Conductance States Analysis

Fig. 9 shows a CIM-based multiply-accumulate operation, where I_{error} is the error current in a single memristor device due to conductance variation. As small I_{error} is desirable, the



Binary value	State name	G_{mean} (μS)	ΔG (%)	ΔG (μS)	I_{error} worst case (μA)
00	G_{00}	2	38.5	0.8	1.6
01	G_{01}	67	24.8	16.6	6.6
10	G_{10}	134	17.2	23.1	9.2
11	G_{11}	200	9.3	18.6	7.4

V_R : Input voltage (0.4 V)

G_{mean} : Mean conductance

ΔG : Worst case variation

Fig. 9. Favorable conductance states analysis for a 2-bit memristor (four conductance states). Reprinted with permission from [54] ©2023 IEEE. The used conductance variation data is obtained from [55].

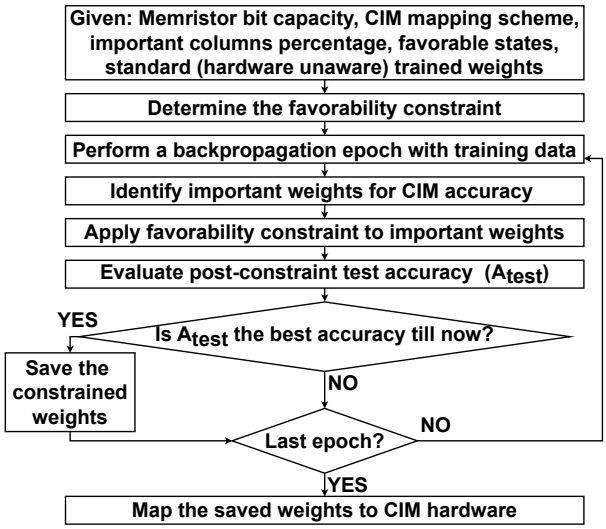


Fig. 10. Flowchart of the proposed mapping-aware biased training. Reprinted with permission from [54] ©2023 IEEE.

preference order of states in Fig. 9 is: G_{00} (best), G_{01}, G_{11}, G_{10} (worst). Despite having a higher variation percentage, G_{00} and G_{01} are preferred over G_{11} and G_{10} as their small mean values result in small I_{error}. Hence, the preference order of conductance states must be based on I_{error} contribution instead of the variation percentage. The ordered conductance states are then grouped into i) unfavorable states (U) to avoid mapping, and ii) favorable states (F) to prefer mapping. Based on Fig. 9, the possible grouping configurations are:

- Config-1: F={G_{00}}, U={G_{01},G_{11},G_{10}}
- Config-2: F={G_{00}, G_{01}}, U={G_{11},G_{10}}
- Config-3: F={G_{00}, G_{01}, G_{11}}, U={G_{10}}

Config-1 sets weights only to zero while config-2 forces them to the same sign. This is undesirable as the neural network requires both positive and negative non-zero weights. As config-3 can represent non-zero weights with different signs, it is used in our mapping-aware biased training methodology.

B. Mapping-aware Biased Training Methodology

The flowchart of the mapping-aware biased training is shown in Fig. 10. Firstly, the network is trained in a conventional manner to extract the initial weights for mapping-aware training. Then, a favorability constraint is determined to ensure the mapping of desired weights to the favorable conductance states. For example, consider 2-bit memristors (slices), 8-bit fixed-point weights (6-bit fraction), and CIM architecture in [56]. Fig. 11 shows the favorability constraint to map the most significant 2-bit slice to favorable states (G_{00}, G_{01}, and G_{10}) as described in Section V-A. The favorability constraint is then used during mapping-aware biased training to determine the weights that are important for high hardware accuracy and map them to the favorable states.

C. Results and discussion

1) Simulation Setup: A Python-based framework is developed for the behavioral simulation of neural network inference on CIM hardware. It is based on in-situ multiply-accumulate

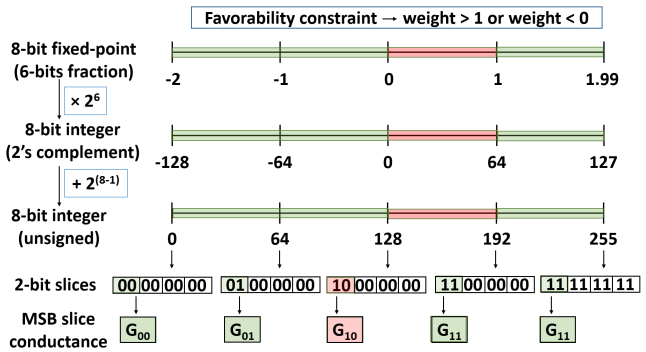


Fig. 11. Illustration of favorability constraint derivation for mapping MSB slice of 8-bit weight to favorable conductance states in a 2-bit memristor. Reprinted with permission from [54] ©2023 IEEE.

(IMA) units in state-of-the-art CIM architectures [56], [57]. We consider 8-bit weights split across four memristors of 2-bit capacity. Experimental memristor device parameters and conductance variation data are obtained from [55]. The evaluation is performed using MNIST [58], Fashion MNIST (FMNIST) [59], and EMNIST (EMNIST-L) [60] datasets.

2) Network accuracy evaluation: Fig. 12 shows the accuracy of the proposed technique in comparison to the conventional training (backpropagation). The proposed mapping-aware biased training has a slightly lower software accuracy compared to conventional training. This is because cost function minimization during training becomes difficult due to the favorability constraint on important weights. However, the proposed biased training provides up to 2.4× hardware accuracy compared to conventional training. This can be attributed to the mapping of important weights to conductance states having a low variation impact.

The comparison of hardware metrics between the proposed mapping-aware biased training and the conventional training (backpropagation) is shown in Table I. They both need identical hardware components and hence consume the same energy and area. We define a new metric "correct operations per unit energy" as the ratio of the number of correct operations to energy consumption (unit: Giga-operations per joule (GOP/J)). Here, the number of correct operations is the product of accuracy (as a fraction) and the total number of operations. Table I shows that the proposed mapping-aware biased training achieves up to 2.4× correct operations per unit energy than conventional training without any hardware overhead.

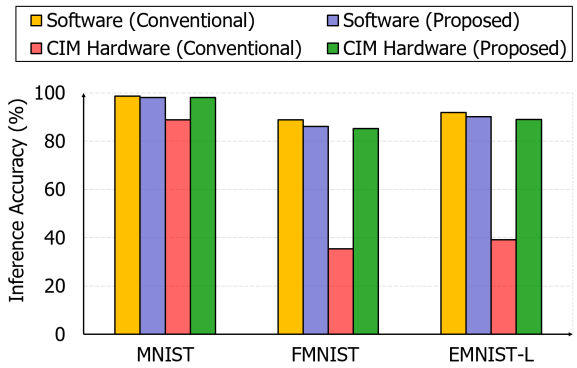


Fig. 12. Inference accuracy comparison across various datasets. Reprinted with permission from [54] ©2023 IEEE.

TABLE I
HARDWARE METRICS PER IN-SITU MULTIPLY-ACCUMULATE UNIT. REPRINTED WITH PERMISSION FROM [54] ©2023 IEEE.

Metric	Conventional Training	Proposed Mapping-aware Biased Training
FMNIST accuracy (%)	35.4	85.2
Energy consumption (pJ)	3738	3738
Area (μm^2)	21765	21765
Correct operations per unit energy for FMNIST (GOP/J)	96.9	233.4

VI. CONCLUSION

RRAM is an emerging non-volatile memory technology that has generated significant interest in the field of computing due to its potential advantages, such as high speed, low power consumption, and scalability. However, its reliability faces serious challenges in two aspects: (1) time-zero reliability and (2) time-dependent reliability. In particular, RRAM-based computing accuracy is impacted by variability as well as intermittent faults (i.e., sudden conductance drifts). Several research directions intended to mitigate these drawbacks of the technology have attracted much attention. Among these, variation-aware training and reliability-aware biased training are developed in this review. The outcomes of this review show that designing reliable RRAM-based computing systems involves hardware combined with software strategies to address non-idealities and ensure reliable operation over time.

REFERENCES

[1] D. Kimovski *et al.*, "Beyond von neumann in the computing continuum: Architectures, applications, and future directions," *Internet Computing*, 2023.

[2] H. A. Du Nguyen *et al.*, "Memristive devices for computing: Beyond cmos and beyond von neumann," in *VLSI-SoC*, 2017.

[3] A. Gebregiorgis *et al.*, "Tutorial on memristor-based computing for smart edge applications," *MMDCS*, 2023.

[4] J. Wang *et al.*, "Emerging memristive devices for brain-inspired computing and artificial perception," *Frontiers in Nanotechnology*, 2022.

[5] S. H. Jo *et al.*, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, 2010.

[6] P. Yao *et al.*, "Face classification using electronic synapses," *Nature communications*, 2017.

[7] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *IEEE Proceedings*, 2014.

[8] F. Cüppers *et al.*, "Exploiting the switching dynamics of hfo2-based reram devices for reliable analog memristive behavior," *materials*, 2019.

[9] Y. Jeong *et al.*, "Utilizing multiple state variables to improve the dynamic range of analog switching in a memristor," *Applied Physics*, 2015.

[10] F. Zahoor *et al.*, "Resistive random access memory (rram): an overview of materials, switching mechanism, performance, multilevel cell (mlc) storage, modeling, and applications," *Nanoscale research letters*, 2020.

[11] J. Woo *et al.*, "Resistive memory-based analog synapse: The pursuit for linear and symmetric weight update," *Nanotechnology magazine*, 2018.

[12] H. Aziza *et al.*, "State: A test structure for rapid and reliable prediction of resistive ram endurance," *Device & Materials Reliability*, 2022.

[13] M. Fieback *et al.*, "Intermittent undefined state fault in rrams," in *ETS*, 2021.

[14] D. Veksler and G. Bersuker, "Advances in rram technology: Identifying and mitigating roadblocks," *High Speed Electronics and Systems*, 2016.

[15] S.-K. Lu *et al.*, "Fault securing techniques for yield and reliability enhancement of rram," in *ATS*, 2022.

[16] S. Pal *et al.*, "Design of power-and variability-aware nonvolatile rram cell using memristor as a memory element," *EDS*, 2019.

[17] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *DATE*, 2017.

[18] C. Liu *et al.*, "Rescuing memristor-based neuromorphic design with high defects," in *DAC*, 2017.
[19] W.-Q. Pan *et al.*, "Strategies to improve the accuracy of memristor-based convolutional neural networks," *TED*, 2020.
[20] D. Joksas *et al.*, "Committee machines—a universal method to deal with non-idealities in memristor-based neural networks," *Nature communications*, 2020.
[21] F. M. Bayat *et al.*, "Memristor-based perceptron classifier: Increasing complexity and coping with imperfect hardware," in *ICCAD*, 2017.
[22] D. Joksas *et al.*, "Nonideality-aware training for accurate and robust low-power memristive neural networks," *Advanced Science*, 2022.
[23] A. Mehonic *et al.*, "Mitigating non-idealities of memristive-based artificial neural networks-an algorithmic approach," in *EDTM*, 2022.
[24] D. Gaol *et al.*, "Reliable memristor-based neuromorphic design using variation-and defect-aware training," in *ICCAD*, 2021.
[25] S. Ambrogio *et al.*, "Statistical Fluctuations in HfOx Resistive-Switching Memory: Part I - Set/Reset Variability," *TED*, 2014.
[26] V. Milo *et al.*, "Optimized programming algorithms for multilevel RRAM in hardware neural networks," in *IRPS*, 2021.
[27] A. Baroni *et al.*, "Low Conductance State Drift Characterization and Mitigation in Resistive Switching Memories (RRAM) for Artificial Neural Networks," *dmr*, vol. 22, 2022.
[28] A. Gebregiorgis *et al.*, "Dealing with non-idealities in memristor based computation-in-memory designs," in *VLSI-SoC*, 2022.
[29] Y. Jeong *et al.*, "Parasitic effect analysis in memristor-array-based neuromorphic systems," *Nanotech*, 2018.
[30] M. Lanza *et al.*, "Standards for the characterization of endurance in resistive switching devices," *ACS nano*, 2021.
[31] W. Shim *et al.*, "Impact of read disturb on multilevel rram based inference engine: Experiments and model prediction," in *IRPS*, 2020.
[32] D. Lee *et al.*, "Oxide based nanoscale analog synapse device for neural signal recognition system," in *IEDM*, 2015.
[33] C. Mohan *et al.*, "Neuromorphic low-power inference on memristive crossbars with on-chip offset calibration," *IEEE Access*, 2021.
[34] E. Pérez *et al.*, "Reduction of the cell-to-cell variability in hf 1-x al x o y based rram arrays by using program algorithms," *TED*, 2016.
[35] H. Aziza *et al.*, "True random number generator integration in a resistive ram memory array using input current limitation," *Nanotech*, 2020.
[36] J. Postel-Pellerin *et al.*, "True random number generation exploiting set voltage variability in resistive ram memory arrays," in *NVMTS*, 2019.
[37] H. Aziza *et al.*, "Experimental analysis of oxide-based ram analog synaptic behavior," *Electronics*, 2022.
[38] H. a. o. Aziza, "Multi-level control of resistive ram (rram) using a write termination to achieve 4 bits/cell in high resistance state," *Electronics*, 2021.
[39] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, 2011.
[40] H. Aziza *et al.*, "Design considerations towards zero-variability resistive rams in hrs state," in *LATS*, 2021.
[41] Y. LeCun *et al.*, "Deep learning," *Nature*, 2015.
[42] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *ISSCC*, 2014.
[43] D. Ielmini and G. Pedretti, "Device and Circuit Architectures for In-Memory Computing," *AIS*, 2020.
[44] S. N. Truong *et al.*, "New Memristor-Based Crossbar Array Architecture with 50-% Area Reduction and 48-% Power Saving for Matrix-Vector Multiplication of Analog Neuromorphic Computing," *SEMICONDUCTOR TECHNOLOGY AND SCIENCE*, 2014.
[45] W. Wang *et al.*, "Integration and Co-design of Memristive Devices and Algorithms for Artificial Intelligence," *iScience*, 2020.
[46] A. Baroni *et al.*, "An energy-efficient in-memory computing architecture for survival data analysis based on resistive switching memories," *Frontiers in Neuroscience*, 2022.
[47] V. Milo *et al.*, "Accurate Program/Verify Schemes of Resistive Switching Memory (RRAM) for In-Memory Neural Network Circuits," *TED*, 2021.
[48] D. Ielmini, "Modeling the Universal Set/Reset Characteristics of Bipolar RRAM by Field- and Temperature-Driven Filament Growth," *TED*, 2011.
[49] G. W. Burr *et al.*, "Experimental Demonstration and Tolerancing of a Large-Scale Neural Network (165 000 Synapses) Using Phase-Change Memory as the Synaptic Weight Element," *TED*, 2015.
[50] A. Zhou *et al.*, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," 2017.
[51] F. E. Harrell, Jr *et al.*, "Regression modelling strategies for improved prognostic prediction," *Stat. Med.*, 1984.
[52] B. Liu *et al.*, "Vortex: Variation-aware training for memristor x-bar," in *DAC*, 2015.
[53] A. Glukhov *et al.*, "End-to-end modeling of variability-aware neural networks based on resistive-switching memory arrays," in *VLSI-SoC*, 2022.
[54] S. Diware *et al.*, "Mapping-aware biased training for accurate memristor-based neural networks," in *AICAS*, 2023.
[55] A. Prakash and H. Hwang, "Multilevel Cell Storage and Resistance Variability in Resistive Random Access Memory," *PSR*, 2016.
[56] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA*, 2016.
[57] A. Ankit *et al.*, "PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference," in *ASPLOS*, 2019.
[58] Y. Lecun *et al.*, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, 1998.
[59] H. Xiao *et al.*, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv*, 2017.
[60] G. Cohen *et al.*, "EMNIST: Extending MNIST to handwritten letters," in *IJCNN*, 2017.