TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Aziza, H., Xun, H., Fieback, M., Taouil, M., & Hamdioui, S. (2026). Optimization Trade-Offs in Memristor-Based Crossbar Arrays for MAC Acceleration. *Electronics (Switzerland)*, *15*(8), Article 1710. https://doi.org/10.3390/electronics15081710

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states "Dutch Copyright Act (Article 25fa)", this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

electronics

MDPI

Article

Optimization Trade-Offs in Memristor-Based Crossbar Arrays for MAC Acceleration

Hassen Aziza [1,*], Hanzhi Xun [2], Moritz Fieback [2], Mottaqiallah Taouil [2] and Said Hamdioui [2]

[1] CNRS, IM2NP, Aix-Marseille University, 13451 Marseille, France
[2] Computer Engineering Laboratory, Delft University of Technology, 2628CD Delft, The Netherlands; h.z.xun@outlook.com (H.X.); m.c.r.fieback@tudelft.nl (M.F.); m.taouil@tudelft.nl (M.T.); s.hamdioui@tudelft.nl (S.H.)
* Correspondence: hassen.aziza@univ-amu.fr

Abstract

Vector–matrix multiplication (VMM), implemented through multiply–accumulate (MAC) operations, represents the dominant computational primitive in many artificial intelligence (AI) workloads. When executed on conventional von Neumann architectures, VMM operations suffer from important energy consumption and latency due to the separation between memory and processing units. To overcome these limitations, crossbar arrays built from Resistive Random Access Memory (RRAM) cells have been proposed for accelerating VMM computations. In this work, we investigate the key optimization trade-offs associated with implementing RRAM-based neural networks for classification applications. A simple two-layer neural network is first defined and trained in software to generate the weight matrices and bias parameters. Next, three hardware implementation scenarios are evaluated depending on whether negative floating-point numbers are used: Positive Weights Only (PWO), Positive and Negative Weights Only (PNWO), and Positive and Negative Weights with Biases (PNWB). The different implementations are analyzed at the hardware level by examining classification accuracy, energy efficiency, latency, and area overhead. The study further incorporates important RRAM limitations, including restricted conductance range and device variability. Hardware results show that the PWO scenario offers the lowest energy consumption (189 fJ/MAC) and area overhead but results in the lowest accuracy. PNWO and PNWB significantly improve accuracy (+177% and +180%) but increase energy consumption (+63% and +87%) and area (×2 and ×2.1). Under variability effects, PWO achieves better accuracy (94.65%), followed by PNWO (93.11%) and PNWB (92.11%).

Keywords: memristors; crossbar arrays; RRAM; hardware acceleration; vector–matrix multiplication; neural networks; image classification

Academic Editor: Mahsa Mehrad

Received: 13 March 2026
Revised: 6 April 2026
Accepted: 12 April 2026
Published: 17 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Modern AI hardware like GPUs and TPUs is engineered for parallel execution and efficient large-scale data processing [1]. In contrast, memory systems often struggle to deliver data at the rate required by these processors, creating a significant performance gap [2]. This bottleneck becomes particularly critical in tasks like deep learning that rely on artificial neural network (ANN) models used to solve complex problems such as pattern recognition [3]. From a computational standpoint, vector–matrix multiplications (VMMs) dominate the workload of ANNs with 99% of the overall operations, particularly in convolutional neural networks (CNNs) [4]. During VMM operations in artificial neural

networks, the processing workload may surpass the speed at which data can be retrieved from memory. This imbalance is often described as the memory wall, which reflects the increasing disparity between processor performance and memory access bandwidth [5]. While this disparity grows, there is a pressing need for alternative approaches, and in-memory computing emerges as a promising paradigm, enabling data processing to occur directly within memory units [6].

One potential candidate for in-memory computing is Resistive Random Access Memory (RRAM). RRAM has gained particular attention in the context of ANNs due to its ability to emulate synaptic behavior, while being easily integrated into the back-end-of-line (BEOL) process of transistors [7]. ANNs are composed of numerous interconnected processing elements, commonly called neurons or nodes, arranged in a particular architecture to enable communication between them. Neurons interact through connections known as synapses, each of which is assigned a weight that affects signal transmission over the network. These weights can either strengthen or weaken the signals passing through the network [8,9]. In this context, RRAM crossbar arrays have emerged as a powerful solution, capable of emulating the structure of an ANN by directly storing and manipulating synaptic weights in memory [10]. In RRAM crossbar architectures, VMM is achieved by exploiting the programmable conductance of individual memory cells. Input vectors are encoded as voltages applied along the rows, while synaptic weights are mapped to the conductance states of the cells. The resulting currents, proportional to the product of voltage and conductance, naturally perform parallel weighted summations across columns in a single operation [11,12]. However, recent research has highlighted several limitations of performing VMM using crossbar arrays of RRAM devices, including device variability [13], a limited conductance range [14], nonlinear I-V characteristics [15], interconnect resistance [16], and sneak path currents [17]. Overcoming these challenges is essential for the efficient deployment of RRAM crossbar arrays in machine learning applications.

The primary contributions of this study are outlined below:

- The modulation capability of manufactured RRAM devices is assessed through electrical characterization.
- An ANN model is trained for image classification, generating three distinct sets of parameters based on constraints imposed on weights and biases.
- Post-trained parameters are mapped onto an RRAM crossbar array, enabling a hardware implementation.
- Electrical simulations are conducted to assess the crossbar array's ability to perform VMM while accounting for critical hardware limitations, including device variability and a restricted conductance range.

The rest of this paper is organized as follows. Section 2 presents the specifications of the fabricated RRAM cells. In Section 3, the ANN hardware weight mapping methodology is presented considering a 2-layer neural network designed for image classification. In Section 4, electrical simulations of the hardware implementation, including VMM multiplication, are conducted and compared with software-level simulations. Section 5 examines in depth the influence of RRAM variability on the ANN performance. Section 6 discusses the key findings and their implications. Section 7 concludes the paper.

2. Specifications of Manufactured RRAMs

2.1. RRAM Memory Cell Characterization

Figure 1a presents a 1T-1R RRAM cell. In the 1T-1R structure a single transistor (W = 0.8 µm and L = 0.5 µm) is serially attached to one resistive element (RRAM) integrated in the BEOL between metal lines of 130 nm technology [7]. The resistive stack is added using Physical Vapor Deposition (PVD). A 10 nm layer of hafnium dioxide (HfO_2) is first

deposited on a TiN bottom electrode (BE). After that, A Ti/TiN bilayer stack is placed to form the top electrode (TE). Figure 1b shows a classical 1T-1R I-V characteristic in the form of an electrical hysteresis. Based on this hysteresis, the device operation can be seen as follows: after an initial electro-Forming (FMG) step, not described here [18], the RRAM cell can be switched between a high-resistance state (HRS) and a low-resistance state (LRS) by applying appropriate voltages to the BL (Bit Line), WL (Word Line), and SL (Source Line) nodes. RRAM switching corresponds to a sudden change between the HRS and the LRS. The resistance change is triggered by applying a specific voltage across the 1T-1R cell. V_{SET} switches the cell to LRS after a SET operation and V_{RST} switches the cell to HRS after a RESET (RST) operation. To highlight the variability of the technology, Figure 1c presents in gray different SET/RST cycles superimposed on the initial I-V characteristic. It shows that the RRAM electrical parameters, including R_{HRS}, R_{LRS}, V_{SET} and V_{RST} are clearly affected by variability. In the 1T–1R structure, the transistor modulates the cell current according to its gate voltage, constraining it to a maximum level I_{cc}, referred to as the compliance current. Regarding device reliability, the technology achieves endurance of >10^6 cycles.

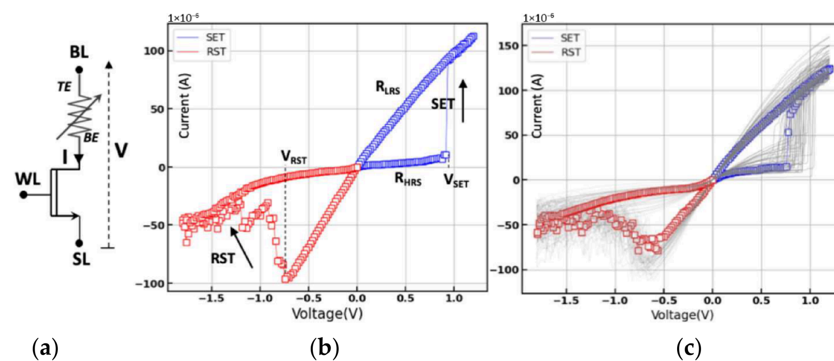

Figure 1. (**a**) Symbol view of a 1T-1R cell. (**b**) RRAM I-V characteristic. (**c**) RRAM I-V characteristics superimposed to show the variability of the technology.

Table 1 summarizes the bias conditions together with the associated nominal resistance and conductance values. A nominal conductance ratio of approximately 16 (66.6 µS/4 µS) can be achieved. During the read operation, a low voltage (around 0.1 V) is used to prevent any alteration of the RRAM cell's current state.

Table 1. RRAM cell operating voltages.

	FMG	RST	SET	READ
WL	2 V	2.5 V	2 V	2.5 V
BL	3.3 V	0 V	1.2 V	0.1 V
SL	0 V	1.2 V	0 V	0 V
Resistance	10 kΩ	240 kΩ	15 kΩ	-
Conductance	100 µS	4 µS	66.6 µS	-
RRAM state	LRS	HRS	LRS	unchanged

The switching mechanism in RRAM devices can be understood at the physical level as the formation or dissolution of conductive filaments (CFs). Under an externally applied voltage across the device, filaments consisting of oxygen vacancies are generated or ruptured according to the voltage polarity. During the SET process, these filaments develop within the metal oxide layer and connect the top and bottom electrodes, thereby creating a conductive path that allows current to pass through the cell. A notable feature of the considered RRAM technology is its ability to provide a wide spectrum of LRS states depending on the SET compliance current level. The higher the gate voltage, the greater

the SET compliance current, and the thicker the CFs, resulting in higher cell conductance. This feature of the technology can be understood as the progressive formation of CFs [18], as shown in Figure 2, leading to multiple LRS levels, ranging from LRS1 to LRS3. This ability of a single memory cell to store multiple bits of data by utilizing different resistance states is also referred to as Multi-Level Cell (MLC).

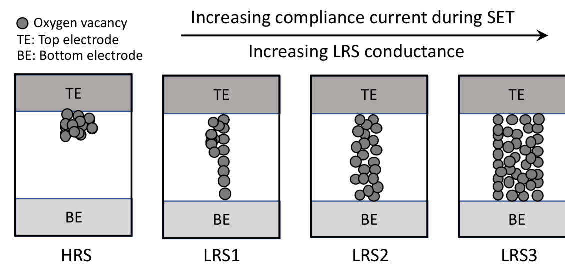

Figure 2. Illustration of the MLC capability of RRAM technology. MLC can be achieved by controlling the SET compliance current through the gate of the select transistor in a 1T-1R cell during SET. As the gate voltage increases, the CF thickness increases between the TE and BE electrodes, resulting in multiple resistance levels (LRS1 to LRS3).

2.2. RRAM Memory Array Characterization

Measurements were performed on a dedicated test chip integrating an RRAM memory array, described in detail in [13]. The fabricated elementary 7×7 cell array is available both at the wafer level and in packaged form for electrical characterization. The measurement protocol is presented in Figure 3a. After an initial FMG stage, all the CFs of the 49 RRAM cells are created. Then, the memory array is programmed 200 times (i.e., 200 successive RST/SET cycles are applied to each cell of the 7×7 memory array). To characterize the distribution of HRS resistances (following individual RST operations) and LRS resistances (following individual SET operations), a READ operation is conducted after each RST and SET operation. Note that this protocol allows capturing device-to-device (D2D) as well as cycle-to-cycle (C2C) variability. D2D variability is captured by considering multiple cells across the array. C2C variability is captured by repeating SET/RST operations on each cell. Figure 3b presents the HRS and LRS resistance cumulative probabilities. Each distribution includes 9800 measurements (49×200). Both HRS and LRS suffer from variability, with a more pronounced HRS spread, which is a classical characteristic of the addressed RRAM technology [19].

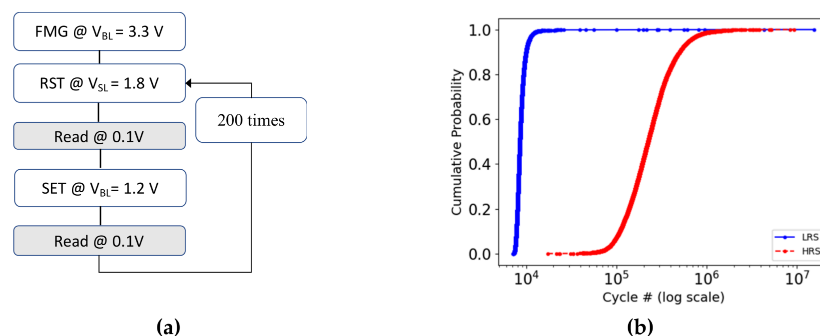

Figure 3. (**a**) Measurement protocol: after FMG, each addressed cell undergoes 200 repeated RST/SET cycles. Each RST and SET operation is followed by a readout to determine the cell resistance. (**b**) Cumulative probability of HRS and LRS obtained from 9800 SET and RST cycles.

2.3. RRAM Conductance Modulation for Synapse Emulation

The synaptic behavior of a RRAM cell relies on its ability to electrically and incrementally increase or decrease the conductance of the cell. Different approaches can be

used to modulate the RRAM cell conductance, including controlling the maximum voltage during RST [20], adjusting the SET compliance current [21], or varying the pulse width and amplitude of SET/RST pulses [22]. In this work, the compliance current modulation approach is employed by adjusting the compliance current I_{cc} through the RRAM cell's gate voltage V_{WL}.

During a conventional SET operation, the final cell resistance depends on the maximum current permitted through the cell. This current, flowing through the bit line, is controlled by adjusting the word line voltage V_{WL}. Figure 4a shows the evolution of the different I-V characteristics in the SET direction for different gate voltage values V_{WL}. 11 different V_{WL} voltage levels are considered, ranging from 1 V to 2 V with a 0.1 V voltage step. When V_{WL} increases, the maximum SET current I_{cc} increases, resulting in 11 distinctive LRS levels. Figure 4b presents the impact of V_{WL} on R_{LRS} during a SET operation at the memory array level. Each WL gate voltage is associated with 490 resistance values related to the cells of the 7 × 7 memory array. When V_{WL} is low, resistance dispersion remains minimal due to the strong control exerted by the gate voltage over the cell current. As V_{WL} increases, resistance dispersion expands [23]. The median values of each distribution, connected by a bold black line, illustrate a consistent decrease in RRAM cell resistance with increasing gate voltage. Table 2 summarizes the key parameters of each box plot distribution shown in Figure 4b. For each gate voltage level, the mean and standard deviation of the resistance distribution have been computed. Two key observations can be made: (i) the median resistance and conductance values range from 11.01 kΩ to 38.56 kΩ and 83.96 μS to 25.92 μS, respectively; (ii) as the gate voltage decreases, the standard deviation increases, limiting the feasibility of implementing lower conductance values.

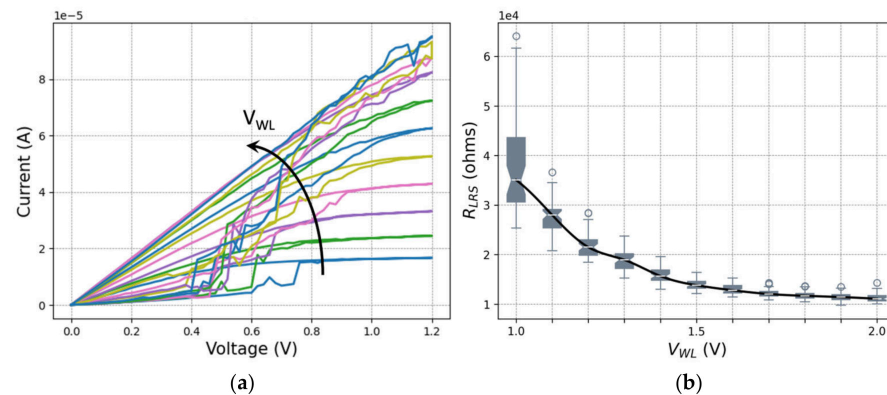

Figure 4. (**a**) RRAM I-V characteristics in the SET direction for increasing V_{WL} and (**b**) $R_{LRS}(V_{WL})$ box plots extracted at the memory array level, where each box plot includes 490 resistance values, with V_{WL} ranging from 1 V to 2 V with a 0.1 V voltage step.

Table 2. LRS(V_{WL}) BOX PLOT ANALYSIS.

V_{WL} (V)	μ (Ω)	μ (μS)	σ (Ω)
2	11.30 kΩ	87.8 μS	1101
1.9	11.56 kΩ	86.52 μS	993
1.8	11.90 kΩ	83.99 μS	934
1.7	12.21 kΩ	81.92 μS	777
1.6	12.99 kΩ	76.94 μS	859
1.5	14.08 kΩ	71.02 μS	1097
1.4	16.02 kΩ	62.43 μS	1546
1.3	18.89 kΩ	52.96 μS	1996
1.2	22.27 kΩ	44.94 μS	3079
1.1	28.83 kΩ	34.69 μS	5413
1	38.56 kΩ	25.92 μS	9851

3. Neural Network Hardware Mapping Methodology

This section outlines a detailed, step-by-step approach for implementing an RRAM-based neural network aimed at image classification. The approach consists of transferring the neural network parameters (i.e., weights and biases) from a pre-trained software model onto a physical RRAM crossbar array, following three steps: (i) defining the ANN architecture and generating the offline model using a software-based approach; (ii) mapping the learned weights and biases onto the hardware; (iii) executing vector–matrix multiplication at the RRAM crossbar array level during inference.

3.1. ANN Architecture Definition and Training

A two-layer fully connected (FC) neural network (NN) is designed, trained, and evaluated for image recognition at the software level. The dataset includes ten different classes consisting of digits (0–9). Each class contains six distinct instances. This simplified dataset is used only to validate the hardware mapping methodology. Each input image consists of a 5 × 4 pixel matrix, totaling 20 pixels. Pixel intensities are encoded as integer values ranging from 0 (black) to 255 (white) and are normalized to the [0, 1] interval after a preprocessing step. The model architecture is presented in Figure 5. It comprises an input layer (20 neurons) and an output layer (10 neurons, representing digit classes). The network has 200 weights (20 × 10), represented by synaptic connections, along with 10 biases. Training is performed over 200 epochs. We deliberately omit the *softmax* output layer to consider raw output values, which are essential for direct comparison with the hardware crossbar array's output responses (i.e., currents).

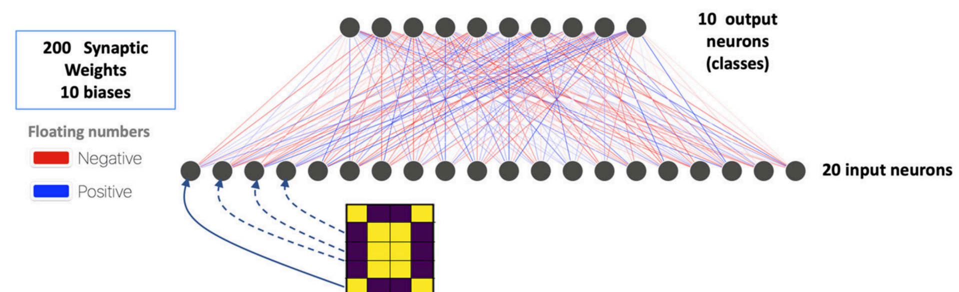

Figure 5. A two-layer FC neural network with 20 input neurons and 10 output neurons (used to encode the 5 × 4 pixel digits), defined by 210 parameters: 200 weights and 10 biases.

In *TensorFlow*, when building a NN model, various options are available for configuring the model parameters according to specific requirements, including polarity. Three scenarios are analyzed at the software level depending on whether negative floating-point numbers are used: Positive Weights only (PWO), Positive and Negative Weights Only (PNWO), and Positive and Negative Weights with positive and negative Biases (PNWB). Table 3 summarizes these three scenarios along with the corresponding software accuracy and loss function results. Software simulations confirm 100% inference accuracy after 200 epochs, validating the model's correct predictions before hardware deployment. However, the loss function results, which quantify the model error, show significant differences, with scenario #3 showing the lowest loss value.

Table 3. Software training results for the three scenarios.

Scenarios	Acronym	Accuracy	Loss
#1: Positive Weights Only	PWO	100%	0.810
#2: Positive and Negative Weights Only	PNWO	100%	0.555
#3: Positive/Negative Weights & Biases	PNWB	100%	0.162

3.2. ANN Hardware Mapping onto a Crossbar Array

Figure 6 illustrates how the 2-layer neural network model, presented in Figure 6a, can be mapped onto the crossbar array shown in Figure 6b. In Figure 6a, X_i are the input neurons and Y_k the output neurons, with i ranging from 1 to n and k ranging from 1 to m. The input and output neurons are connected through wires that act as synapses ($W_{k\text{i}}$). The software model can be converted into a hardware representation by mapping its parameters to conductance values, as illustrated in Figure 6b. This process is enabled by the select transistor, which allows individual access to each cell and precise control of the current passing through it, thereby ensuring accurate conductance mapping. In practice, the gate voltage of the selection transistor is controlled to precisely adjust the compliance current during the SET operation to obtain conductance values ranging from approximately 25 µS to 88 µS, according to Table 2. At the crossbar array level, cells in a row are arranged by connecting the transistor gates to the Source Line (*SL*) and the transistor drains to the *WL*, while the cells in a column are arranged by attaching the bottom electrodes of the 1T-1R cells to the Bit Line (*BL*). Input vectors (X_i) are associated with input voltages (V_i), while the outputs Y_k are associated with output currents I_k. Regarding the matrix of weights (W_{ki}), it is translated into a conductance matrix, where each element is represented as the inverse of the corresponding resistance R_{ki} (i.e., equivalent RRAM cell resistance). It is worth mentioning that the conductance values are extracted from the weight matrix via a linear transformation [24].

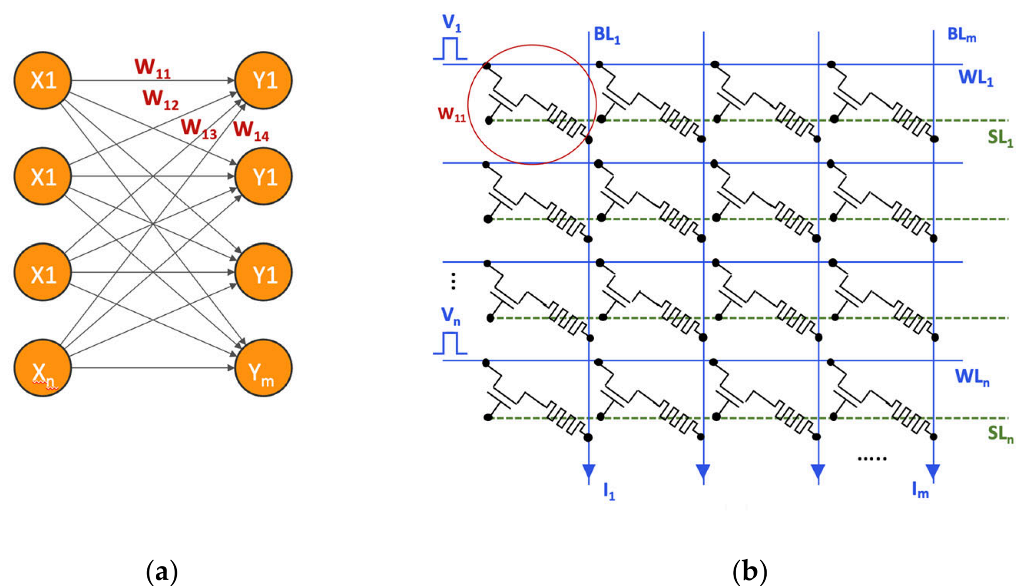

Figure 6. (**a**) A two-layer fully connected neural network with 20 inputs and 10 outputs. (**b**) A 1T-1R neural network mapped onto a crossbar array.

3.3. Vector–Matrix Multiplication in a 1T-1R Array

VMM involves multiplying a one-dimensional vector by a two-dimensional matrix to produce another vector. As previously mentioned, an array of RRAM cells can inherently perform VMM in a single step by summing the output currents of the array. At the software level, Equation (1) shows that VMM involves computing the outputs of the network Y given the input X and the weight matrix W. This operation relies on a weighted sum of inputs (i.e., each input X_i is multiplied by its corresponding weight W_{ki} and all products are summed to obtain the outputs Y_k). At the hardware level, during inference, appropriate input voltages V_i are applied to the crossbar inputs WL_i, while all selection transistors are activated via the SL_i control signals. This operation generates currents that are collected at the column lines BL_k, as expressed in Equation (2). In this equation, R denotes the programmable

resistance of the RRAM cell ($G = 1/R$ is the corresponding conductance), V_i is the applied input voltage at each row of the crossbar, and I_k is the resulting current at each column node. I_k is proportional to the product of the input voltage and the cell's conductance (which represents the weight), effectively computing weighted sums simultaneously across multiple columns. In the rest of the paper, I_k currents are considered the circuit responses.

$$Y_k = \sum_{i=1}^{i=n} W_{ki} \cdot X_i \quad 1 < k \leq m \tag{1}$$

$$I_k = \sum_{i=1}^{i=n} \frac{1}{R_{ki}} \cdot V_i \quad 1 < k \leq m \tag{2}$$

4. Neural Network Hardware Emulation

4.1. Hardware Simulation Setup

The design and simulation of a 20 × 10 1T–1R array were carried out using the *Eldo* simulator in the *Cadence Virtuoso* framework. The RRAM cells were described by a compact model calibrated on silicon measurements [25], incorporating variability tuned to match experimental data.

After conductance mapping, the inference phase converts digital pixel values into corresponding voltage levels. A white pixel (255) is mapped to 0.3 V applied to the crossbar rows, while a black pixel (0) corresponds to 0.0 V. These voltage levels are carefully chosen to avoid any unintended modification of the RRAM conductance states during inference.

Since RRAM cannot directly store negative values, differential encoding has been adopted [26]. This strategy involves using two columns for each synaptic weight, combined with bipolar (positive and negative) input signals, to differentiate between positive and negative weights. In this scheme, each synaptic weight, referred to as W, is represented by two conductances (or two columns in the crossbar array). One column, biased with a positive voltage, stores the positive component G^+, while the other, biased with a negative voltage, stores the negative component G^-. Depending on the weight sign, the corresponding column is programmed with the appropriate weight value. Similarly to weights, biases can also be encoded differentially to represent both positive and negative values using two additional rows b^+ and b^-. Biases are implemented through dedicated rows with fixed conductance values, driven by constant input voltages. This configuration allows each neuron to include a constant bias term, which is inherently summed with the weighted input signals at the crossbar outputs. The inclusion of positive and negative biases enables a shift in the neuron activation threshold, improving the representational capability of the network and facilitating the mapping of software-trained models to hardware. In particular, biases help compensate for limited conductance ranges, thereby enhancing classification accuracy without modifying the input encoding scheme. Figure 7 illustrates this approach with a 4 × 4 RRAM array. For simplicity, single-resistor (1R) RRAM cells are used to represent 1T-1R cells. In Figure 7, the green array corresponds to the positive weight's contribution, while the blue array represents the negative weight's contribution. Also, two red rows are included to account for positive and negative biases. In this approach, unused weights or biases are set to HRS values. Additionally, negative weights and biases induce negative currents I^-, meaning the current flows towards the negative terminals of the voltage sources. Also, note that transimpedance amplifiers (TIAs) can be added to the output bit lines for current-to-voltage conversion and signal amplification.

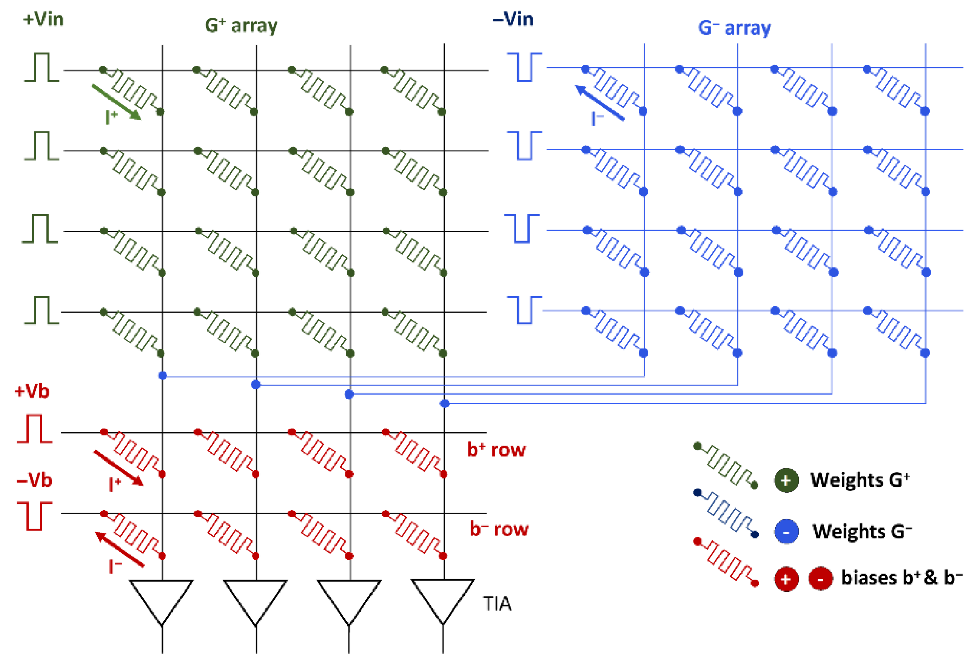

Figure 7. Illustration of a 4 × 4 elementary RRAM array, where separate 4 × 4 sets of RRAM cells are used to represent positive and negative weights. Additionally, two dedicated rows, each containing four RRAM cells, are allocated for the positive and negative biases.

4.2. Simulation Results

The accuracy of the prediction is assessed by comparing the response of the model at the software level and the currents measured for the three different scenarios. The first analysis is conducted for a NN model generated at the software level with Positive Weights Only (PWO). Figure 8 presents inference results for digits (0–9), comparing software-based (red bars) and hardware-based (blue data points) inference results. The figure is arranged into 10 subplots. The X-axis corresponds to the output classes, while the Y-axis represents both the output logits (left side) and the current values in amperes (right side). Logits represent the results of the VMM operation before applying any activation function, while currents are extracted after hardware-based simulations. For each of the ten digits, red bars and blue data points represent the amplitudes of the output logits and currents, respectively. Dark red indicates the target digit logit; light red shows logits meant to be ignored. A correct inference occurs when the amplitude of the target logit (dark red bar) exceeds all other logits in its respective subplot. The margin between the target logit and other logits indicates the inference confidence. The same principle applies to the amplitude of the data points representing currents. Based on these observations, Figure 8 demonstrates a clear correlation between current measurements and software logits, confirming the efficiency of the hardware implementation. Both software and hardware achieve a perfect inference accuracy of 100%.

Figure 9 presents inference accuracy results for the scenario where only Positive and Negative Weights (PNWO) are considered. Similarly to the previous configuration, both software and hardware achieve an overall accuracy of 100%. However, a notable difference is that negative weights lead to negative current or logit values for some digits. This characteristic enhances classification robustness, as most ignored classes receive negative values, thereby increasing the separation between the target digit and the ignored digits.

https://doi.org/10.3390/electronics15081710

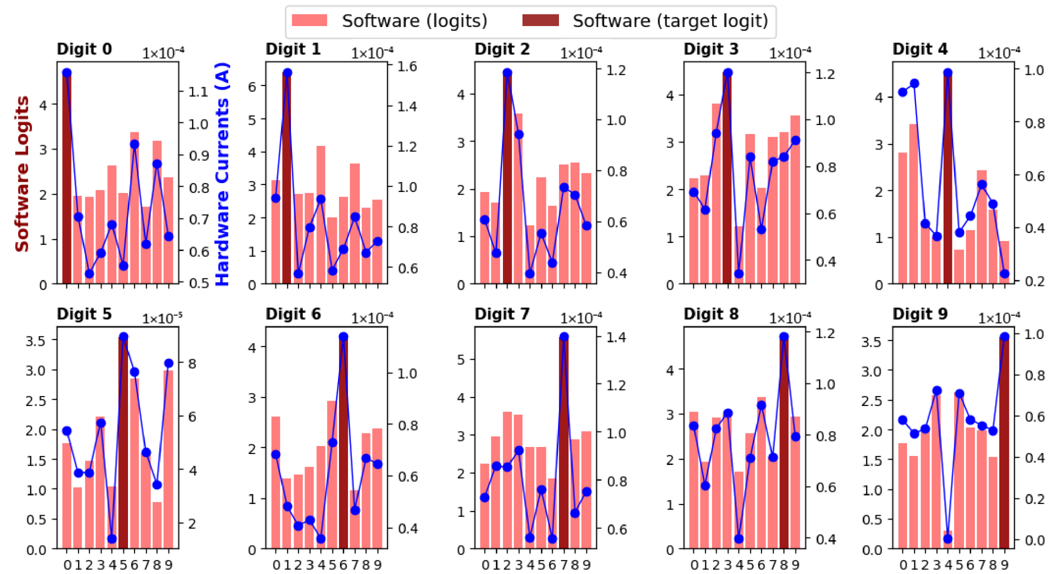

Figure 8. Inference test results achieved after NN parameters generation with Positive Weights Only (PWO) at the software (bars) and hardware (data points) level.

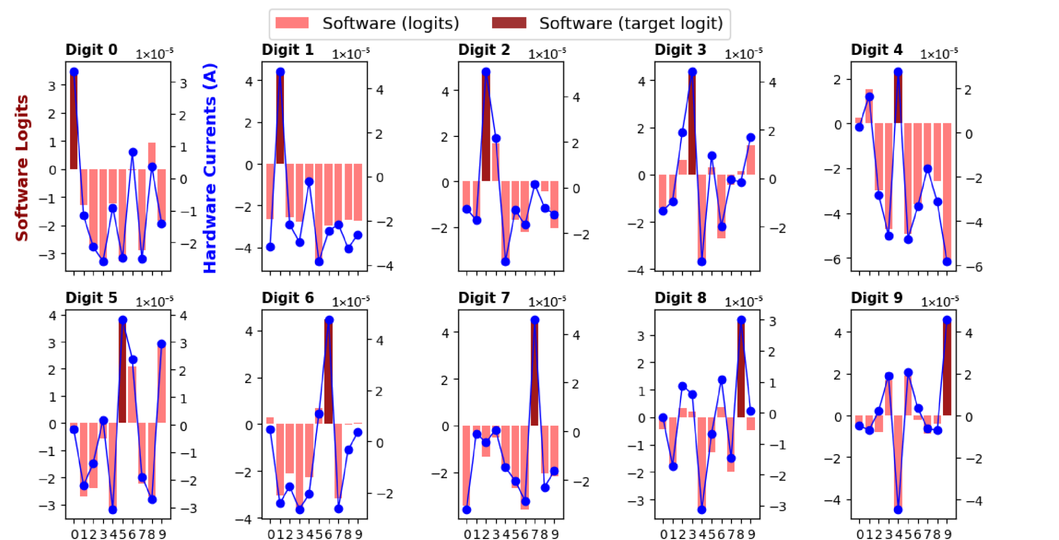

Figure 9. Inference test results achieved after NN parameters generation with Positive and Negative Weights Only (PNWO) at the software (bars) and hardware (data points) levels.

This approach, however, results in a larger area footprint, as the number of rows and columns in the crossbar array is doubled (see Figure 7). The last analysis, presented in Figure 10, is conducted for a NN model generated at the software level with Positive and Negative Weights and Biases (PNWB). Accuracy results at both the hardware and software levels show an accuracy of 100% in both cases. Qualitatively, no significant difference is observed compared to the previous configuration, indicating that the inclusion of bias has a negligible impact on inference results. From a crossbar matrix size standpoint, in addition to doubling the number of columns, two additional rows are required at the crossbar array level to accommodate positive and negative biases (see Figure 7).

4.3. Result Analysis

To evaluate prediction confidence during inference, we employ the Margin Confidence (MC) [27], defined as the margin between the largest and second-largest logits (or currents). This difference is then normalized by the highest logit (or current) in each inference result

and expressed as a percentage, providing a relative measure of the confidence of the different models. Starting from the MC metric, a global measure of the prediction confidence over the entire dataset is obtained by computing the average margin confidence MC_{avg}. This metric is computed by averaging the margin confidence MC_i, for each digit up to $N = 10$, as defined in Equation (3).

$$MC_{avg} = \frac{1}{N} \sum_{i=1}^{i=N} MC_i \quad 1 < k \leq N \tag{3}$$

MC_{avg} relative results are summarized in Table 4 for the three evaluated scenarios. A higher MC value reflects a greater prediction confidence. For the PWO scenario, the relative MC average of the software approach is 23.54%, while the hardware approach achieves an MC_{avg} of 17.50% (6.04% drop), highlighting the superior prediction efficiency of the software approach. Similarly, in the PNWO and PNWB scenarios, MC_{avg} reaches 73.12% and 72.54%, respectively, at the software level. It drops to 65.24% (7.99% drop) and 63.19% (9.35% drop), respectively, at the hardware level.

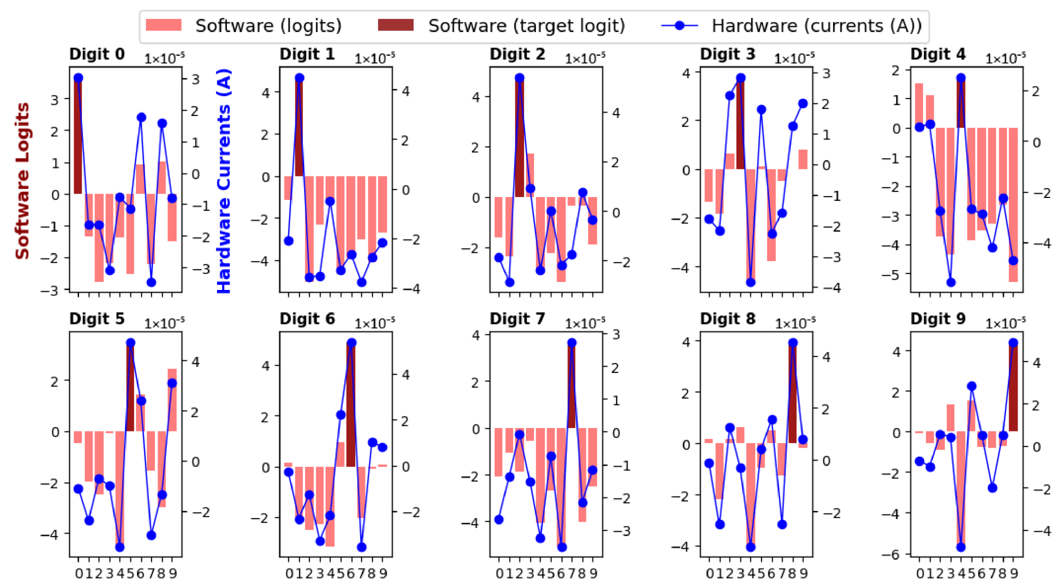

Figure 10. Inference test results achieved after NN parameters generation with Positive and Negative Weights along with Biases (PNWB) at the software (bars) and hardware (data points) level.

Table 4. Hardware & software scenario metrics.

Scenarios	PWO	PNWO	PNWB
MC_{avg}	Soft: 23.54% Hard: 17.50%	Soft: 73.12% Hard: 65.24%	Soft: 72.54% Hard: 63.19%
Current	1261 µA	2056 µA	2148 µA
Energy	37.8 pJ	61.6 pJ	63.2 pJ
Energy/MAC	189 fJ/MAC	308 fJ/MAC	316 fJ/MAC
Area	1	2	2.1

From an energy standpoint, hardware simulations reveal that the average current consumption per inference is 1261 µA for the PWO scenario while for the PNWO and PNWB scenarios, it increases to 2056 µA and 2148 µA, respectively. Energy efficiency can be assessed using various metrics, such as the total inference energy, which corresponds to the energy needed to process an entire input digit. Considering an input voltage pulse of 0.3 V with a duration of 100 ns during inference, the total energy consumption is

estimated at 37.8 pJ, 61.6 pJ, and 63.2 pJ for the PWO, PNWO, and PNWB configurations, respectively. Another way to quantify energy efficiency is through the energy consumed per MAC operation. Processing a single input sample involves 200 MAC operations (20 input neurons $\times$ 10 output neurons), yielding an energy per MAC of 189 fJ/MAC, 308 fJ/MAC, and 316 fJ/MAC for the PWO, PNWO, and PNWB scenarios, respectively (see Table 4). Note that the reported values correspond to the core crossbar energy only, excluding DAC/ADC and peripheral circuitry. The latency of the VMM engine is on the nanosecond scale, as it is primarily dictated by RC delays (i.e., resistance–capacitance effects) within the crossbar interconnects. However, since this calculation does not account for peripheral components like DACs and ADCs, the reported latency should be considered carefully, as the crossbar array produces output currents nearly instantaneously. From an area overhead perspective, the compactness of the hardware design is determined by the crossbar array size. As shown in Table 4, the PNW configuration occupies twice the area of the PWO configuration, which serves as the reference. The PNWO configuration has a slightly larger footprint, requiring 2.1 times the area of the PWO configuration.

5. Neural Network Performance Under Variability

To investigate the consistency of the predictions, variability is introduced at the hardware level and targets RRAM memory cells. As already mentioned in the introduction and according to experimental results presented in Figure 4, RRAM variability is dependent on its nominal conductance value. Assuming that resistance variations follow a statistical distribution, typically modeled as normal [28], the standard deviation of the resistance will serve as the key metric to characterize RRAM variability. Equation (4), extracted from Table 2, presents the relationship between RRAM resistance standard deviation σ and the RRAM mean resistance value μ. As resistance increases, the dispersion expands, following a quadratic law [29].

$$\sigma = 0.00000824 \, * \, \mu^2 + -0.0778270 \, * \, \mu + 663.72 \tag{4}$$

Based on Equation (4), variability is introduced at the crossbar array level as follows: (i) the resistance of each cell in the crossbar array is randomly sampled from a normal distribution with mean μ (nominal RRAM cell resistance) and standard deviation σ; (ii) the crossbar array is then simulated to compute predictions; (iii) this process is repeated over multiple trials to determine the percentage of trials in which the predictions deviate from the correct predictions referred to as Cp. Finally, the robustness R of the prediction is computed using Equation (5).

$$R = \frac{number\ of\ Cp}{Total\ Trials} * 100 \tag{5}$$

Note that variability is automatically generated and integrated into the crossbar array using programming scripts within the *Cadence* environment. More specifically, variability injection is based on an RRAM model calibrated on silicon, ensuring realistic and accurate simulation results.

Figure 11a illustrates the robustness (R) of predictions for the three different scenarios (PWO, PNWO, and PNWB) across 1000 inference trials. The PWO scenario achieves the highest robustness (94.65%), followed by PNWO (93.11%) and PNWB (92.11%), indicating a degradation in prediction performance versus variability when negative weights and biases are used. Figure 11b–d highlight inference errors across the digits (0–9) for each scenario. All scenarios exhibit significantly higher errors for digits '4' and '5'. The PNWB scenario

shows the highest number of errors for digit '4' (489 errors), indicating a pronounced sensitivity to this digit under that scenario. The PWO scenario maintains relatively lower error rates across all digits, demonstrating its superior robustness. The PNWO scenario, while worse than PWO but better than PNWB, follows a similar error distribution.

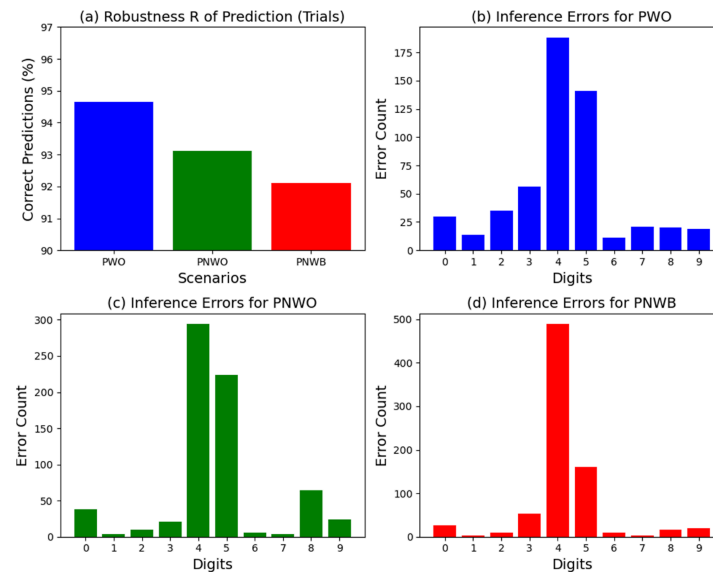

Figure 11. (**a**) Robustness R of predictions for the three different scenarios (PWO, PNWO, and PNWB). Inference errors across all the digits (0–9) for scenario (**b**) PWO, (**c**) PNWO, and (**d**) PNWB.

6. Discussion

The proposed analysis focuses on evaluating the inference confidence, energy efficiency, robustness, and variability of neural network models implemented in hardware for different weight configuration flavors (PWO, PNWO, and PNWB) generated at the software level. From an inference prediction standpoint, the MC_{avg} metric reveals a systematic decrease in prediction confidence when moving from software to hardware implementations. The software approach outperforms the hardware-based approach, with the latter exhibiting a decrease in MC_{avg} ranging from 6.04% to 9.35% across the different scenarios. Hardware inference remains effective while showing a difference with the lowest prediction confidence observed for PWO (17.50%). Interestingly, MC_{avg} is higher for PNWO (65.24%) compared to PNWB (63.19%), suggesting that integrating software biases at the hardware level does not enhance prediction accuracy. From an energy efficiency standpoint, the current consumption increases significantly when transitioning from PWO (1261 μA) to PNWO (2056 μA) and PNWB (2148 μA), since the introduction of negative weights and biases increases circuit area by at least a factor of 2 as well as circuit complexity due to the need to handle multiple positive and negative signals. Energy consumption per inference follows a similar trend, rising from 37.8 pJ (PWO) to 61.6 pJ (PNWO) and 63.2 pJ (PNWB).

Regarding the prediction robustness, we assessed the impact of RRAM variability on classification performance. Remarkably, the prediction robustness decreases as we move from PWO (94.65%) to PNWO (93.11%) and further to PNWB (92.11%). This trend shows that weight complexity can lead to a higher number of RRAM cells at the hardware level, which, in turn, increases variability-induced errors [30]. The PWO scenario provides lower error rates, reinforcing its greater robustness against hardware-induced variability. Latency was not deeply investigated. Indeed, while crossbar arrays enable near-instantaneous current generation, delays are mainly related to peripheral circuits including DACs and ADCs. These conversion stages, not addressed in this study, are necessary to interface memristor-based MAC operations' analog results with digital blocks.

Additionally, DAC and ADC power consumption scales exponentially with their resolution, leading to significant energy costs when higher conversion precision is required [31]. It has also been demonstrated in [32] that the power-hungry nature of ADCs and DACs can significantly offset the efficiency gains of crossbar-based acceleration [33], reinforcing the need to optimize bit precision to balance accuracy, latency, and power efficiency.

Although prediction robustness was evaluated against variability, it is important to note that additional factors can further degrade the performance of RRAM-based crossbar arrays during inference. Owing to the specific manufacturing processes and switching mechanisms, RRAM devices are vulnerable to distinct faults such as over-forming and undefined state faults [34,35]. Consequently, it is crucial to establish a comprehensive fault analysis methodology [36] for this memory class at the crossbar array level. Furthermore, RRAM-based neural networks show promise for deployment in harsh spatial environments [37]. However, ensuring reliable operation in such conditions requires addressing challenges like exposure to high levels of radiation, including ionizing particles [38,39]. Finally, from a system-level perspective, the results highlight a fundamental trade-off between accuracy, energy efficiency, and hardware complexity in RRAM-based neural network implementations.

The results indicate that future research should prioritize co-design strategies that jointly optimize neural network architectures, quantization schemes [40], and hardware constraints [2]. In particular, adaptive precision techniques [41], variability-aware training [42], and circuit-level compensation mechanisms [43] could play a key role in bridging the gap between software performance and hardware efficiency. Such approaches are essential to fully exploit the potential of RRAM-based accelerators for edge AI applications, where constraints on power, latency, and reliability must be simultaneously satisfied [44]. Furthermore, the underlying switching mechanism of RRAM devices is a key factor influencing the performance of crossbar-based neural networks. While this work focuses on filamentary switching, alternative approaches relying on carrier trapping and detrapping at defect sites have been extensively reported [45,46]. These electronic-based mechanisms are generally associated with enhanced stability, lower variability, and improved endurance compared to filamentary devices, which can lead to more reliable and accurate analog in-memory computing. In parallel, novel material systems, including halide perovskite memristors, are being explored to further improve device performance and scalability [47]. Therefore, both the switching mechanism and the choice of material constitute critical design parameters [48] that directly impact the accuracy–energy–reliability trade-offs in RRAM-based crossbar architectures [49,50].

7. Conclusions

RRAM crossbar arrays naturally perform VMM in parallel through the principles of Ohm's Law and Kirchhoff's Current Law, enabling highly efficient data processing. In this context, this study evaluates the potential of employing RRAM devices as synapses within crossbar arrays used as MAC accelerators for image processing tasks. Based on the constraints applied to the neural network parameters at the software level, the presented results highlight a clear trade-off at the hardware level between prediction confidence, energy efficiency, robustness, and hardware complexity. While software implementations achieve higher prediction confidence, hardware implementations are limited by the RRAM conductance range and variability. Constraining weights and biases to non-negative values (PWO scenario) simplifies hardware implementation and reduces energy consumption but comes at the cost of reduced classification accuracy. However, in terms of robustness against RRAM variability, the PWO scenario has demonstrated greater resilience. In contrast, allowing both positive and negative values in the model parameters enhances

accuracy but increases energy consumption, design complexity, and area requirements, while lowering resilience to variability. These findings are essential for optimizing future hardware-accelerated neural networks leveraging RRAM technology.

Author Contributions: Conceptualization, H.A., H.X. and M.F.; formal analysis H.A., H.X. and M.F., methodology, H.A.; project administration, H.A.; supervision, H.A.; writing—original draft, H.A.; writing—review and editing, H.A., H.X., M.T. and S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pomsar, L.; Brecko, A.; Zolotova, I. Brief overview of Edge AI accelerators for energy-constrained edge. In *2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI)*; IEEE: New York, NY, USA, 2022; pp. 000461–000466. [CrossRef]
2. Shooshtari, M.; Serrano-Gotarredona, T.; Linares-Barranco, B. Review of Memristors for In-Memory Computing and Spiking Neural Networks. *Adv. Intell. Syst.* **2025**, *8*, e202500806. [CrossRef]
3. Aziza, H.; Moreau, M.; Perez, A.; Virazel, A.; Girard, P. A Capacitor-Less CMOS Neuron Circuit for Neuromemristive Networks. In *2019 17th IEEE International New Circuits and Systems Conference (NEWCAS)*; IEEE: New York, NY, USA, 2019; pp. 1–4. [CrossRef]
4. Chadha, A.; Bi, Y.; Abbas, A.; Andreopoulos, Y. Neuromorphic Vision Sensing for CNN-based Action Recognition. In *ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: New York, NY, USA, 2019; pp. 7968–7972. [CrossRef]
5. Lee, Y.-L.; Tsung, P.-K.; Wu, M. Technology trend of edge AI. In *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*; IEEE: New York, NY, USA, 2018. [CrossRef]
6. Shirinzadeh, S.; Soeken, M.; Gaillardon, P.-E.; Drechsler, R. Fast Logic Synthesis for RRAM-based In-Memory Computing using Majority-Inverter Graphs. In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: New York, NY, USA, 2016; pp. 948–953. [CrossRef]
7. Barlas, M.; Grossi, A.; Grenouillet, L.; Vianello, E.; Nolot, E.; Vaxelaire, N.; Blaise, P.; Traore, B.; Coignus, J.; Perrin, F.; et al. Improvement of HfO_2-based RRAM array performances by local Si implantation. In *2017 IEEE International Electron Devices Meeting (IEDM)*; IEEE: New York, NY, USA, 2017; pp. 14.6.1–14.6.4. [CrossRef]
8. Aziza, H.; Hamdioui, S.; Fieback, M.; Taouil, M.; Moreau, M. Density Enhancement of RRAMs using a RESET Write Termination for MLC Operation. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: New York, NY, USA, 2021; pp. 1877–1880. [CrossRef]
9. Lee, M.K.; Jeong, J.-S.; Jung, J.; Seo, H.K.; Shin, C.; Kim, G.H.; Yang, M.K. Fast and energy-efficient resistive switching in ZrOx/TaOx bilayer structures under ultra-low voltage operation for next-generation memory applications. *Adv. Compos. Hybrid Mater.* **2026**, *9*, 143. [CrossRef]
10. Papandroulidakis, G.; Vourkas, I.; Abusleme, A.; Sirakoulis, G.C.; Rubio, A. Crossbar-Based Memristive Logic-in-Memory Architecture. *IEEE Trans. Nanotechnol.* **2017**, *16*, 491–501. [CrossRef]
11. Chou, T.; Tang, W.; Botimer, J.; Zhang, Z. CASCADE. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*; ACM: New York, NY, USA, 2019; pp. 114–125. [CrossRef]
12. Ahmad, R.W.; Wouters, D.; Bengel, C.; Waser, R.; Menzel, S. Analysis of VMM Operations on 1S1R Crossbar Arrays and the Influence of Wire Resistances. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*; IEEE: New York, NY, USA, 2022; pp. 91–95. [CrossRef]
13. Postel-Pellerin, J. True random number generation exploiting SET voltage variability in resistive RAM memory arrays. In *2019 19th Non-Volatile Memory Technology Symposium (NVMTS)*; IEEE: New York, NY, USA, 2019; pp. 1–5. [CrossRef]
14. Bhardwaj, K.; Paasio, E.; Majumdar, S. Toward Capacitive In-Memory-Computing: A Device to Systems Level Perspective on the Future of Artificial Intelligence Hardware. *Adv. Intell. Discov.* **2025**, e202500143. [CrossRef]
15. Wu, L.; Yu, Z.; Qin, Y.; Chen, Q.; Cai, Y.; Huang, R. Nonlinear Weight Quantification for Mitigating Stress Induced Disturb Effect on Multilevel RRAM-Based Neural Network Accelerator. *IEEE J. Electron Devices Soc.* **2021**, *9*, 1257–1261. [CrossRef]
16. Bhardwaj, K.; Semenov, D.; Sotner, R.; Majumdar, S. Preventing False Activations in Autonomous Vehicles: A Memristive Associative Learning Approach with Selective Sensor Pairing. In *2025 14th International Conference on Modern Circuits and Systems Technologies (MOCAST)*; IEEE: New York, NY, USA, 2025; pp. 1–4. [CrossRef]

17. Zidan, M.A.; Fahmy, H.A.H.; Hussain, M.M.; Salama, K.N. Memristor-based memory: The sneak paths problem and solutions. *Microelectron. J.* **2013**, *44*, 176–183. [CrossRef]
18. Lorenzi, P.; Rao, R.; Irrera, F. Forming kinetics in HfO_2-based RRAM cells. *IEEE Trans. Electron Devices* **2013**, *60*, 438–443. [CrossRef]
19. Zahoor, F. A comprehensive study on resistive random access memory based true random number generators. *Comput. Electr. Eng.* **2026**, *130*, 110901. [CrossRef]
20. Kim, W.; Menzel, S.; Wouters, D.J.; Waser, R.; Rana, V. 3-Bit Multilevel Switching by Deep Reset Phenomenon in Pt/W/TaOX/Pt-ReRAM Devices. *IEEE Electron Device Lett.* **2016**, *37*, 564–567. [CrossRef]
21. Ielmini, D. Resistive switching memories based on metal oxides: Mechanisms, reliability and scaling. *Semicond. Sci. Technol.* **2016**, *31*, 063002. [CrossRef]
22. Zhao, L.; Chen, H.-Y.; Wu, S.-C.; Jiang, Z.; Yu, S.; Hou, T.-H.; Wong, H.-S.P.; Nishi, Y. Improved multi-level control of RRAM using pulse-train programming. In *Proceedings of Technical Program—2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*; IEEE: New York, NY, USA, 2014; pp. 1–2. [CrossRef]
23. Aziza, H.; Moreau, M.; Fieback, M.; Taouil, M.; Hamdioui, S. An Energy-Efficient Current-Controlled Write and Read Scheme for Resistive RAMs (RRAMs). *IEEE Access* **2020**, *8*, 137263–137274. [CrossRef]
24. Aguirre, F.; Sebastian, A.; Le Gallo, M.; Song, W.; Wang, T.; Yang, J.J.; Lu, W.; Chang, M.-F.; Ielmini, D.; Yang, Y.; et al. Hardware implementation of memristor-based artificial neural networks. *Nat. Commun.* **2024**, *15*, 1974. [CrossRef]
25. Hajri, B.; Mansour, M.M. Oxide-based RRAM models for circuit designers: A comparative analysis. In *2017 12th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*; IEEE: New York, NY, USA, 2017. [CrossRef]
26. Aziza, H. Image Classification in Memristor-Based Neural Networks: A Comparative Study of Software and Hardware Models Using RRAM Crossbars. *Electronics* **2025**, *14*, 1125. [CrossRef]
27. Hu, M.; Strachan, J.P.; Li, Z.; Grafals, E.M.; Davila, N.; Graves, C.; Lam, S.; Ge, N.; Yang, J.J.; Williams, R.S. Dot-product engine for neuromorphic computing. In *Proceedings of the 53rd Annual Design Automation Conference*; ACM: New York, NY, USA, 2016; pp. 1–6. [CrossRef]
28. Malhotra, A.; Lu, S.; Yang, K.; Sengupta, A. Exploiting Oxide Based Resistive RAM Variability for Bayesian Neural Network Hardware Design. *IEEE Trans. Nanotechnol.* **2020**, *19*, 328–331. [CrossRef]
29. García, H.; Ossorio, O.G.; Dueñas, S.; Castán, H. Controlling the intermediate conductance states in RRAM devices for synaptic applications. *Microelectron. Eng.* **2019**, *215*, 110984. [CrossRef]
30. Aziza, H.; Bocquet, M.; Portal, J.-M.; Muller, C. Evaluation of OxRAM cell variability impact on memory performances through electrical simulations. In *2011 11th Annual Non-Volatile Memory Technology Symposium Proceeding*; IEEE: New York, NY, USA, 2011; pp. 1–5. [CrossRef]
31. Singh, J. Implementation of Memristor Towards Better Hardware/Software Security Design. *Trans. Electr. Electron. Mater.* **2021**, *22*, 10–22. [CrossRef]
32. Ibrayev, T.; Garg, I.; Chakraborty, I.; Roy, K. Pruning for Improved ADC Efficiency in Crossbar-based Analog In-memory Accelerators. *arXiv* **2024**, arXiv:2403.13082. [CrossRef]
33. Firdauzi, A.; Grewing, C.; Ashok, A.; Kusuma, S.; Winterberg, K.; Zambanini, A.; Van Waasen, S. Power Efficient Current-Mode SAR ADC for Memristor Readout in 28 nm CMOS. In *2024 31st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*; IEEE: New York, NY, USA, 2024; pp. 1–4. [CrossRef]
34. Fieback, M.; Medeiros, G.C.; Wu, L.; Aziza, H.; Bishnoi, R.; Taouil, M.; Hamdioui, S. Defects, Fault Modeling, and Test Development Framework for RRAMs. *ACM J. Emerg. Technol. Comput. Syst.* **2022**, *18*, 52. [CrossRef]
35. Aziza, H.; Bocquet, M.; Portal, J.-M.; Muller, C. Bipolar OxRRAM memory array reliability evaluation based on fault injection. In *2011 IEEE 6th International Design and Test Workshop (IDT)*; IEEE: New York, NY, USA, 2011; pp. 78–81. [CrossRef]
36. Garcia-Redondo, F.; Lopez-Vallejo, M. On the Design and Analysis of Reliable RRAM-CMOS Hybrid Circuits. *IEEE Trans. Nanotechnol.* **2017**, *16*, 514–522. [CrossRef]
37. Vaz, P.I.; Girard, P.; Virazel, A.; Aziza, H. Improving TID Radiation Robustness of a CMOS OxRAM-Based Neuron Circuit by Using Enclosed Layout Transistors. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 1122–1131. [CrossRef]
38. Castellani-Coulie, K.; Aziza, H.; Rahajandraibe, W.; Micolau, G.; Portal, J.-M. Development of a CMOS Oscillator Concept for Particle Detection and Tracking. *IEEE Trans. Nucl. Sci.* **2013**, *60*, 2450–2455. [CrossRef]
39. Lyu, H.; Zhang, H.; Mei, B.; Yu, Q.; Mo, R.; Sun, Y.; Gao, W. Research on single event effect test of a RRAM memory and space flight demonstration. *Microelectron. Reliab.* **2021**, *126*, 114347. [CrossRef]
40. Park, K.; Kim, S.; Park, J.-H.; Choi, W.Y. A Quantized-Weight-Splitting Method of RRAM Arrays for Neuromorphic Applications. *IEEE Access* **2024**, *12*, 59680–59687. [CrossRef]
41. Khan, M.F.F.; Jao, N.A.; Shuai, C.; Qiu, K.; Mahdavi, M.; Narayanan, V. Mixed Precision Quantization Scheme for Re-configurable ReRAM Crossbars Targeting Different Energy Harvesting Scenarios. In *Internet of Things. A Confluence of Many Disciplines*; Springer: Cham, Switzerland, 2020; pp. 197–216. [CrossRef]

42. Deng, Z.; Orshansky, M. Variability-Aware Training and Self-Tuning of Highly Quantized DNNs for Analog PIM. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*; IEEE: New York, NY, USA, 2022; pp. 712–717. [CrossRef]
43. Freire, P.; Srivallapanondh, S.; Spinnler, B.; Napoli, A.; Costa, N.; Prilepsky, J.E.; Turitsyn, S.K. Computational Complexity Optimization of Neural Network-Based Equalizers in Digital Signal Processing: A Comprehensive Approach. *J. Light. Technol.* **2024**, *42*, 4177–4201. [CrossRef]
44. Shooshtari, M.; Kim, S.; Pahlavan, S.; Rivera-Sierra, G.; Través, M.J.; Serrano-Gotarredona, T.; Bisquert, J.; Linares-Barranco, B. Advancing Logic Circuits with Halide Perovskite Memristors for Next-Generation Digital Systems. *SmartMat* **2025**, *6*, e70032. [CrossRef]
45. Hu, L.; Yang, J.; Wang, J.; Cheng, P.; Chua, L.O.; Zhuge, F. All-optically controlled memristor for optoelectronic neuromorphic computing. *Adv. Funct. Mater.* **2021**, *31*, 2005582. [CrossRef]
46. Yoon, J.H.; Song, S.J.; Yoo, I.; Seok, J.Y.; Yoon, K.J.; Kwon, D.E.; Park, T.H.; Hwang, C.S. Highly Uniform, Electroforming-Free, and Self-Rectifying Resistive Memory in the $Pt/Ta_2O_5/HfO_2$-x/TiN Structure. *Adv. Funct. Mater.* **2014**, *24*, 5086–5095. [CrossRef]
47. Mao, M.; Cao, Y.; Yu, S.; Chakrabarti, C. Optimizing latency, energy, and reliability of 1T1R ReRAM through appropriate voltage settings. In *2015 33rd IEEE International Conference on Computer Design (ICCD)*; IEEE: New York, NY, USA, 2015; pp. 359–366. [CrossRef]
48. Prezioso, M.; Merrikh-Bayat, F.; Hoskins, B.D.; Adam, G.C.; Likharev, K.K.; Strukov, D.B. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **2015**, *521*, 61–64. [CrossRef]
49. Roy, S.K.; Shanbhag, N.R. Energy-Accuracy Trade-Offs for Resistive In-Memory Computing Architectures. *IEEE J. Explor. Solid-State Comput. Devices Circuits* **2024**, *10*, 22–30. [CrossRef]
50. Tyagi, A.; Sahay, S. Assessing the Performance of Reinforcement Learning on Passive RRAM Crossbar Array. *TechRxiv* **2023**. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

https://doi.org/10.3390/electronics15081710