FULLY HARDWARE ANALOG COMPUTING SYSTEMS USING ARRAYS OF MEMRISTOR DEVICES

FULLY HARDWARE ANALOG COMPUTING SYSTEMS USING ARRAYS OF MEMRISTOR DEVICES

A Dissertation Presented

By

FATEMEH KIANI

Submitted to the Graduate School of the

University of Massachusetts Amherst in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2024

Electrical and Computer Engineering

FULLY HARDWARE ANALOG COMPUTING SYSTEMS USING ARRAYS OF MEMRISTOR DEVICES

A Dissertation Presented

By

FATEMEH KIANI

Approved as to style and content by:

Qiangfei Xia, Chair

Yeon Sik Noh, Member

Jianhua Joshua Yang, Outside Member

Christopher V. Hollot, Department Head

Electrical and Computer Engineering

ACKNOWLEDGMENTS

First, I would like to thank my advisor professor Qiangfei Xia whose mentorship, consistent presence, and knowledgeable feedback was the main contributing factor in successfully completing every stage of my work during my Ph.D. studies. I also express my gratitude to professors J. Joshua Yang and Zhongrui Wang whose constructive comments and feedback during my Ph.D. research helped move forward my research considerably. I also appreciate our collaborators from Duke University, Professor Hai Li, for her constructive feedback. I am also grateful for the contribution of my colleagues at the University of Massachusetts Amherst including Yi Huang, Jun Yin, Fan Ye, and Alireza Jaberirad.

Finally, I am truly grateful for my family, my husband Shayan whose presence and support made this journey enlightening, and my parents, sister, and brothers who mentally empowered me in this journey from far away.

ABSTRACT

FULLY HARDWARE ANALOG COMPUTING SYSTEMS USING ARRAYS OF MEMRISTOR DEVICES

MAY2024

FATEMEH KIANI

B.Sc., UNIVERSITY OF TEHRAN

M.Sc., UNIVERSITY OF TEHRAN

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Qiangfei Xia

A memristor device is an electrical component whose internal state depends on past current and/or voltage. It is highly sought-after for various applications due to its non-volatile nature, retaining memory even without power. Integrating memristor devices into crossbar arrays offers significant improvements in throughput and power through in-memory analog computing.

The implementation of hardware-based perceptron using memristor arrays has gained substantial interest recently. However, previous demonstrations often relied on software or digital processors for critical functionalities like activation functions which led to frequent analog/digital conversions, extensive back-and-forth communication, and more complex hardware, compromising energy efficiency and computing parallelism.

In these implementations, memristor arrays are widely used in multiply-accumulating (MAC) operations while performing parallel vector-matrix-multiplication (VMM). However, achieving VMM operations requires unrolling two-dimensional (2D) inputs into one-dimensional (1D) vectors, requiring data preprocessing and additional computing resources and time.

This thesis proposes a novel fully hardware-based activation function designed for implementation between different layers of a perceptron. This innovative approach enables the transmission of analog signals to the subsequent layer without unnecessary digital conversion, communication, or processing. Consequently, the system experiences improved power efficiency and throughput by eliminating power-intensive peripheral circuitry and minimizing unnecessary data movements.

Compact rectified linear units (ReLUs) and memristor arrays serve as the building blocks of a two-layer perceptron. Experimental results demonstrate a recognition accuracy of 93.63% for the Modified National Institute of Standard and Technology (MNIST) dataset, comparable to its software-based counterpart. Adopting a fully hardware-based neural network minimizes both data shuttling and conversion, significantly enhances computing throughput and power efficiency.

Finally, an innovative three-dimensional (3D) memristor array architecture is proposed to perform matrix-matrix pointwise multiplication, directly multiplying 2D inputs with weight matrices without the need for data unrolling. This yields notable improvements in both energy efficiency and computing throughput. The proposed system finds applications in various domains, such as classification and edge detection in a single iteration. The proposed architecture achieves a recognition accuracy of 94.36% in classifying the MNIST dataset using MATLAB and LTspice. Furthermore, the complete system is designed with printed circuit boards to validate its functionality as a proof-of-concept for feature extraction tasks.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Neural networks, complicated computational models inspired by the structure and function of human brain, find extensive applications in industry, business, and science, profoundly influencing various aspects of people's everyday lives [1-3]. These networks play a pivotal role in diverse applications, ranging from image and speech recognition to natural language processing and autonomous vehicles. Their growing utilization underscores the continuous demand for innovative approaches to enhance power efficiency and computing throughput [4-7].

As the demand for more efficient computing solutions intensifies, researchers and engineers are exploring novel approaches to meet these needs. One promising avenue involves leveraging emerging memory technologies, with particular attention directed towards memristor devices. Memristors exhibit unique characteristics that enable them to perform both computing and data storage functions simultaneously [7-11]. This integration of functionalities presents a paradigm shift in computing architecture, offering the potential for significant performance enhancements. These devices have proven successful in hardware accelerators for machine learning [12-26]. When organized into crossbar arrays, a configuration where rows and columns intersect to form memory cells, researchers have demonstrated the capability to execute the VMM, a computationally intensive operation, in a single step, resulting in significantly higher computing throughput, irrespective of the array size [27, 28]. This advancement holds the promise of revolutionizing the landscape of computing, paving the way for more efficient and powerful neural network implementations.

The VMM is achieved through analog computing, leveraging physical laws such as Ohm's law for multiplications and Kirchhoff's current law for summations [16-18, 28], performed at the same site where the data are stored. This in-memory computing paradigm substantially reduces power consumption by eliminating the constant data shuttling required in traditional von Neumann architecture [24, 28-30]. While prior work successfully employed the analog multilevel resistance of memristor devices as synaptic weights in a neural network, most efforts still relied, at least partially, on software or digital processors to implement hidden neurons [16-18, 25]. Consequently, there remained frequent analog/digital (A/D) data conversion and back-and-forth data communication during computing, resulting in inefficient resource utilization.

To fully unlock the potential of emerging accelerators, there is a critical need for a hardware-based neural network with efficient analog-based hardware hidden neurons [26, 31]. Previous attempts towards this goal have been constrained by extremely small-scale arrays, substantial circuit area requirements for hardware hidden neurons, and low system speed [32-34]. Activation functions play a crucial role in hidden neurons, with various types being employed in neural networks [35-37]. Among these, ReLU stands out as the most desirable due to its numerous advantages, including easier optimization, faster convergence, greater computational efficiency, and high speed [38-44].

ReLUs are characterized by a single parameter, their gain, making them easier to optimize. They exhibit rapid convergence, devoid of vanishing problems, ensuring that the slope does not saturate as the input increases. With high computational efficiency, ReLUs contribute to faster training and testing of networks, given their straightforward operations, outputting zero for negative inputs and linearly proportional values for positive inputs. Notably, to the best of our knowledge, there has been no experimental implementation of ReLU on hardware using solely analog components.

In previous demonstrations of accelerators, memristors were employed for parallel VMM [45-49]. However, in these systems, the inputs require conditioning, meaning they need to be converted into 1D vectors suitable for VMM with the weight matrices. Consequently, the use of additional computing resources, such as digital processing units, becomes unavoidable, leading to extra energy consumption and delays that hinder energy efficiency and real-time computation. To achieve higher parallelism and a denser architecture, previous research efforts have explored the design of 3D arrays to enable a greater amount of parallel computation [50, 51]. Nevertheless, these approaches still necessitate the unrolling of input data, introducing additional computing resources and time requirements. Notably, to our knowledge, there has been no implementation of 3D arrays that circumvent the need for unrolling inputs, thereby enabling direct processing of input data from sensor arrays. This advancement enables higher parallelism and improved performance in terms of energy efficiency and throughput.

1.1 Fully Hardware Memristor-based Multilayer Neural Network

In a recent development, we successfully designed and implemented a compact multichannel ReLU using off-the-shelf or discrete analog components. Additionally, we constructed a two-layer fully hardware-based perceptron featuring 64 ReLUs as hidden neurons, connecting two 128×64 memristor crossbar arrays. Through this configuration, we achieved an impressive 93.63% recognition accuracy in classifying MNIST [52] images.

Comparatively, our analog-based fully hardware implementation holds significant advantages over solutions employing partial software hidden neurons. The removal of unnecessary data conversion, back-and-forth communication, and the elimination of power-hungry and bulky peripheral circuitry contribute to superior performance in terms of power efficiency, reduced area requirements, and lower latency.

Research Contribution: The design and development of ReLU contribute to a neural network architecture that is more power-efficient, exhibits high throughput, and is compact in nature. These attributes hold substantial benefits, particularly in the context of deep neural networks that demand hundreds of layers. The efficiency gains achieved through ReLU design are crucial in enhancing the overall performance of such complicated neural network structures.

1.2 Matrix-matrix Pointwise Multiplication with Three-dimensional Memristor Arrays

We introduced a novel 3D memristor array architecture capable of performing matrix-matrix pointwise multiplication in a single cycle or one-shot operation. This architecture exhibits superior parallelism, more efficient power consumption, and enhanced computing throughput compared to state-of-the-art alternatives. Utilizing MATLAB and LTspice, we modeled the architecture and conducted benchmarking by implementing a CNN. Our evaluation included tasks such as image classification and feature extraction on the MNIST dataset.

Furthermore, we translated our proposed design into a practical implementation by experimentally realizing the full system on PCBs. This involved the use of off-the-shelf components like Digital to Analog Converters (DACs), operational amplifiers, digital potentiometers, Analog to Digital Converters (ADCs), and others. Multiple tests were conducted to demonstrate the proof-of-concept and validate the effectiveness of our innovative architecture.

Finally, we conducted an assessment of resource utilization for the implementation of our proposed system in an integrated version. Specifically, we provided data on energy consumption projecting to the integrated version of our system within the 16 nm FinFet technology node.

Research Contribution: The innovative 3D array architecture we propose offers higher parallelism and computing throughput, coupled with improved energy consumption. This is achieved by facilitating matrix-matrix pointwise multiplication in a single operation within the array, enhancing overall efficiency.

1.3 Dissertation Outline

In Chapter 2, we describe the details of our developed fully analog ReLU circuits, explaining their characteristics and their application as hidden neurons in constructing a perceptron for MNIST image recognition.

Chapter 3 provides an in-depth exploration of our proposed 3D array architecture, including its modeling using MATLAB and LTspice. Additionally, we present the outcomes of employing this architecture in implementing a CNN for MNIST image recognition.

Moving on to Chapter 4, we provide a comprehensive account of the design and experimental implementation of the proposed 3D array architecture. Utilizing variable resistors, or potentiometers, to represent memristors, we devised peripheral circuitry to facilitate matrix-matrix pointwise multiplication. Notably, our design is adaptable to incorporate the use of our developed fully analog ReLUs, offering flexibility for handling more complex applications in the future. Additionally, this chapter offers valuable insights into the utilization of memristor devices and provides estimations for energy consumption of the envisioned system when implemented in the integrated circuit design version.

In Chapter 5, we present a comprehensive summary of the topics explored in this dissertation and describe prospective avenues for future research.

1.4 Research Publications to Date

The research results presented in this dissertation proposal have been published in the venues listed below:

1. F. Kiani, J. Yin, Z. Wang, J.J. Yang, and Q. Xia, "All hardware-based two-layer perceptron implemented in memristor crossbar arrays," In IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2021. [53]
2. F. Kiani, J. Yin, Z. Wang, J.J. Yang, and Q. Xia, "A fully hardware-based memristive multilayer neural network," Science advances, vol. 7, no. 48, pp. eabj4801, Nov. 2021. [48]

3. F. Kiani, and J. Yin, "A fully hardware-based memristive multilayer perceptron," The International Conference on Memristive Materials, Devices & Systems (MEMRISYS’22), Nov 2022. [126]
4. F. Kiani, A. Jaberirad, Y. Huang, and Q. Xia, "Matrix-matrix pointwise multiplication with Three-dimensional Memristor Arrays," In Preparation. [54]

1.5 Other Publications not Covered in this Dissertation

The following publications are not covered in this dissertation.

1. X. Zhang, Y. Zhuo, Q. Luo, Z. Wu, R. Midya, Z. Wang, W. Song, R. Wang, N.K. Upadhyay, Y. Fang, F. Kiani, M. Rao, Y. Yang, Q. Xia, Q. Liu, M. Liu, and J. J. Yang, "An artificial spiking afferent nerve based on Mott memristors for neurorobotics," Nature communications, vol. 11, no. 1, pp. 51, Jan. 2020. [127]
2. M. Rao, W. Song, F. Kiani, S. Asapu, Y. Zhuo, R. Midya, N. Upadhyay, Q. Wu, M. Barnell, P. Lin, C. Li, Z. Wang, Q. Xia, and J. J. Yang, "Timing selector: Using transient switching dynamics to solve the sneak path issue of crossbar arrays," Small Science, vol. 2, no. 1, pp. 2100072, Oct. 2021. [128]
3. F. Ye, F. Kiani, Y. Huang, and Q. Xia, "Diffusive memristors with uniform and tunable relaxation time for spike generation in event-based pattern recognition," Advanced Materials, vol. 35, no. 37, pp. 2204778, Sep. 2023. [129]
4. F. Ye, and F. Kiani, "Hierarchy of Event-Based Time-Surfaces Based on Diffusive Memristors with Uniform and Tunable Relaxation Time-A Preliminary Study," The International Conference on Memristive Materials, Devices & Systems (MEMRISYS’22), Nov. 2022. [130]

5. Y. Huang, F. Kiani, F. Ye, and Q. Xia, "From memristive devices to neuromorphic systems," Applied Physics Letters, vol. 122, no. 11, pp. 110501, Mar. 2023. [131]
6. M. Rao, H. Tang, J. B. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang, H. Liu, H. Y. Chen, R. Midya, F. Ye, H. Jiang, Z. Wang, M. Wu, M. Hu, H. Wang, Q. Xia, N. Ge, J. Li, and J. J. Yang, "Thousands of conductance levels in memristors monolithically integrated on CMOS," Nature, vol. 615, no. 7954, pp. 823-829, Mar. 2023. [132]
7. F. Kiani, and Q. Xia, "Bipolar flash for bifunctional computing operations," Nature Nanotechnology, vol. 18, no. 5, pp. 444-445, May 2023. [133]

CHAPTER 2

FULLY HARDWARE MEMRISTOR-BASED MULTILAYER NEURAL NETWORK

In this chapter, we present a detailed explanation of the design and implementation of a two-layer perceptron, comprising two layers of fully connected neural networks. The synapses of these neural networks are represented using two memristor arrays, while our designed fully analog ReLUs are employed to depict the hidden neurons. This chapter elaborates on the memristors, the designed analog ReLUs, and their respective characteristics.

Furthermore, the utilization of the MNIST images dataset is employed as a benchmark for assessing functionality. The training and inference methodologies are thoroughly explained, and the classification results are meticulously reported. Additionally, we explored nonidealities and investigated resource estimation, particularly focusing on the utilization of the 65 nm technology node.

2.1 Design and Construction of the Hardware

Our two-layer perceptron is constructed with two memristor crossbar arrays, representing the matrices of synaptic weights for each layer, and ReLUs serving as activation functions in between, as illustrated in Figure 2.1A. The analog signal, generated by an in-house peripheral circuitry, is fed into the first crossbar array. The analog output, which is the weighted sum along the columns, is then directed to the hidden neurons, as depicted in Figure 2.1B. Subsequently, the signal after passing through the ReLUs becomes the input for the second crossbar array.

To enable both positive and negative synaptic weights, we implement differential pairs, utilizing both the original and the inverted signals from the ReLUs, as shown in Figure 2.1C. In this study, we employed two individual chips, each containing 128×64 memristor crossbar arrays in a one-transistor-one-resistance switch (1T1R) architecture [55-57], to execute VMM operations in the two layers. The fabrication of the chips involved the back-end-of-the-line (BEOL) integration of tantalum oxide-based memristors in our university lab, with foundry-made transistor arrays. A detailed explanation of the fabrication process is provided in Section 2.6, and the electrical performance of individual cells and arrays, along with conductance tuning schemes, is illustrated in Figure 2.8.

To connect the two 1T1R arrays, we employed a stack of PCBs hosting a total of 64 ReLU channels and 64 inverters, constructed using off-the-shelf analog components, as demonstrated in Figures 2.1D-F. The layout of the PCBs is presented in Figure 2.9 and is elaborated upon in detail in Section 2.7.

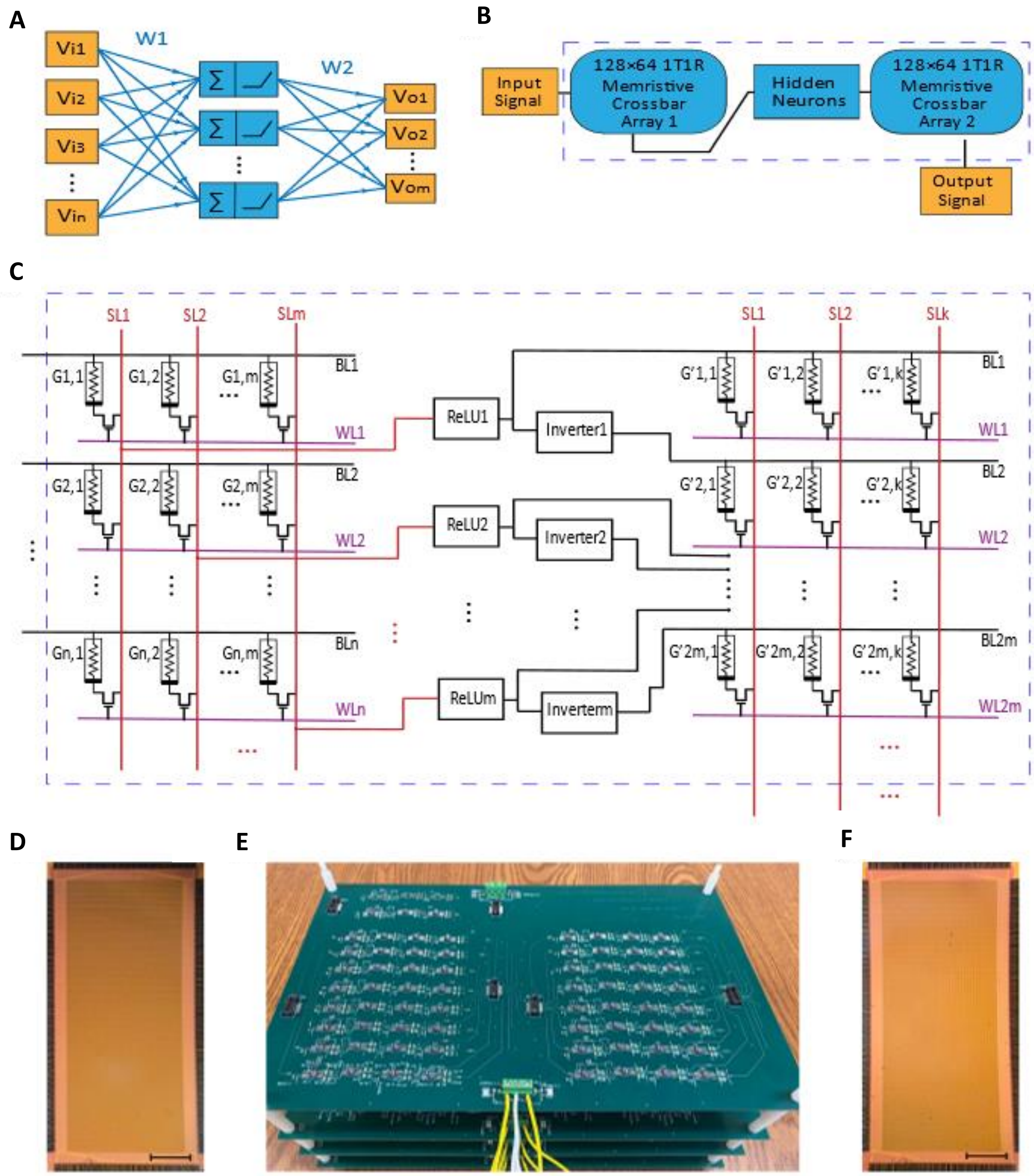


Figure 2.1: Two-Layer Memristor-Based Perceptron with Hardware Neurons and Synapses.

(**A**) Schematic representation of the perceptron, where the hidden layer computes the weighted sum of inputs (V_i) and applies activation functions to the result. (**B**) Block diagram illustrating the implemented perceptron. The weights (W1 and W2 as represented by arrows in **A**) are determined by the conductance of memristor devices in crossbar arrays.

Activation functions are applied using hidden neurons. (**C**) Circuit schematic of the perceptron, featuring 1T1R cells in crossbar arrays connected in series between the selection line (SL) and bit line (BL), with the transistor gate linked to the word line (WL). Hidden neurons (ReLUs) and inverters provide positive and negative inputs for the second layer. (**D**) Optical micrograph of the 1T1R array, serving as the left crossbar in **C** for the first layer. Scale bar: 1 mm. (**E**) A stack of four PCBs housing hardware neurons, each with 16 channels of ReLU activation functions and 16 inverters supporting 16 differential pairs for the second layer. (**F**) Optical micrograph of the 1T1R array, used as the right crossbar in **C** for the second layer. Scale bar: 1 mm.

Each ReLU channel comprises a half-wave current rectifier, a voltage follower, and an inverting amplifier, all constructed using operational amplifiers, as depicted in Figure 2.2A. The current rectifier generates a rectified output voltage directly from the input current, eliminating the need for an additional trans-impedance amplifier for current-to-voltage conversion, resulting in a more compact design. The voltage follower, serving as a unity gain buffer, isolates the first stage from the subsequent one and ensures stability for this stage, and its very high input resistance and very low output resistance prevent undesired impacts from the next stage that might interfere with normal operation [58].

The inverting amplifier is responsible for providing a positive output voltage required for a ReLU activation function. Additionally, it scales down the output voltage to the range of 0 to 0.2 V to avoid altering the conductance of the memristors in the second layer during inference, as illustrated in Figure 2.2B. Figure 2.2C presents a representative output characteristic curve of the hardware ReLU. As anticipated, the output voltage is zero for negative input currents and linearly proportional to the current for positive inputs. We conducted 25 measurements of the output voltage for each input current, with the variation being less than 0.072% in each case on the positive side. The experimental data was processed, and the coefficient of determination (R-squared) [59-61] was calculated to be 1.0, indicating excellent linearity on the positive side.

The relationship between the input current ($\mathrm{I_{in}}$) and the output voltage ($\mathrm{V_{out}}$) of the ReLU is defined by:

$$V_{out} = \begin{cases} R_1 \times R_3/R_2 \times I_{in} & I_{in} \geq 0 \\ 0 & I_{in} < 0 \end{cases} \tag{2.1}$$

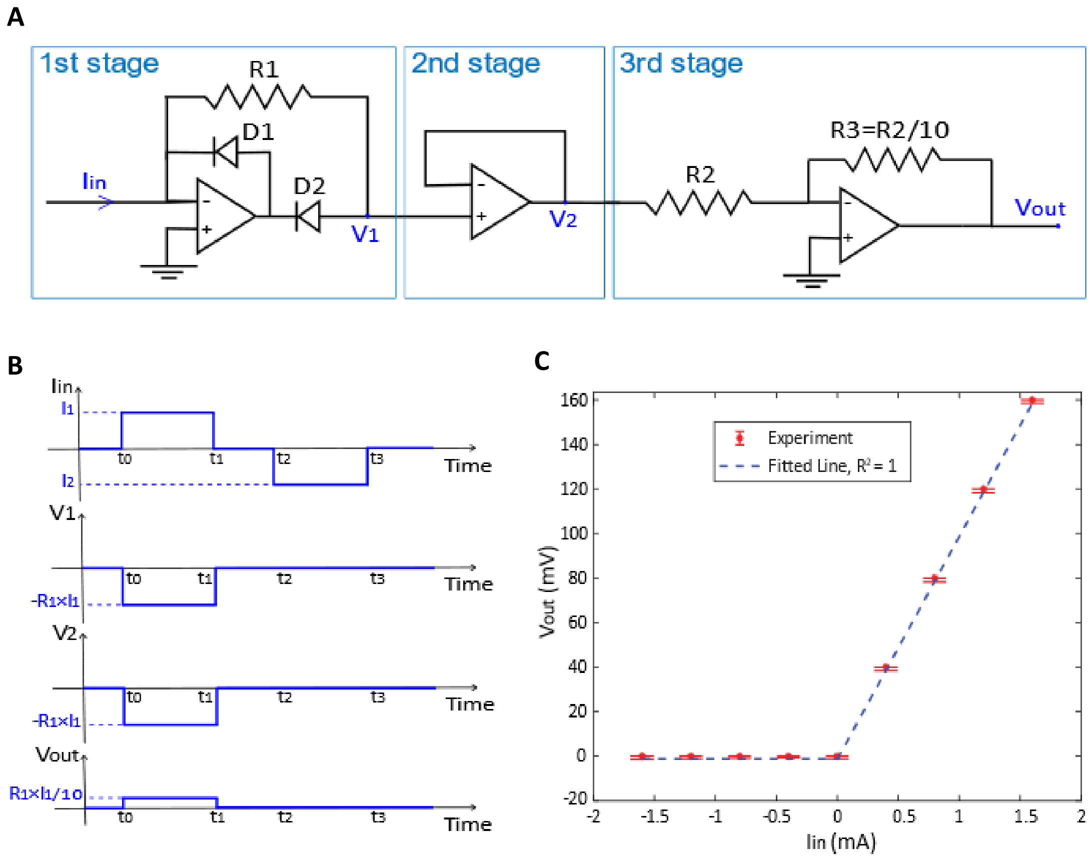
where $\mathrm{R_1}$ represents the feedback resistor of the current rectifier and $\mathrm{R_3/R_2}$ denotes the scaling factor of the inverting amplifier. The product $\mathrm{R_1 \times R_3/R_2}$ signifies the gain of the ReLU.

The DC response illustrated in Figure 2.2C is achieved when the gain of ReLU is set to 100, with a scaling factor of 0.1. The selection of gain is influenced by the range of current flowing through the columns of the crossbar. Opting for a lower gain expands the range of detectable current, albeit at the cost of reduced sensing resolution, as depicted in Figure 2.10.

The measured delay time of the ReLU is 35 ns, as shown in Figure 2.11 (refer to Section 2.8 for detailed information). It is anticipated that this delay time could be further reduced with the adoption of more advanced manufacturing technology. In the current PCB implementation, the bottleneck is observed in the data communication between various modules in the perceptron and between the perceptron and off-chip peripheral circuits. This bottleneck could be significantly mitigated by integrating all the components on a single chip.

Despite these considerations, our hardware ReLU exhibits stability for both positive and negative values of input current, distinguishing it from previous approaches [33, 34]. Moreover, the integration of current-to-voltage conversion and rectification in a single stage contributes to a reduced number of components in the ReLUs, potentially leading to a decrease in circuit area and power consumption compared to prior art.

Figure 2.2: Hardware ReLU Design and Performance.



(**A**) Circuit design of the ReLU. Each unit/channel comprises three stages: a current rectifier, a voltage follower, and an inverting amplifier (labeled in the blue boxes). The first stage rectifies the input current, with the resistor (R1) on the feedback used for gain tuning, and diodes (D1, D2) for rectification. The voltage follower prevents loading by the next stage, and the third stage provides a positive voltage output, consistent with ReLU activation function, while also scaling down the output by adjusting resistors R2 and R3. (**B**) Schematic illustrating signal propagation from input to output. The input current (Iin) undergoes rectification in the first stage, is converted to voltage with a gain of -R1, and the output is duplicated and scaled in the last stage (Vout) by multiplying with -R3/R2 (-0.1 in this design). (**C**) DC characteristic of the hardware ReLU, demonstrating positive and linearly proportional output voltage for positive inputs, and near zero for negative inputs. The error bars, indicating minimal deviation from expected output values, were obtained from 25 measurements for each input. Linear regression shows excellent linearity on the positive side with an R-squared value of 1.

2.2 Training and Inference Methodology

For the training and testing of the two-layer perceptron, we employed electrical pulses with varying amplitudes to represent the intensities of different pixels in an image. These pulses served as inputs to the first layer along the rows of the first 1T1R array. The training process was conducted in-situ and partially on software, utilizing in-house peripheral circuitry.

During the training phase, software ReLU and SoftMax [62] activation functions were applied in the hidden and output layers, respectively. Cross-entropy SoftMax was employed as the loss function for backpropagating errors from the output to the input [63, 64]. This choice facilitates faster convergence and lower error rates compared to the mean square error (MSE) function, particularly in classification applications [62, 65, 66].

To determine the necessary weight updates, we utilized the root mean square propagation (RMSProp) optimizer due to its faster convergence compared to stochastic gradient descent (SGD) [52, 62, 67-73]. The updating of memristor device weights was achieved through a one-shot blind update method [74]. This method involved one reset and one set pulse on the memristor electrode, synchronized with another pulse on the gate of the transistor.

In contrast, the inference process was exclusively executed in hardware. The input pulses were directed into the first memristor crossbar array. The computation results passed through the hardware ReLUs and then proceeded to the second crossbar array. The output current of the second layer was sensed and digitized to facilitate the classification of MNIST digits.

2.3 Classification of MNIST Digits

To showcase its functionality, we employed the fully hardware-based two-layer perceptron, as depicted in Figure 2.12, to classify handwritten digits from the MNIST dataset, spanning zero through nine. The perceptron comprises 128 input neurons, 64 hidden neurons, and 10 output neurons. The first 128×64 1T1R array serves as the first layer, while a 128 by 10 segment of the second 1T1R array functions as the second layer. During computation, two adjacent cells on different rows in the 1T1R arrays were utilized to represent one synaptic weigh.

To accommodate the network dimensions, we cropped and downsampled the images in the MNIST dataset to 8 by 8 pixels. All 60,000 training images in the MNIST dataset were utilized for the training process, while the 10,000 test images were employed for inference.

The training accuracy is influenced by various factors, including device non-ideality in the 1T1R crossbar arrays (refer to Section 2.9 for details), the number of hidden neurons, and the number of weight updates. To comprehensively understand the impact of these factors, we conducted both experimental and simulation training, utilizing a batch size of 50 over 2 training epochs with a total of 2400 weight updates.

In the simulation, we employed both a pure software model, assuming ideal devices, and a practical model that considered noise levels and conductance dynamic range. During experimental training, partial software ReLUs were adopted. The results indicated that an increase in the number of hidden neurons led to higher training accuracy, as illustrated in Figure 2.3A. For instance, by elevating the number of hidden neurons from 16 to 64, the training accuracy improved from 92.38% to 95.77% in the experimental case. This improvement is attributed to the increased complexity and enhanced capability of the perceptron, compensating better for the non-idealities of devices in the network.

Performing more weight updates also had a noticeable effect on the training accuracy, particularly in the early stages, as depicted in Figure 2.3B. However, as the weight updating progressed, the increase in training accuracy slowed down, and the difference between experimental and simulation training results narrowed. In essence, with more weight updating, the experimental accuracy reached nearly identical levels to that of pure software simulation, with less than a 1% difference.

It is noteworthy that the achieved high training accuracy (95.77%), despite device-to-device variations and existing imperfections in the arrays (refer to Section 2.9 and Figure 2.8), can be attributed to the in-situ training scheme employed in this study.

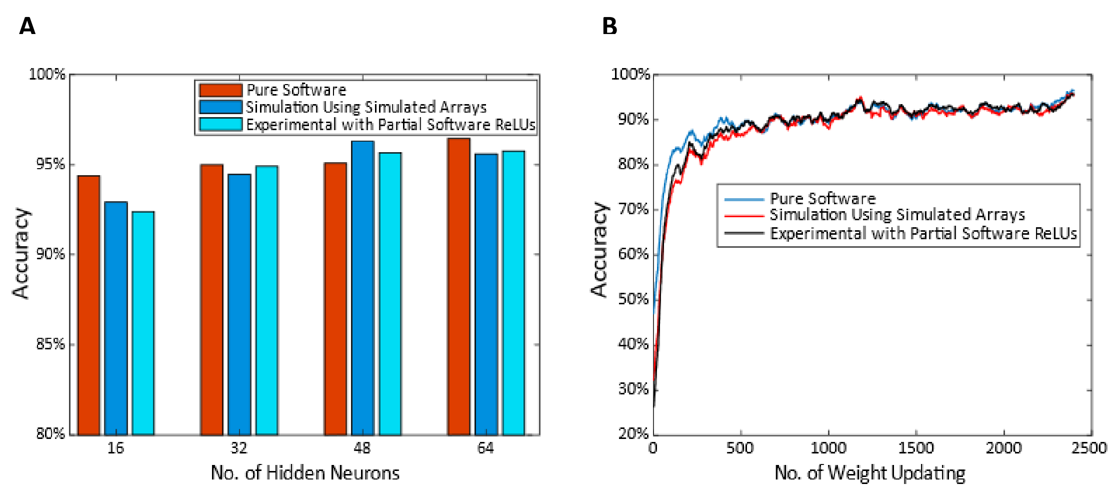


Figure 2.3: In-Situ Training of the Two-Layer Networks.

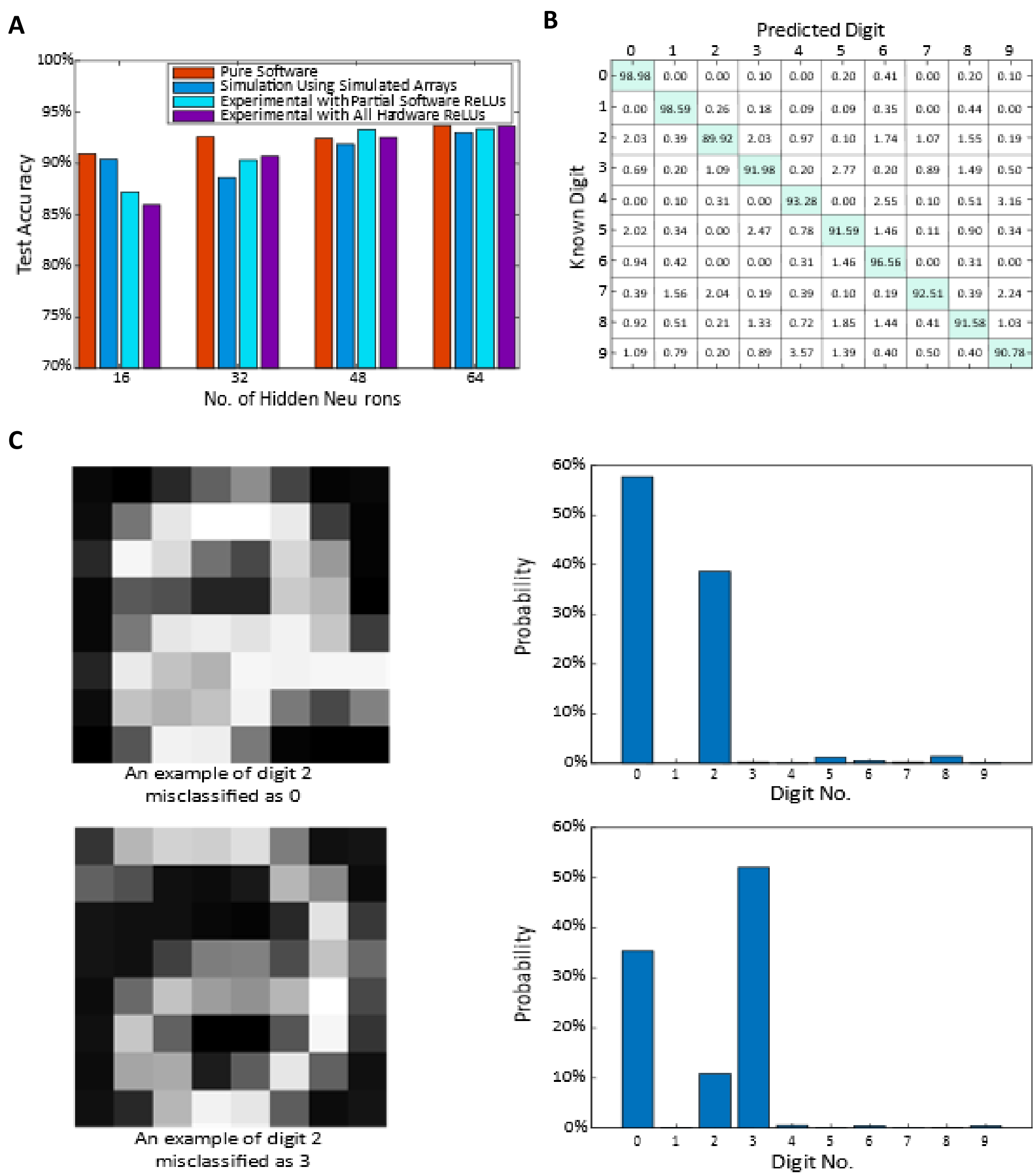
(**A**) Increase in training accuracy with the number of hidden neurons. The network is trained on the MNIST dataset with a batch size of 50 and a total of 2400 weight updates. A comparison is made among three cases: pure software training, simulation using simulated arrays (hardware parameters extracted and used for modeling arrays), and experimental implementation with partial software ReLUs. (**B**) Rise in training accuracy with the number of weights updated, accompanied by a narrowing difference in training accuracy among the three cases. During training, weights in each array were updated, and network accuracy was measured after each batch (50 images).

Likewise, the inference accuracy exhibits significant improvement with a larger number of hidden neurons. To assess performance, we utilized 10,000 test images on both our fully hardware perceptron and a network with hidden neurons partially implemented in software. The detailed discussion and presentation of our fully hardware perceptron is available in Section 2.11 and Figure 2.13.

The inference accuracy increased from 85.94% to 93.63% for our fully hardware perceptron as the number of hidden neurons was raised from 16 to 64, as depicted in Figure 2.4A. Concurrently, the disparity between the inference accuracy of the perceptron with hardware and partial software neurons decreased from 1.27% to nearly zero. Examining the confusion matrix, as illustrated in Figure 2.4B, revealed that the most frequent misclassification occurred with the digit 2, primarily misclassified as digits 0 or 3, as shown in Figure 2.4C. It is noteworthy that digits with higher classification errors are often challenging for human eyes to differentiate as well, possibly a consequence of the image downsampling done to accommodate the network dimension.

Figure 2.4: Inference Results of the Two-Layer Hardware Perceptron.

(**A**) Increase in inference accuracy with the number of hidden neurons. Accuracy is measured in different cases: pure software, simulated arrays in simulation, experimental



implementation with partial software hidden neurons, and experimental implementation with all-hardware hidden neurons. Noticeable improvement in test accuracy is observed in the case of experimental implementation with all-hardware hidden neurons. (**B**) Confusion matrix depicting the inference accuracy using all-hardware hidden neurons. Digit 2 is most frequently misclassified. (**C**) Two images of digit 2 (left column) applied as input, and based on the output current, the corresponding probability of each digit (right column) is calculated using a SoftMax function. The digit with the highest probability represents the classification result. Both images are misclassified, the first one as digit 0 and the second one as digit 3, which are challenging even for human eyes to recognize.

2.4 Examination of Nonidealities

To assess the impact of device non-idealities and built ReLUs on the perceptron's functionality, we incorporated non-idealities into our models and evaluated their effects on classifying MNIST digits through simulation. The linearity of conductance tuning during gate voltage increments and decrements, as illustrated in Figure 2.8B, is a crucial factor in perceptron performance. Significant non-linearity can markedly degrade learning accuracy.

To model the non-linearity of devices in our perceptron, we adopted an exponential relationship between the conductance of the devices and the gate voltage [75], as depicted in Figure 2.5A. These non-linear models were then employed during the programming phase of training in our perceptron. The results indicate that small non-linearities have a limited impact on learning accuracy, causing less than a 3% decrease. However, substantial non-linearities can significantly degrade accuracy, as shown in Figure 2.5B.

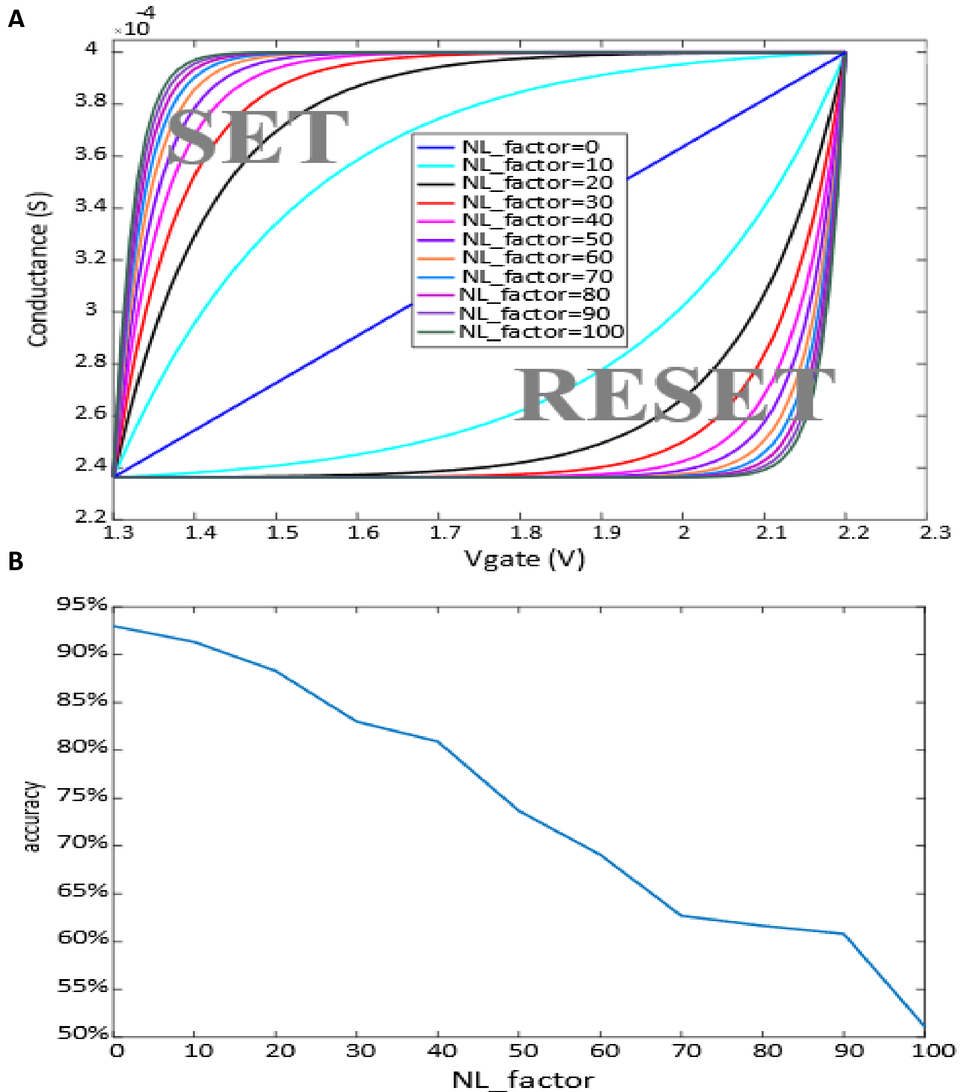


Figure 2.5: Effect of Non-Linearities of Memristors.

(**A**) Non-linearity models of memristors. The conductance of memristors during set and reset is modeled using exponential functions, where the conductance is tuned with respect to the exponent of the gate voltage instead of a linear tuning relationship. (**B**) Recognition accuracy with non-linear models. The two-layer perceptron was trained in simulation using various non-linearities for memristors, and the recognition accuracy after training was observed. Significant non-linearities can markedly degrade accuracy.

Another non-ideality that can impact the perceptron's performance is the noise associated with our built ReLUs, originating from the thermal noise of resistors as well as voltage and current noise of amplifiers, as illustrated in Figure 2.6A. The total noise at the output of each channel of the ReLU is calculated to be 61.05 µV (refer to Section 2.12). To assess the effect of noise, we conducted an experiment in simulation using our practically modeled perceptron by introducing different levels of noise to the output of ReLUs, observing the learning accuracy during inference.

Figure 2.6B indicates that with the addition of 61.05 µV of noise at the output, accuracy remains constant. Furthermore, the results demonstrate that our perceptron exhibits high resilience to noise, as even with the introduction of substantial noise (20 mV, equivalent to 10% of the output voltage), the accuracy drops by less than 5%.

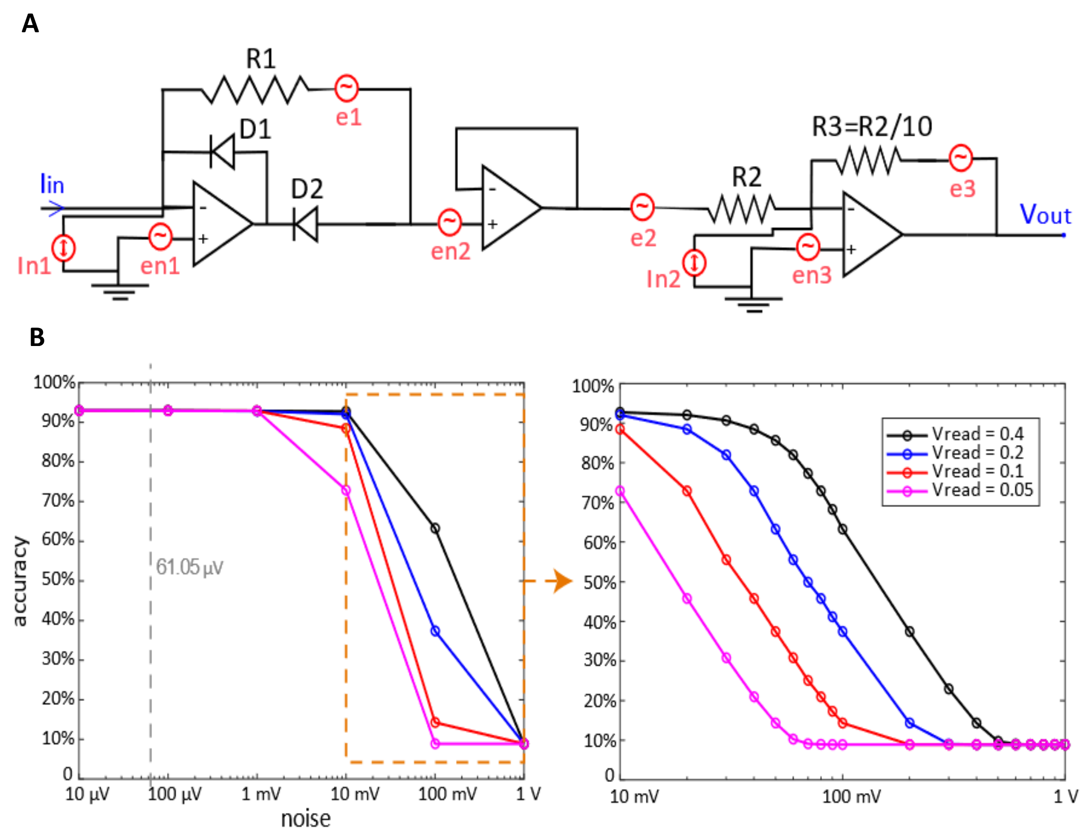


Figure 2.6: Noise of the ReLU and Its Effect on Perceptron Recognition Accuracy.

(**A**) Noise sources in the constructed ReLU. Resistors are associated with thermal noises modeled as voltage sources (e1-e3). Amplifiers are associated with voltage and current noises due to their internal circuits, modeled as (en1-en3) and (In1-In2). These noises combine at the output of the ReLU. (**B**) Effect of ReLU noise on perceptron recognition accuracy. Various total noises were added at the output of the ReLU in the perceptron in simulation, and the recognition accuracy was observed. The total noise of the built ReLU, equal to 61.05 µV, has no impact on accuracy. The built perceptron shows high resilience to noise at the output of the ReLU, with less than 5% accuracy degradation when adding noise of 10% of the output voltage for read voltages of 0.1V, 0.2V, or 0.4V, and around 15% accuracy degradation when the read voltage is 0.05V.

Two additional factors that can impact the perceptron's performance are mismatches and errors in ReLU gains. The gain of the ReLU is primarily determined by the feedback resistor R1, as illustrated in Figure 2.2A. In an integrated design, it is feasible to fabricate resistors with errors ranging from 10% to 30% compared to the nominal value [76], and less than 1.5% mismatch with an optimal layout configuration [76-78]. To evaluate the impact of errors, we introduced errors of up to 30% to the gain of ReLUs and examined the perceptron's performance, which exhibited no accuracy degradation compared to the ideal case with zero error.

Additionally, we conducted Monte Carlo simulation [79], where resistor mismatch was modeled by introducing a normal distribution to the gain of ReLUs, and the effect of this mismatch was observed in the system results. The standard deviation of the mismatch was considered up to 3%, and the learning accuracy was measured 20 times for each standard deviation. The mean accuracy value changed by 0.1%, with a maximum variation of 0.7%. This indicates that our perceptron can tolerate a 3% mismatch in resistors and up to 30% error in resistors without experiencing significant degradation in performance.

2.5 Examination of Power, Area, and Latency

To evaluate efficiency in terms of power, area, and throughput, we compare a perceptron with analog hidden neurons to an equivalent perceptron with digital-based hidden neurons that encode inputs using the amplitude of input pulses. The data flow for a perceptron with hidden neurons implemented using digital units involves analog-to-digital conversions and data communication between the hardware and those digital units for each layer (refer to Figure 2.7 and Table 2.1). The intensive use of power-hungry peripheral circuits such as ADCs/DACs and digital units contributes to increased power consumption and latency.

On the other hand, when using analog-based hardware neurons, the analog signal is directly sent to the next layer of the perceptron through the ReLU boards. Consequently, there is no need for peripheral analog circuits and digital-based data communication between the layers, leading to a substantial improvement in power efficiency and computing throughput. It is worth mentioning that in our measurements using our implemented perceptron, the latency of the hidden layer using analog-based hardware ReLUs is 35 ns, as mentioned previously, compared to the much higher latency that could be resulted from the hidden layer using digital-based ReLUs.

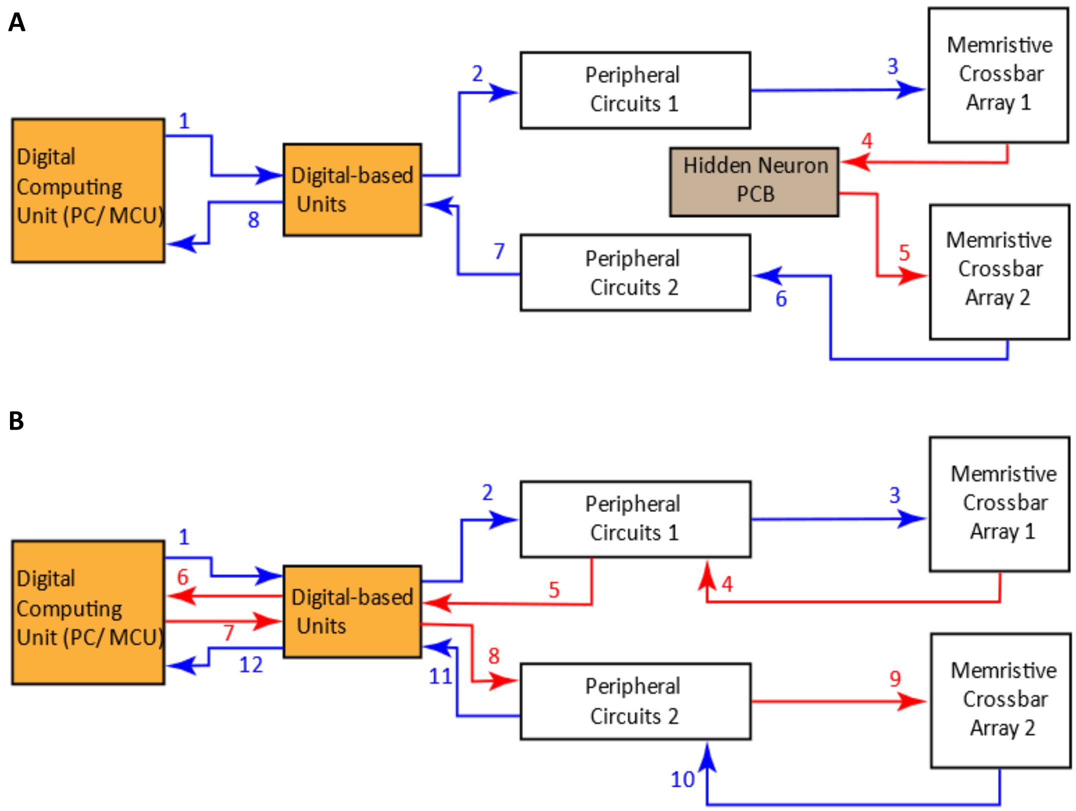


Figure 2.7: Inference Schemes for Neural Networks with Hardware and Digital-Based Neurons.

(**A**) Block diagram of the case using all-hardware hidden neurons. Two memristor crossbar arrays on two chips are employed, each controlled by a separate set of MCUs and peripheral circuits. Arrows depict the forward path, with numbers indicating steps during inference. Hidden neurons (red arrows) are fully implemented in hardware using the designed hidden neuron PCB with multiple analog ReLUs. (**B**) Block diagram of the case using digital-based hidden neurons. All steps are identical to the all-hardware neuron cases, except hidden neurons (red arrows) are implemented on a digital computing unit. The use of hardware neurons reduces unnecessary data communication, resulting in fewer power-hungry circuit components, enhancing power and area efficiencies, and achieving higher throughput, as explained in Section 2.13.

Table 2.1: Components used in an all-hardware hidden neuron and a digital-based hidden neuron.

All-hardware neuron	**Digital-based neuron [18]**

trans-impedance amplifier	trans-impedance amplifier
buffer	buffer
inverting amplifier	sample and hold
-	ADC
-	DAC
-	Digital-based units
-	Digital computational unit (MCU or PC)

To achieve optimal efficiency in terms of power and area, we conducted estimates for power consumption and area occupation of the designed analog-based hidden layer and its digital-based counterpart in the 65 nm CMOS technology node, considering state-of-the-art DACs, ADCs, and amplifiers. The estimations reveal that utilizing all hardware hidden neurons will enhance power and area efficiencies by at least 32.8 times and 5.64 times, respectively. Detailed information on these estimations is provided in Section 2.13. It is important to note that previous efforts [80] aimed at reducing the power consumption of the hidden layer by proposing a new ADC. However, their proposed ADC is only applicable to a perceptron with inputs encoded using pulse-width, and its resolution is subject to a tradeoff with energy consumption. Additionally, the memristor devices were programmed either binary or 2-bit, and the hidden layer required a digital controller, introducing extra latency.

2.6 Device Fabrication, Chip Integration, and their Electrical Performance

The 1T1R cells and interconnections between cells were fabricated in a commercial foundry with 2 µm technology. The integration of memristor devices onto the CMOS substrates was carried out at UMass [17]. The exposed metal vias on the CMOS substrates were cleaned using argon plasma to eliminate the native oxide. Subsequently, three different metal layers, approximately 7 nm Ag, 3 nm Ti, and 50 nm Pd, were deposited sequentially through DC sputtering under an 8.8×10^{-7} torr background vacuum. The layers were then lifted off in acetone with a sonication process lasting 10 seconds at room temperature, followed by annealing at 300 °C for 40 min in an N_2 atmosphere. The bottom electrodes (20 nm Pt with a 2~2.5 nm Ti adhesion layer) were patterned and evaporated onto the metal pads, followed by lift-off in acetone. A 5 nm Ta_2O_5 blank layer was sputtered using 90 W RF power and 20 sccm Ar flow (background vacuum in the sputterer was better than 3.3×10^{-7} torr). The top electrodes (20 nm Ta and 20 nm Pt) were patterned using photolithography, deposited through DC sputtering, and lifted off in acetone at room temperature.

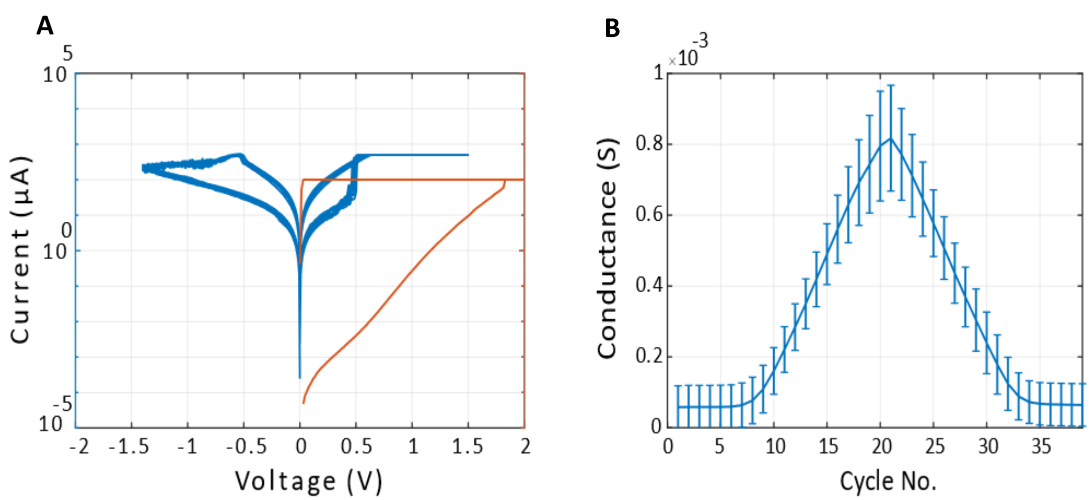


Figure 2.8: Electrical Performance of an Individual Cell and an Array.

(**A**) DC voltage forming (orange) process and 10 DC voltage sweeping loops (blue) of one device in an array. The initial resistive state of the discrete memristor is high. To form a filament inside the switching layer, a DC voltage forming process was performed by sweeping a positive voltage across the memristor in the order of 0 → 2 V → 0 with a current compliance of 100 μA applied by Keysight B1500A. For the DC voltage sweeping loops, reset and set processes were performed 10 times. The reset process was done by negative voltage sweeping in the order of 0 → -1.5 V→ 0 without current compliance, while a set process was done by a positive voltage sweeping in the order of 0 → 1.5 V → 0 with a current compliance of 500 μA. (**B**) The average conductance of the devices on the 128×64 array during conductance tuning. To tune the conductance of the devices on the array, their corresponding gate voltage was gradually increased and then decreased for a total of 39 cycles. Conductance was read after each cycle.

2.7 Layout Design of the Hardware ReLU

The four hardware ReLU PCBs consist of 64 channels of ReLUs along with TIAs in parallel that convert positive input currents to positive output linearly with a specific gain and convert negative input currents to zero. Each PCB itself includes 16 channels as shown in Figure 2.9. The PCB itself is structured into four layers, with the first and fourth layers serving as the signal layer, the second layer as the GND layer, and the third layer as a power layer for routing. The layout of the hardware ReLU is depicted in Figure 2.9.

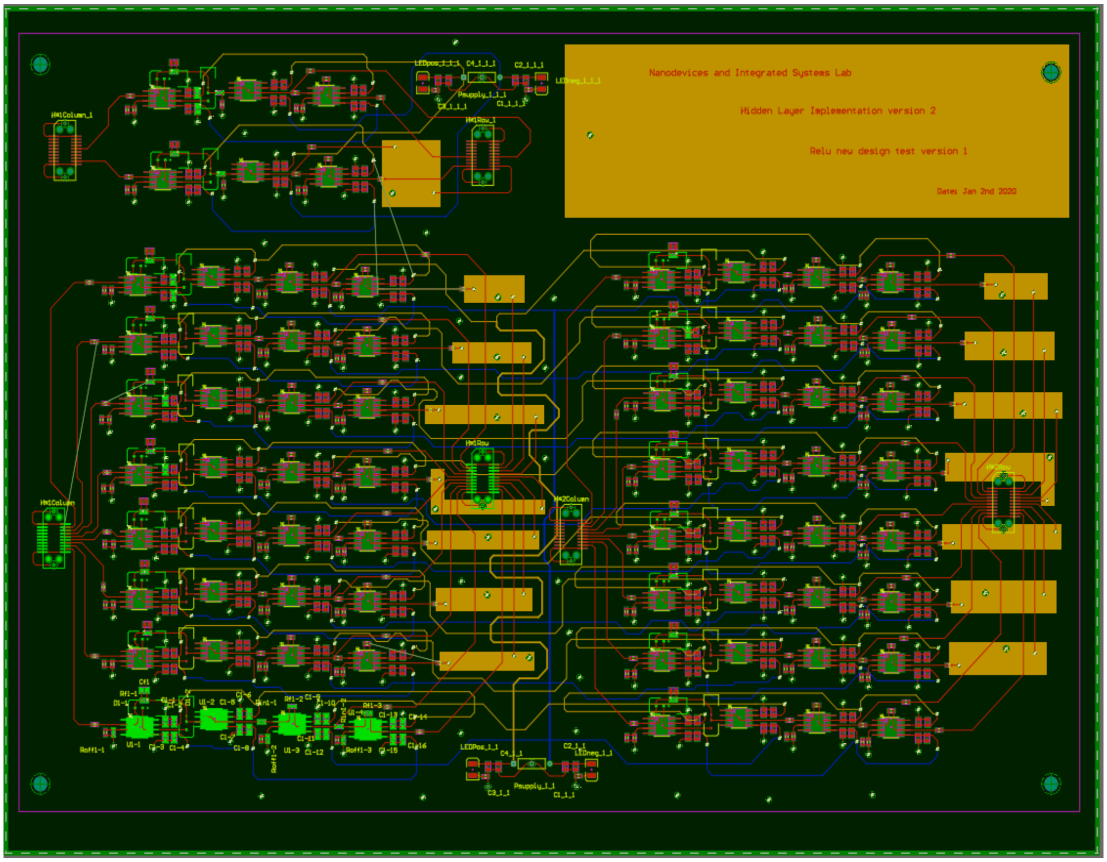


Figure 2.9: PCB of The Hardware ReLU Consisting of 16 Channels.

Red and blue traces are routed on the first and fourth layers. The green plane represents the GND plane and is located as the second layer. The yellow traces and polygons are on the third layer.

2.8 DC and AC Response of the Hardware ReLU and Delay Measurement

The DC characteristics of the ReLU are influenced by its gain, as depicted in Figure 2.10. Elevating the gain of the ReLU enhances sensing resolution; however, it results in a reduced range of current that can be sensed.

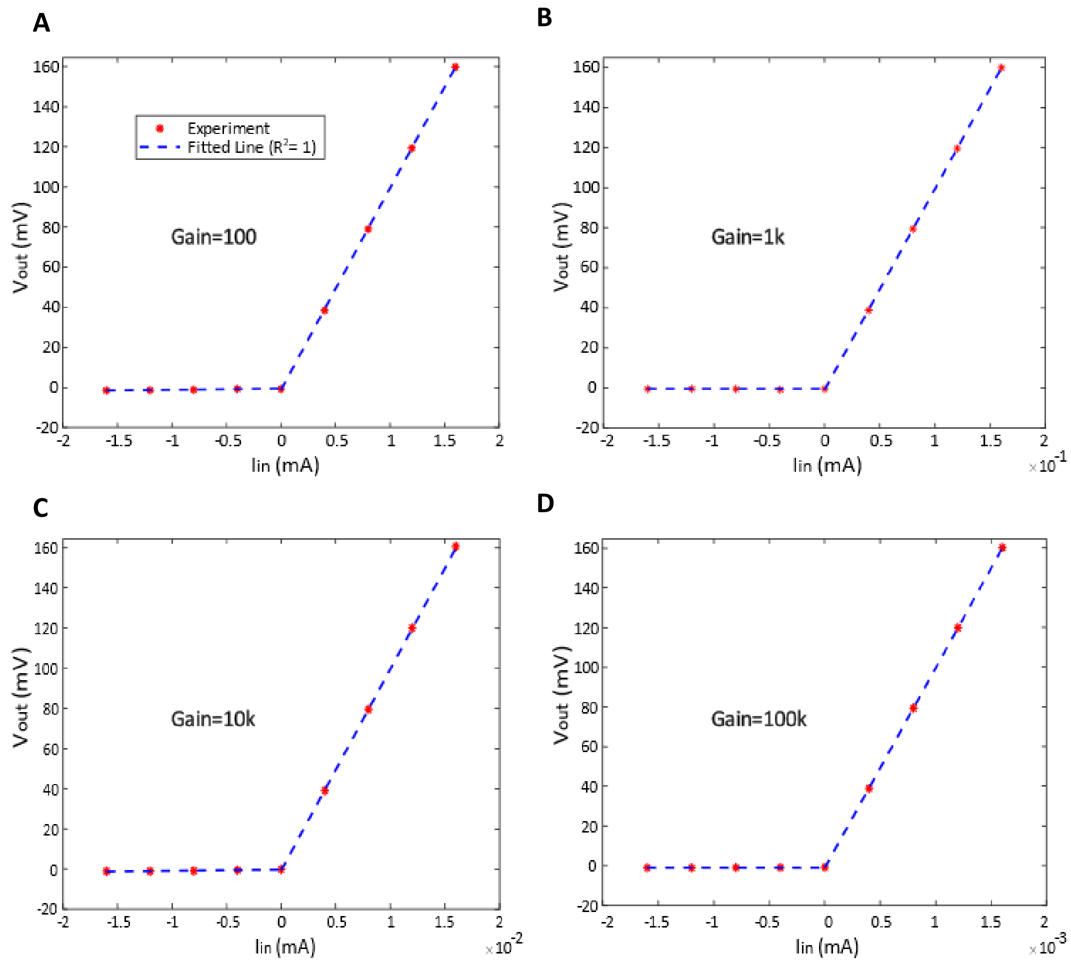


Figure 2.10: Effect of the ReLU Gain on its DC Characteristics.

The gain was tuned to be (**A**) 100, (**B**) 1k, (**C**) 10k, and (**D**) 100k by changing the feedback resistor (R1) in the first stage of the ReLU. In all cases, the output voltage is zero when the input current is negative, while linearly proportional to the input current on the positive side ($R^2 = 1$). A higher gain of the ReLU leads to a higher resolution in input current sensing but a smaller range of the current that can be sensed.

To assess the AC characteristics of the ReLU, we connected a 1 kΩ resistor between a voltage source and the negative input of the amplifier within the initial ReLU stage. This arrangement provided input current to the ReLU. The combination of this resistor with parasitic and amplifier input capacitors formed an RC circuit, acting as a low-pass filter.

This configuration introduced distortion and delay, thereby constraining the frequency response of the ReLU, as illustrated in Figure 2.11. The integrated design approach, where all components are integrated into a single chip, offers a potential solution to mitigate or eliminate the observed delay and distortion. This stands in contrast to the PCB version employing discrete components, as demonstrated in this thesis.

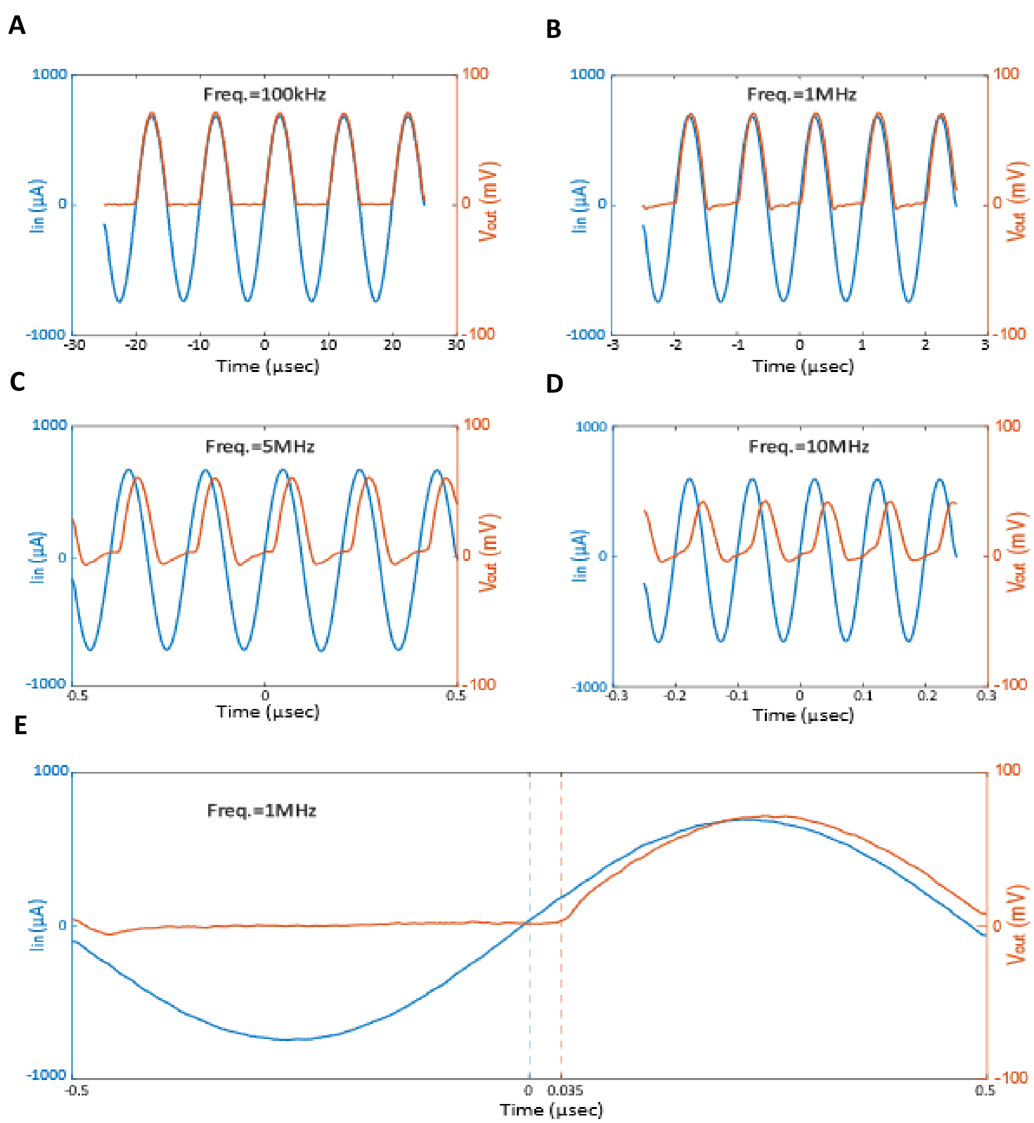


Figure 2.11: AC Characteristics of the Hardware ReLU.

The gain was set to be 100, and the AC characteristic of the ReLU was measured at (**A**) 100KHz, (**B**) 1MHz, (**C**) 5MHz, and (**D**) 10MHz, respectively. The output voltage follows the input current better at lower frequencies. As the frequency increases, the output can still follow the input; however, the amplitude of the output decreases. (**E**) zoom-in plot for **B**, showing a noticeable delay. The delay becomes more visible at higher frequencies due to the finer resolution of the oscilloscope. The delay and amplitude distortion result from a low pass filter in the input circuitry. The delay does not prohibit the network from operating at a reasonable frequency (e.g., 1 MHz) and would be much reduced by building the ReLUs on an integrated chip.

2.9 Non-idealities and Programming Yield of the Arrays

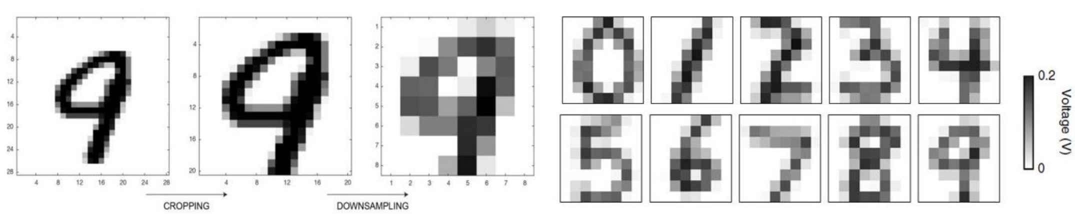
After forming and programming, we evaluated the device conductance, categorizing a device as stuck-on if its conductance exceeded an upper threshold (1 mSiemens) or stuck-off if the conductance fell below a lower limit (10 µSiemens). Both stuck-on and stuck-off devices were considered defects, whereas devices with conductance levels in between were classified as normal. The yield of the arrays was quantified as the ratio between the number of normal devices and the total number of devices in the array (8192 for a 128×64 array). In our experiments, the first array demonstrated a yield of 99.85%, while the yield for the second array was 88.97%.

As depicted in Figure 2.1, we successfully deployed a two-layer fully hardware perceptron featuring 64 hidden neurons, utilizing both arrays. The entire devices in the first array (128×64 devices) were employed for the first layer, while only the 128×10 segment of the second array served for the second layer. Given the non-uniform distribution of defective devices in the second array, we strategically selected a portion with a lower number of defects (yielding 99.61%) to mitigate their potential adverse impact.

2.10 MNIST Digits Downsampling and their Mapping to the Perceptron Input

The initial MNIST digits, originally sized at 28×28, undergo a preprocessing step involving image cropping to eliminate pixels without any useful information. This results in a reduction of image size to 20×20. To further optimize the images for our system, a downsampling process is employed, reducing them to 8×8 using interpolation, as illustrated in Figure 2.12A. Simultaneously, the intensity of the images is scaled down to a range of 0 to 0.2V, ensuring compatibility with the proper functioning of memristors. Figure 2.12B visually represents how the input image pixels are mapped to our perceptron for the classification task. It is noteworthy that our system utilizes both the original intensity value and the inverted intensity value of the pixels as inputs to implement differential pairs, enabling the incorporation of both positive and negative synaptic weights.

A



B

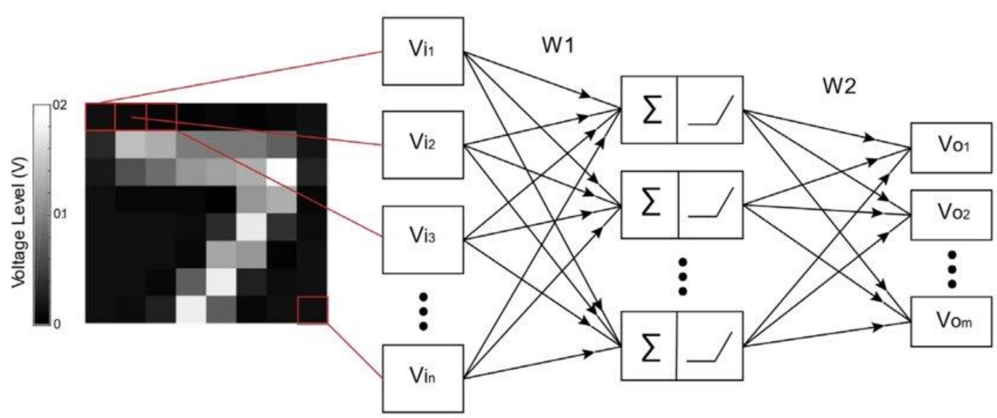
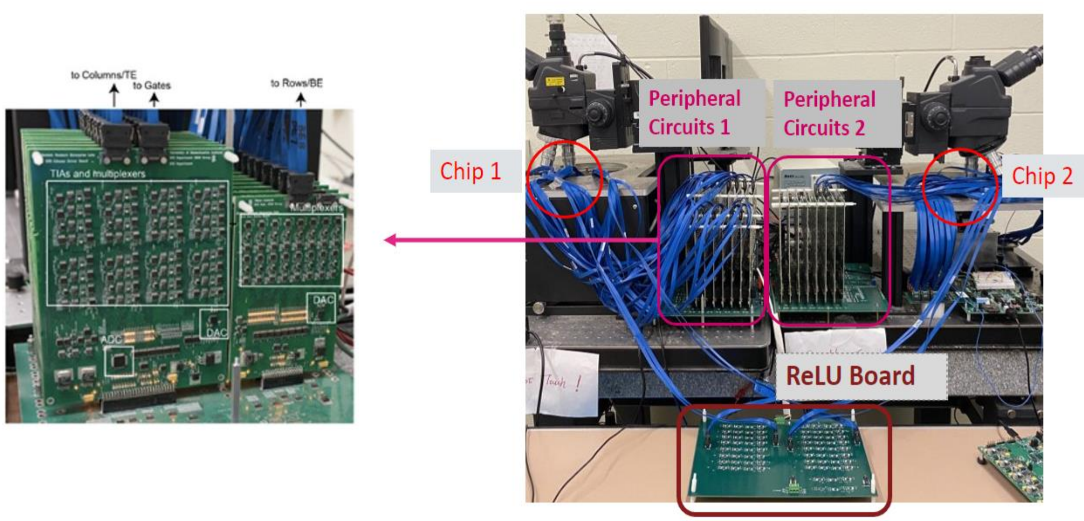


Figure 2.12: MNIST Digits Conditioning for the Perceptron.

(**A**) MNIST digits are cropped and downsampled using interpolation from 28×28 to 8×8. Additionally, the intensity of the pixels is scaled down to the range of 0 to 0.2V. (**B**) Each pixel is mapped to one input of the perceptron for the classification task.

2.11 Perceptron with Fully Analog ReLUs

The perceptron architecture comprises three layers, with the input layer having a size of 128×64 and the output layer having a size of 128×10. The intermediate layer is a non-linear layer, incorporating ReLU channels to introduce the necessary non-linearity for classification tasks. To implement the input and output layers, two sets of peripheral circuits and memristor arrays, fabricated on integrated chips, are utilized to perform VMMs.



A ReLU board is strategically positioned between these layers, tasked with sensing current flows through the first memristor array generated by the initial VMM. This board applies a non-linear activation function, along with current-to-voltage conversions. The resulting voltage outputs from the ReLU board are then directed to the second set of peripheral circuits for the subsequent VMM operation.

Figure 2.13: Perceptron with Fully Analog ReLUs Implemented Experimentally.

The memristor arrays are fabricated on integrated Chip 1 and Chip 2. Peripheral circuits are responsible for applying voltage inputs to the memristor arrays for VMMs and reading the current flowing through the memristors as the results of VMMs. The ReLU Board is placed between those two VMMs to apply nonlinear activation functions as required for the implementation of classification tasks and current-to-voltage conversions.

2.12 Calculation of ReLU Output Referred Noise

The overall noise at the output of the ReLU is influenced by multiple sources as depicted in Figure 2.6A. These include the thermal noise of resistors, characterized by a voltage spectral density in the range of e_1^2 - e_3^2, the voltage noise of amplifiers with a spectral density of e_{n1}^2 - e_{n3}^2, and the current noise of amplifiers with a spectral density of I_{n1}^2 - I_{n2}^2.

In general, the voltage spectral density of thermal noise in a resistor is expressed as 4KTR, where K represents the Boltzmann constant, T is the temperature, and R denotes the resistor value. The voltage and current noise of an amplifier are determined by the internal circuit of the amplifier and are extracted from the amplifier datasheet [81] as $(4.3)^2$ $(nV/\sqrt{Hz})^2$ and $(0.1)^2$ $(pA/\sqrt{Hz})^2$, respectively.

The total noise at the output is computed as the sum of the products of the noise sources multiplied by their respective gains to the output. At 1 MHz bandwidth, it is calculated using

$$E_{e1\_out}{}^2 = \left(e_1 \times {}^{R_3}/_{R_2}\right)^2 = \left({}^{R_3}/_{R_2}\right)^2 \times \int_0^{1MHz} 4KTR_1 df = 16.56 \times (10)^4 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.2)$$

$$E_{e2\_out}{}^2 = \left(e_2 \times {}^{R_3}/_{R_2}\right)^2 = \left({}^{R_3}/_{R_2}\right)^2 \times \int_0^{1MHz} 4KTR_2 df = 16.56 \times (10)^6 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.3)$$

$$E_{e3\_out}{}^2 = (e_3)^2 = \int_0^{1MHz} 4KTR_3 df = 16.56 \times (10)^7 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.4)$$

,

$$E_{en1\_out}{}^2 = \left(e_{n1} \times {}^{R_3}/_{R_2}\right)^2 = \left({}^{R_3}/_{R_2}\right)^2 \times \int_0^{1MHz} (4.3)^2 df = 18.49 \times (10)^4 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.5)$$

$$E_{en2\_out}{}^2 = \left(e_{n2} \times {}^{R_3}/_{R_2}\right)^2 = \left({}^{R_3}/_{R_2}\right)^2 \times \int_0^{1MHz} (4.3)^2 df = 18.49 \times (10)^4 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.6)$$

$$E_{en3\_out}{}^2 = \left(e_{n3} \times (1 + {}^{R_3}/_{R_2})\right)^2 = \left(1 + {}^{R_3}/_{R_2}\right)^2 \times \int_0^{1MHz} (4.3)^2 df$$
$$= 22.3729 \times (10)^6 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.7)$$

$$E_{In1\_out}{}^2 = \left(I_{n1} \times R_1 \times {}^{R_3}/_{R_2}\right)^2 = \left(R_1 \times {}^{R_3}/_{R_2}\right)^2 \times \int_0^{1MHz} (0.1 \times (10)^{-3})^2 df$$
$$= (10)^2 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.8)$$

$$E_{In2\_out}{}^2 = (I_{n1} \times R_3)^2 = (R_3)^2 \times \int_0^{1MHz} (0.1 \times (10)^{-3})^2 df = (10)^6 \quad (\mathbf{nV})^{\mathbf{2}} \qquad (2.9)$$

which leads to the total noise of

$$
\begin{aligned}
&E_{total\_rms} \\
&= \sqrt{E_{e1\_out}{}^{2} + E_{e2\_out}{}^{2} + E_{e3\_out}{}^{2} + E_{en1\_out}{}^{2} + E_{en2\_out}{}^{2} + E_{en3\_out}{}^{2} + E_{In1\_out}{}^{2} + E_{In2\_out}{}^{2}} \\
&= 61.05\ \mu V
\end{aligned}
\tag{2.10}
$$

The calculated noise was incorporated into the output of ReLUs within the perceptron during simulation. The simulation results, as depicted in Figure 2.6B, reveal no noticeable degradation in accuracy despite the presence of this added noise.

2.13 Estimation of the Power and Area Efficiencies

As per Figure 2.7, the employment of hardware neurons in a multilayer perceptron eliminates the necessity for peripheral analog circuits, digital-based units, digital computational units (MCU or PC), and data communication between layers. Conversely, when utilizing digital-based neurons, the inclusion of digital-based units and digital computational units (MCU or PC) becomes essential between adjacent layers. Additionally, multiple peripheral circuits, including a trans-impedance amplifier, a buffer, a sample and hold, an ADC, and a DAC (as outlined in Table 2.1), are required. The integration of analog-based hardware neurons in the multilayer perceptron significantly reduces circuit complexity, resulting in improved power and area efficiencies. Equally noteworthy is the substantial reduction in data communication between digital and analog units, leading to enhanced overall efficiency and throughput.

The power efficiency can be estimated based on the state-of-the-art components available at the 65 nm CMOS technology node. As outlined in Table 2.2, a cutting-edge amplifier [82] consumes only 0.72 mW of power, an ADC [83] consumes 1.4 mW, and a DAC [84] consumes 68 mW. For the sake of a simplified comparison, the digital units, digital computational unit (MCU or PC), and S&H are disregarded in the digital-based design, focusing solely on the remaining components in the subsequent calculations.

By using the analog-based hardware neurons instead of the digital-based ones, power consumption of the ReLUs is projected to improve by at least (2×0.72 mW + 1.4 mW + 68 mW) / (3×0.72 mW) = 32.8 times.

The area of the circuit components at the 65 nm node are detailed in Table 2.2, with a state-of-the-art amplifier occupying 0.014 mm^2, an ADC covering 0.079 mm^2, and a DAC spanning 0.13 mm^2. Thus, it can be inferred that the adoption of analog-based hardware neurons, as opposed to digital-based ones, would lead to an improvement in area occupation by at least (2×0.014 mm^2 + 0.079 mm^2 + 0.13 mm^2) / (3×0.014 mm^2) = 5.64 times.

Table 2.2: Power consumption and area occupation of several state-of-the-art components in 65 nm CMOS technology node.

Component	**Power Consumption (mW)**	**Area Occupation (mm^2)**
ADC [83]	1.4	0.079
DAC [84]	68	0.13
Amplifier [82]	0.72	0.014

Given the inherent trade-off among power, speed, area, and resolution in integrated circuit design, several constraints were taken into account for their minimum speed and resolution. Specifically, a resolution exceeding 8 bits and a bandwidth (BW) surpassing

100 MS/s were considered for the ADC and DAC, a unity gain bandwidth (GBW) exceeding 40 MHz were considered for the amplifier.

It is important to note that in the comparisons of power and area, the consideration was limited to the power and area occupied and consumed by the specified analog components of the digital-based neuron. This excluded the contributions of other analog components like Sample and Hold (S&H), as well as the power and area consumed by digital-based units and digital computational units (MCU or PC), for the sake of simplicity. The inclusion of these components would result in even more significant improvements in area and power efficiencies achieved by the analog-based hardware neurons compared to the digital counterparts.

2.14 Discussion

The fully hardware-based two-layer perceptron inherits the benefits of using memristor devices as synapses for in-memory computing. Additionally, the integration of hardware neurons enhances power and area efficiencies. To fully realize efficiency in terms of power, area, and throughput, there is a need for an integrated design of such an analog-based fully hardware perceptron, building upon the proof-of-concept demonstration presented in this dissertation.

The integrated chip is anticipated to exhibit higher area efficiency and lower power consumption compared to board-level implementations. Furthermore, integrating all components onto a single chip reduces parasitic capacitance and resistance, resulting in substantially lower latency according to research [85]. Eliminating data communication and A/D conversions between layers of the perceptron contributes to a significant reduction in latency and an improvement in computing throughput. This throughput enhancement is particularly noteworthy when considering the computing overhead caused by digital computational units. Therefore, the design and implementation of neurons, crossbar arrays, and peripheral circuits, all on hardware using specially designed analog components, are crucial to fully leverage the substantial benefits of memristor crossbars in perceptron implementation. Therefore, it is crucial to design and implement neurons, crossbar arrays, and the peripheral circuits, all on hardware using specifically designed analog components to enjoy the significant benefits of memristor crossbars to the most in the perceptron implementation.

Our fully analog-based perceptron is well-suited for in-sensor computing [86, 87], which can further reduce power consumption by processing sensory data on-site.

To achieve a high level of recognition accuracy within the perceptron, it is necessary to fabricate memristor crossbar arrays that exhibit both high yield and uniformity. This requires minimizing the presence of defective devices within the array. Additionally, a critical aspect involves incorporating a substantial number of hidden neurons, as an increase in their quantity correlates with enhanced accuracy. Furthermore, achieving exceptionally high recognition accuracy for larger input images necessitates the utilization of larger arrays, removing the need for downsampling the input images to fit the array size.

It is noteworthy to acknowledge the inherent trade-off between the gain of ReLUs and their bandwidth. Furthermore, in the integrated design scenario where amplifiers are being designed with transistors, it is necessary to minimize the mismatch of transistors and the offset of amplifiers. Consequently, there is a need to meticulously design the ReLUs in alignment with the specified requirements.

2.15 Summary

We successfully designed and implemented a 64-channel hardware ReLU activation function using off-the-shelf analog components. This was integrated into a two-layer fully hardware-based perceptron alongside memristor crossbar arrays. The perceptron demonstrated its capabilities by accurately classifying digits from the MNIST dataset, achieving a recognition accuracy of 93.63%. This performance is on par with a perceptron that incorporates neurons partially implemented in software.

The fully hardware perceptron has advantages in power efficiency, area occupation, and computing throughput, which will be further enhanced with an implementation in integrated circuits.

CHAPTER 3

MATRIX-MATRIX POINTWISE MULTIPLICATION WITH THREE-DIMENSIONAL MEMRISTOR ARRAYS: MODELLING AND SIMULATION

Within this chapter, a novel architectural design for a 3D memristor array is presented. Previous accelerator demonstrations utilized memristors for parallel VMM, requiring conditioning of inputs for conversion into 1D vectors suitable for computation which led to the necessity of additional computing resources like digital processing units, increasing energy consumption and causing delays. While researchers have explored 3D arrays to enhance parallel computation, they still require unrolling of input data, adding to resource and time demands. Notably, there has not been an implementation of 3D arrays bypassing input unrolling, enabling direct processing of sensor array data, promising improved parallelism, energy efficiency, and throughput.

The proposed array architecture in this chapter demonstrates the capability to perform matrix-matrix pointwise multiplication in a single operation, directly multiplying 2D inputs with weight matrices without the necessity for data unrolling as seen in prior art [88]. This innovative approach leads to significant enhancements in both energy efficiency and computing throughput. The versatility of the proposed system extends to various domains, enabling applications such as classification and edge detection to be accomplished in a single iteration, thereby eliminating the requirement for multiple iterations.

The proposed array architecture is simulated, and the outcomes are presented using a model where each memristor is emulated by a resistor in MATLAB. Circuit simulation is conducted using LTspice, with the execution of a netlist generated by MATLAB. Employing the modeled array for matrix-matrix pointwise multiplication in a single shot, a CNN is developed in MATLAB for the classification of the MNIST image dataset. The methodology is meticulously explained, and comprehensive details of the classification results are provided.

3.1 Convolutional Neural Network Implementation Using Three-Dimensional Array

We designed a 3D memristor array featuring multiple input (red) and output (blue) electrodes, as depicted in Figure 3.1A. The array comprises independent row banks, each hosting several staircases. Within each row bank, each staircase establishes connections between an output electrode and multiple input electrodes through a memristor (black) situated at the cross point, with memristors stacked across different layers. Consequently, memristors are locally interconnected, mitigating the concern of sneak path leakage currents, a significant issue in passive fully connected memristor arrays [89-91]. Sneak paths represent alternative routes for current parallel to the primary current path within memristor crossbar circuits. The bidirectional nature of memristors in crossbar arrays introduces the possibility of sneak paths, which, if present, can compromise the accuracy of output current, leading to incorrect operations in memristor arrays.

Through programming the memristors to specific weights, Multiply-Accumulate (MAC) operations occur between the inputs and weights in the staircases. Subsequently, the current resulting from these MAC operations, flowing through the output electrodes, is converted to output voltages using peripheral circuits. This design facilitates fully analog-based matrix-matrix pointwise multiplication, where the 2D inputs connect to the input electrodes from the top without the need for data unrolling. The inputs are then multiplied into the weights, and the resulting output currents are collected from the output electrodes at the bottom.

Our proposed architecture finds application in representing the convolutional layer of a CNN. To illustrate the mapping of 2D inputs, outputs, and weights onto the array, an example is presented in Figure 3.1B. In this example, the weight matrix takes the form of a kernel with dimensions 2×2, and the input is an image sized at 3×3. Assuming a stride of 1 in this case, the convolutional operation output, with zero-padding, results in a 2×2 matrix. The yellow arrows visually depict how the top-leftmost portion of the input (a 2×2 matrix) is multiplied by the weight matrix.

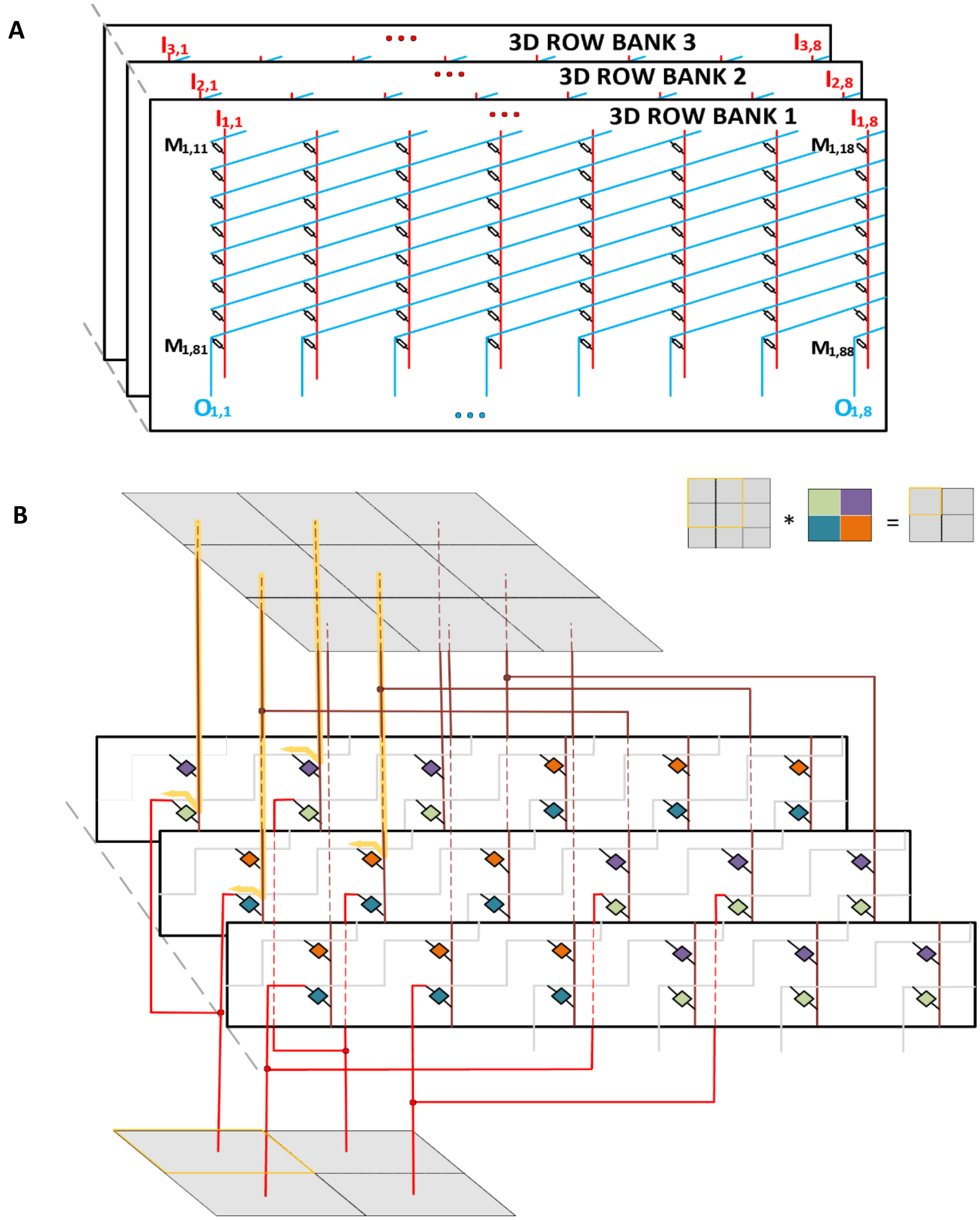


Figure 3.1: Three-Dimensional Memristor Array Architecture for Matrix-Matrix Pointwise Multiplication.

(**A**) Design of the three-dimensional memristor array. (**B**) Mapping of input, output, and weights to the array for matrix-matrix pointwise multiplication in a convolution.

To assess the functionality of the proposed array, we employed MATLAB to model it. Essentially, a circuit-based netlist is generated using MATLAB and subsequently utilized by LTspice for analog circuit simulation. The process begins with the creation of the netlist, which includes input voltage sources, 3D arrays, and TIAs. Subsequently, the weights of the array and the gain of the TIAs are incorporated into the netlist. The input voltage sources in the netlists are then assigned values based on the input voltage matrix.

Following the netlist preparation, MATLAB initiates the LTspice simulation by executing a command. After allowing LTspice some time to complete the simulation, MATLAB reads the output file generated by LTspice. The output matrix, representing the matrix-matrix pointwise multiplication of the input and weights, is then extracted from the output file.

Notably, for each pair of input and weight matrices, the entire multiplication process occurs within a single step.

3.2 MNIST Digits Feature Extraction and Classification

To benchmark our proposed design and mapping approach, we conducted a feature extraction and classification task by implementing a two-layer CNN. The CNN was tested using digits from the MNIST dataset (zero through nine). The block diagram of the CNN is illustrated in Figure 3.2. The inputs consist of 8×8 images, and the convolutional layer employs four kernels with a size of 3×3. A ReLU activation function is applied, along with a 2×2 average pooling layer in the hidden layers. The output layer comprises a 36×10 fully connected layer. To adapt the input images, we cropped and downsampled the MNIST dataset images, resizing them from 28×28 pixels to 8×8 pixels [48], which were then fed into the network as inputs.

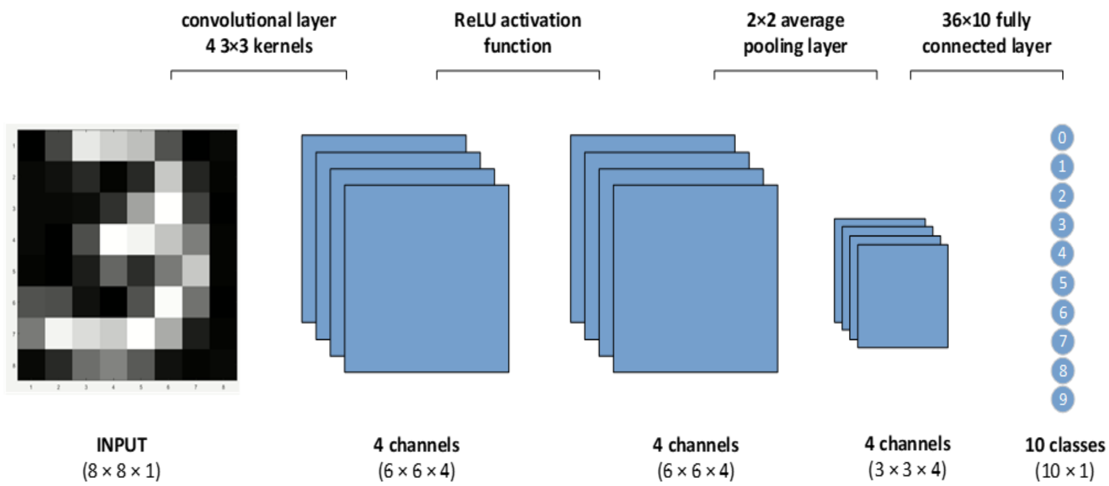


Figure 3.2: Outputs from Each Layer of the Implemented Convolutional Neural Network.

The initial input image, sized 8×8, undergoes convolution with 4 kernels of dimensions 3×3. Considering a stride of 1, the convolution yields 4 images each sized 6×6. Subsequently, ReLU activation functions are applied to the convolutional outputs. Following this, an average pooling layer with dimensions 2×2 is employed, resulting in image resizing to 3×3. Ultimately, an FC layer is employed for image classification. The FC layer receives 36 inputs, equivalent to the unrolling of 4 images sized 3×3 and produces 10 outputs corresponding to the classes 0 through 9.

The input image is directly fed into the convolutional layer (the first layer), and the ultimate output is obtained from the fully connected layer (the last layer). In between these layers, a ReLU activation function and a pooling layer are applied to represent the hidden layers, and the output from each layer is illustrated.

For the implementation of the convolutional layer in the network, our modeled 3D array is utilized. Considering the input size of the network and the mapping method, 48 input electrodes, 48 output electrodes, and 16 row banks are employed. To represent each weight in the network, accounting for the possibility of weights being either negative or positive, the output electrodes of every two adjacent row banks are connected, and their input electrodes receive inverted inputs. Consequently, two memristors positioned at the same location in these row banks constitute a differential pair [16]. This setup facilitates the representation of both positive and negative weights within the network. The presence of twice as many row banks as the number of rows in the input images serves this purpose.

In training the network, all 60000 training images from the MNIST dataset were utilized. The training process was conducted in batches with a batch size of 50 and extended over 10 epochs, totaling 12000 weight updates. The achieved accuracy during training after each weight update is depicted in Figure 3.3A.

To compute training errors using a loss function, a SoftMax function was implemented. Additionally, to determine weight gradients during the weight updating process through an optimizer, RMSProp was employed.

3.3 Inference Methodology of the Network for MNIST Digits Feature Extraction and Classification

To evaluate the inference of the implemented network, two distinct schemes were employed. These schemes differ in terms of the type of convolutional layer they utilize. The first scheme utilizes our modeled 3D array, creating netlists, conducting circuit based LTspice analyses, and reading out results, as explained in Section 3.1. Meanwhile, the second scheme executes all computations within MATLAB using the deep learning toolbox.

In the initial phase, a feature extraction task was undertaken, and the outputs from both schemes were compared. Figure 3.3B illustrates the outputs of the convolutional layers for two different input images, digit 3 and digit 7. The outputs from the convolutional layers are identical, demonstrating that both schemes exhibit similar performance in feature extraction tasks.

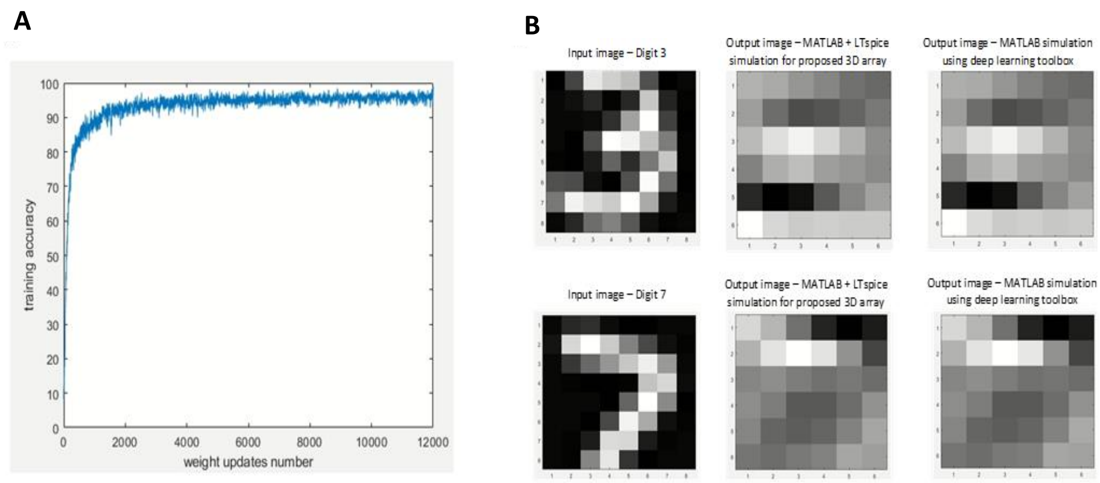


Figure 3.3: Training Results and Mid-layer Outputs Showing Feature Extraction.

(A) Accuracy progression during training after each weight update. **(B)** Feature extraction task employing two schemes for two digits from the MNIST dataset as inputs.

Subsequently, MNIST digits were employed for classification. The first scheme achieved an accuracy of 94.36%, while the second scheme achieved an accuracy of 96%. This indicates that our designed three-dimensional array and the proposed mapping method perform effectively without significant functionality degradation. They emerge as potential candidates for matrix-matrix pointwise multiplication in future experimental convolutional-based networks, offering improved energy efficiency and computing throughput without compromising functionality.

3.4 Sequence Processing

Expanding the complexity of the proposed architecture through the addition of more layers offers the opportunity to demonstrate sequence processing capabilities for future integration. The fundamental distinction between a network designed solely for image classification and one designed for sequence classification lies in the incorporation of a recurrent layer in the latter. Recurrent layers possess the ability to extract features from temporal inputs, a functionality notably absent in networks focused solely on image classification. Two prominent approaches for implementing recurrent layers are long short-term memory (LSTM) [92] and reservoir computing (RC) [93]. Notably, reservoir layers offer distinct advantages over LSTM layers as they operate without the need for explicit training. This characteristic allows reservoir layers to effectively compute intricate temporal data at a comparatively lower training cost. However, the LSTM-based networks typically outperform the RC counterparts due to their higher computational complexity.

To demonstrate the integration of the proposed architecture with a recurrent layer for sequence processing, we generated a dataset with 5958 sequences of 8×8 images for training and 1010 sequences for test using the digits 1, 2, and 3 from the MNIST dataset. Sequences of three frames were created, each comprising all three digits in various combinations. In this context, as depicted in Figure 3.4A, six distinct sequences are achievable. Employing the network illustrated in Figure 3.4B yielded an accuracy of 99.9% with LSTM and 93.56% with RC.

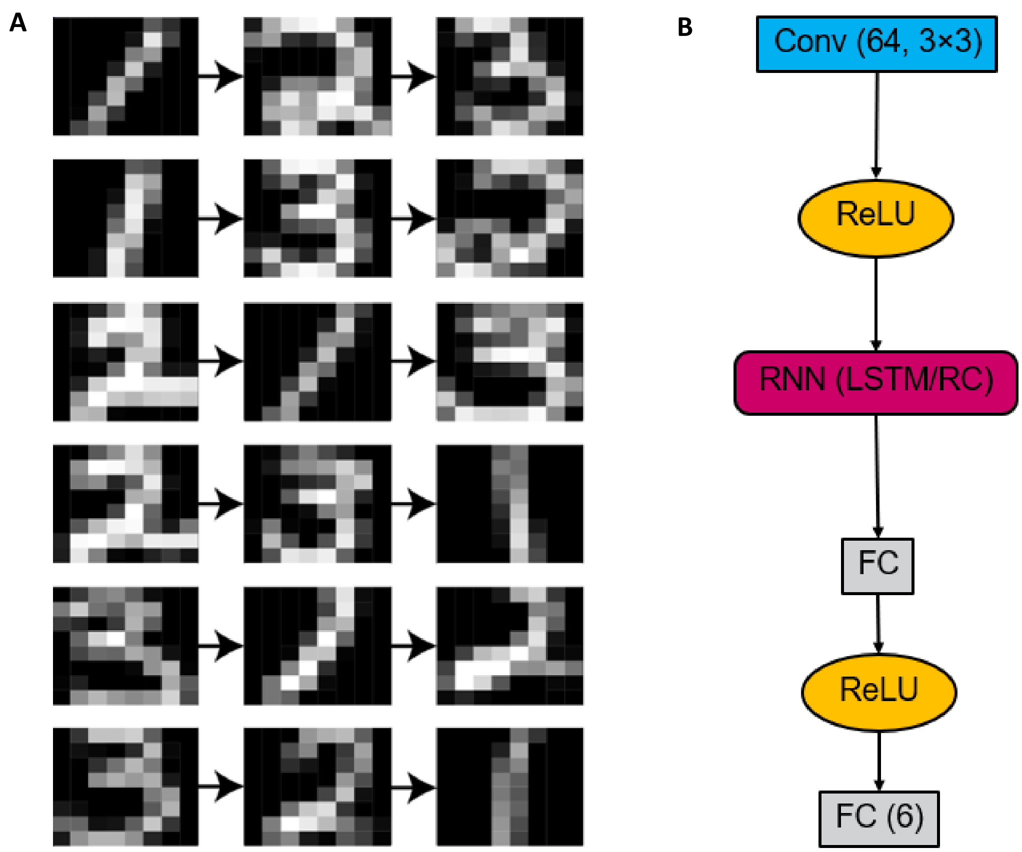


Figure 3.4: Integration of Convolutional Layer with Recurrent Layer for Sequence Classification.

(A) Dataset generation using MNIST digits. Six distinct classes represent various possible three-frame sequences of MNIST digits 1, 2, and 3, specifically created for sequence classification. **(B)** Network configuration featuring the integration of a convolutional layer with recurrent layer (LSTM and RC) for sequence processing.

In prior research, a variety of datasets have been employed for sequence classification, including moving MNIST [94-96], UCF sports clips [97-98], USF gait [92, 99-100], CASIA-B gait [100-102], and MNIST pen stroke [103] dataset. Selection of the appropriate dataset should consider its compatibility with hardware requirements.

Our study incorporates the MNIST pen stroke sequences dataset, presenting an alternative application of our proposed concept through sequence classification. This dataset, a modification of the original MNIST dataset, converts handwritten digit images into pen stroke sequences that visually represent the skeletal structures of the digits, as illustrated in Figure 3.5A. The dataset comprises 60000 training data and 10000 test data. Due to the substantial time required for simulations, we opted to utilize only the first 10000 training data and 2500 test data. Furthermore, we downsampled the images from the original size of 28×28 to 8×8, as illustrated in Figure 3.5B, to reduce simulation time and enhance feasibility for future experimental demonstrations.

For classification, the network architecture depicted in Figure 3.5C is employed, with the convolutional layer implemented using our innovative 3D array concept. The recurrent layer incorporates LSTM, resulting in an accuracy of 94.1%, surpassing the 93.4% achieved with the RC recurrent layer.

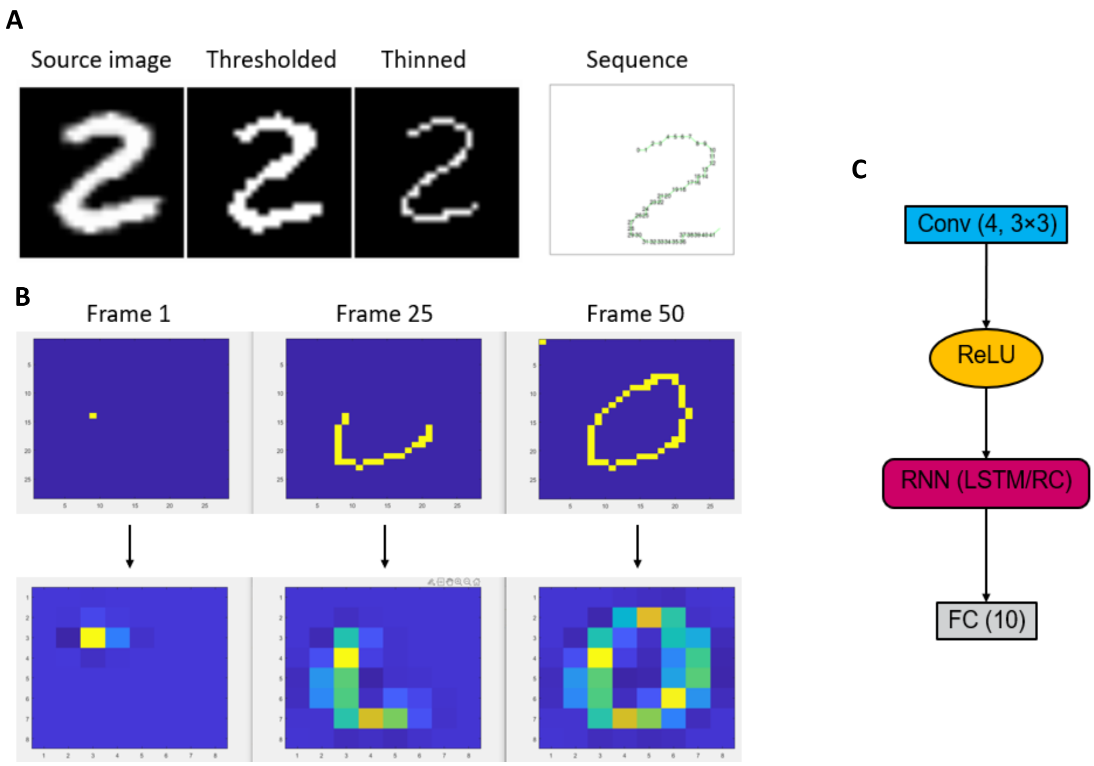


Figure 3.5: MNIST Pen Stroke Dataset Conditioning and Classification.

(A) Generation of the dataset involves thinning and preprocessing the original MNIST digits to extract their skeletal structures. **(B)** The original 28×28 images are downsized to 8×8 for effective processing. **(C)** Illustration of the network architecture employed for the classification process.

3.5 Summary

In this chapter, we introduced a 3D memristor crossbar array design along with a proposed mapping method, facilitating matrix-matrix pointwise multiplication in a single cycle. This advancement enhances the energy efficiency and computing throughput of memristor-based in-memory computing paradigms. To validate the functionality of the design, the memristor arrays were modeled in MATLAB and executed with LTspice and

MATLAB. This resulted in the successful implementation of a CNN, which was subsequently employed for MNIST feature extraction and classification tasks. The implemented network exhibited highly accurate feature extraction and achieved a recognition accuracy of 94.36%, matching the results obtained from a comparable CNN utilizing the deep learning toolbox of MATLAB. Furthermore, through the incorporation of recurrent layers, we successfully demonstrated the efficacy of the proposed system in the sequential classification task involving the pen stroke MNIST dataset, achieving a recognition accuracy of approximately 94%.

CHAPTER 4

MATRIX-MATRIX POINTWISE MULTIPLICATION WITH THREE-DIMENSIONAL PROGRAMMABLE RESISTOR ARRAYS EXPERIMENTAL IMPLEMENTATION

In this chapter, we undertake the design, modeling, and experimental implementation of our proposed 3D array architecture. Memristors are modeled using variable resistors or potentiometers, and the construction of the 3D array serves as a tangible demonstration of matrix-matrix pointwise multiplication, validating the feasibility of our concept. The potentiometers of the system are programmable digitally via a Serial Peripheral Interface (SPI) to 1024 levels. They belong to the EEPROM type offering an endurance of 200 thousand programming cycles and a data retention period of 50 years. The peripheral circuitry comprises multiplexers/demultiplexers and an ARM-core MCU responsible for programming the potentiometers. Additionally, DACs, TIAs, buffers, inverters, and ADCs are incorporated to assist in the multiplication process. The MCU plays a key role in communication with ADCs and DACs for input conditioning and output retrieval.

Furthermore, our system is designed with flexibility, allowing for the incorporation of additional layers such as ReLU activation functions, recurrent layers like reservoir computing layers, and fully connected layers. This adaptability positions the system for future applications, including but not limited to image classification and sequence processing.

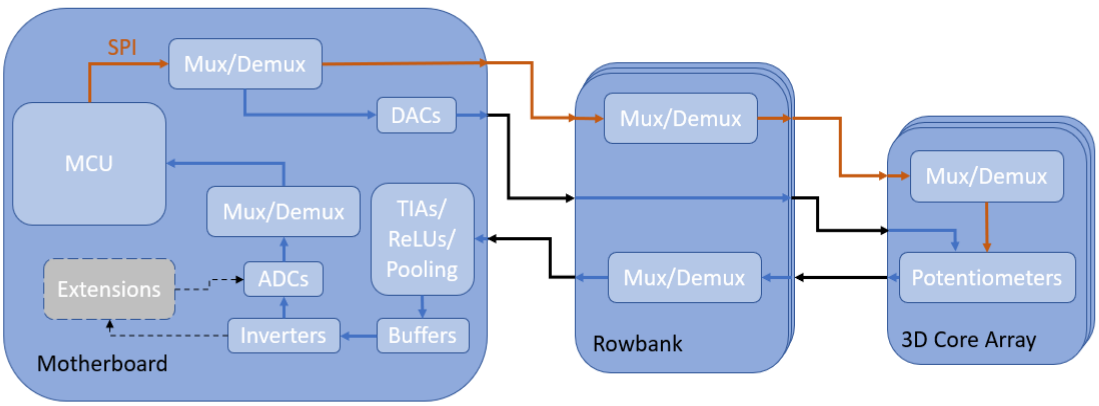
Moreover, we present insights into the deployment of the proposed hardware-centric system designed for matrix-matrix pointwise multiplication, wherein memristor devices are employed instead of resistors. Additionally, we offer preliminary assessments regarding energy consumption of the potential integrated system design utilizing the 16 nm FinFet technology node.

4.1 Experimental Implementation of the Proposed 3D Analog Multiplier

Memristor devices are renowned for their low energy consumption, high speed, and scalability. The utilization of a 3D memristor array offers a highly compact architecture capable of executing numerous operations concurrently in a notably efficient manner. However, the fabrication of such an array poses challenges and demands a considerable amount of time. Hence, in this thesis, we opt to model these memristors using digital potentiometers, essentially variable resistors controllable through digital units. This approach enables the programming of potentiometers to varying levels of resistance, resembling the capabilities of memristor devices. Subsequently, these programmable potentiometers are employed to construct the same architecture as previously proposed for the 3D memristor array.

The block diagram presenting the flow of signals in our experimentally implemented system is shown in Figure 4.1. To program the resistance of each potentiometer within the resistive array, a series of multiplexers and demultiplexers are employed throughout the system. These components are controlled by an ARM-core MCU using SPI communication protocols and Input/Output (IO) pins as shown by orange arrows. Following the programming of potentiometers to the desired resistance levels, DAC channels are programmed to convey voltage inputs to the array. The current outputs from the array undergo processing through TIAs, buffers, and inverters to be converted into voltage, preparing them for reading by ADC channels. Subsequently, the ADC channels receive and convert all signals into a digital format, which is then read by the MCU for subsequent processing.

Figure 4.1: Signal Flow of The Experimentally Implemented System.



The MCU sends the required signals through SPI protocol all the way to the potentiometers for programming as shown by orange arrows. The other signals (blue and black) are responsible for the matrix-matrix pointwise multiplication by providing the inputs and collecting the outputs. The system has the flexibility of adding more layers through connections to the block labeled as Extensions for more complex tasks.

In addition, our system incorporates ReLUs, enabling the execution of activation function operations essential for classification tasks. Furthermore, by altering the connections of inverters and ADCs, supplementary boards can be linked via the designated block named Extensions. This allows the incorporation of more complicated layers such as reservoir computing and fully connected layers, thereby enhancing the adaptability of the system and enabling the execution of more complex tasks such as image classification and sequence processing.

The system is implemented using off-the-shelf components on PCBs, comprising three distinct types of boards. Firstly, there is the core 3D array board, responsible for implementing the staircase structure described in Section 3.1. Secondly, the rowbank board is designed to integrate 24 core 3D array boards together. Lastly, the motherboard is engineered to integrate 16 rowbank boards. The subsequent sections provide a comprehensive explanation of the design and functionality of each board.

4.2 Design of the Core 3D Array Board

The potentiometers are positioned at the crosspoints of staircases, with each potentiometer connected to its individual input. However, they collectively share a common output. Illustrated in Figure 4.2A is an example of one staircase, where the output (O1) is defined as:

$$O1 = M1 \times I1 + M2 \times I2 + M3 \times I3 + M4 \times I4 + M5 \times I5 \tag{4.1}$$

where I1, I2, I3, I4, and I5 represent inputs and M1, M2, M3, M4, and M5 denote the conductance of the potentiometers. The board is depicted in Figure 4.2B which incorporates potentiometers, multiplexers/demultiplexers, and a right-angled header for connection to other boards. To facilitate independent programming of each potentiometer, the multiplexers/demultiplexers are strategically positioned on the board, as illustrated in Figure 4.2C. The board comprises four layers, with the top and bottom layers dedicated to routing signals and placing components. Meanwhile, the middle layers serve as a GND plane and a plane for routing power signals. The layout of the board, depicting signal routing, is presented in Figure 4.2D.

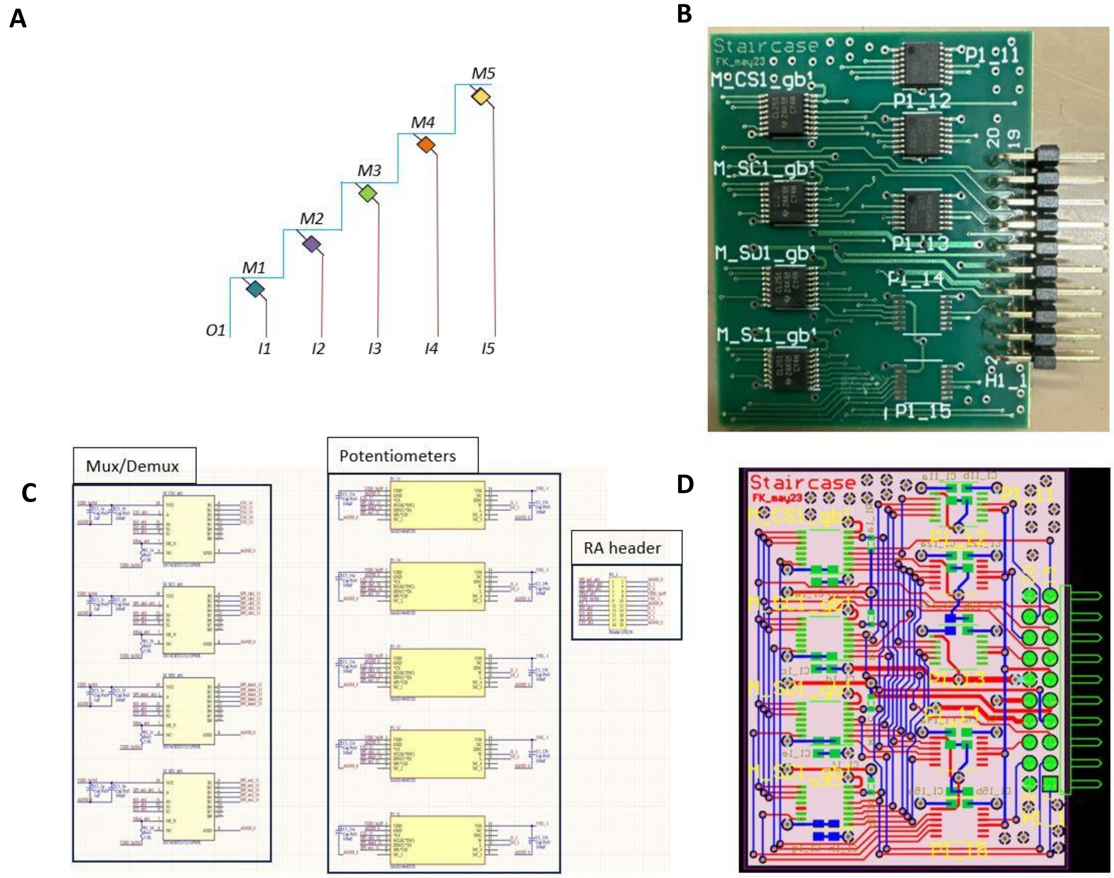


Figure 4.2: Core 3D Array Board.

(A) The board exhibits a staircase structure with one potentiometer on each crosspoint. All potentiometers share a common electrode as the output, while their other electrodes function as inputs. **(B)** Staircase PCB featuring 4 multiplexer/demultiplexers and 3 soldered potentiometers. The right-angled header facilitates connections to other boards. **(C)** Schematic representation of the core 3D array board, detailing the components. **(D)** PCB layout (48.6 mm × 39.9 mm) providing insight into the layers and signal routing of the board.

To validate the linear programmability of the potentiometers through the digital SPI, various values are programmed onto a potentiometer, and the resulting resistance is measured. As depicted in Figure 4.3A, the R-squared value is one, indicating a complete match between the actual resistance values of the potentiometer and the expected values.

Additionally, through this testing process, we determined that the resistance of the potentiometer is programmable across a range from approximately 40 Ω to 51.6 KΩ.

To assess the consistency of potentiometer resistance on a core 3D array board, considering potential impacts from signal routing on the board, all potentiometers are programmed to the same value at seven different steps, distributed randomly across the entire range from the lowest to the highest possible resistance. As illustrated in Figure 4.3B, the potentiometers exhibit uniform resistance.

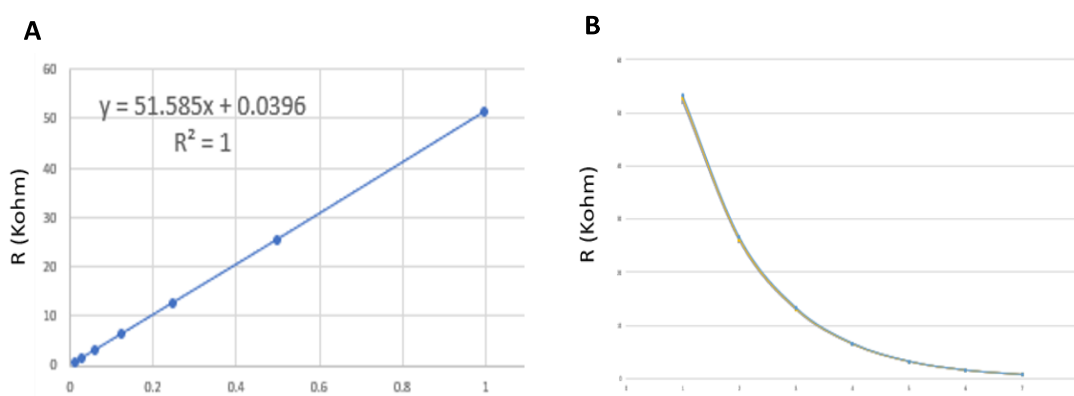


Figure 4.3: Linearity and Uniformity of Resistance of Potentiometers.

(A) The linear programmability of the potentiometers through digital programming. The resistance range is all the way from approximately 40 Ω to 51.6 KΩ. **(B)** Uniformity of resistance of potentiometers along the core 3D array.

4.3 Design of the Rowbank Board to Address the Potentiometers

As explained in Section 3.1, the rowbank boards serve as the foundation for integrating staircases together on a perpendicular plane. Essentially, multiple staircases are connected on each rowbank. While these staircases share some of the inputs, each possesses its individual output, as depicted in Figure 4.4A. Consequently, different outputs can be achieved by programming the potentiometers to varying levels of conductance. In our system, a total of 24 staircases can be connected on one rowbank through headers, as illustrated in Figure 4.4B. The schematic of the rowbank, along with the layout, is presented in Figure 4.4C and Figure 4.4D. The PCB is designed as a 4-layer board, with the top and bottom layers allocated for routing signals and placing components, while the middle layers function as the GND plane and accommodate power traces.

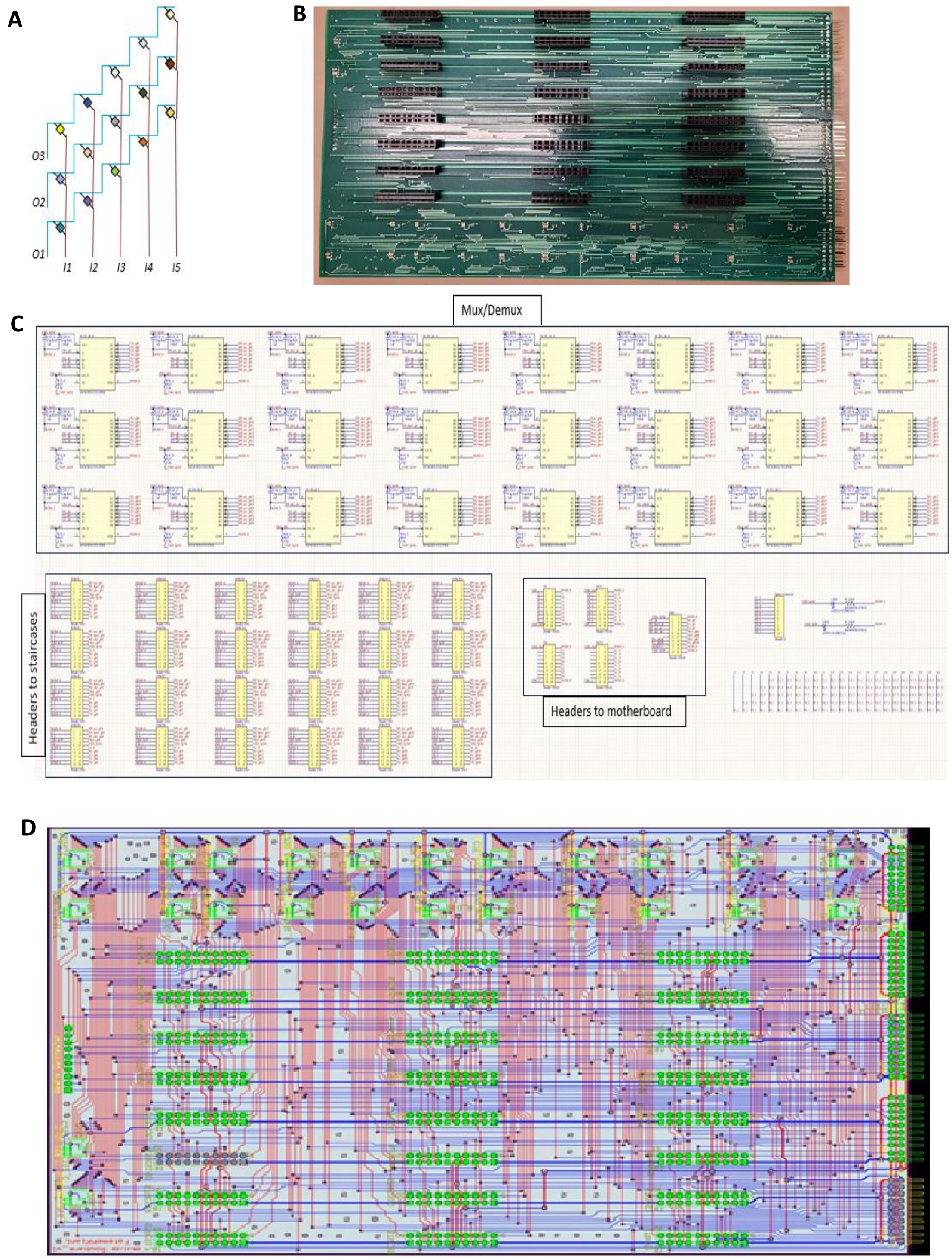


Figure 4.4: Rowbank board.

(A) The board features a staircase structure with 5 potentiometers on each staircase. All staircases share some of the inputs, and each has its individual output. **(B)** Rowbank PCB equipped with 24 headers for mounting 24 staircase boards. **(C)** Schematic representation of the rowbank board, including multiple multiplexers/demultiplexers and headers for connecting to the core 3D array boards and the motherboard. **(D)** PCB layout (235.4 mm × 159 mm) illustrating the layers and signal routing of the board.

4.4 Design of the Motherboard to Provide Inputs and Read Outputs

The motherboards serve the purpose of integrating all the rowbanks, resulting in the implementation of a 3D array. Two distinct motherboards have been designed, and each can accommodate a total of 8 rowbanks. The motherboard incorporates essential components such as ADCs, DACs, TIAs, buffers, inverters, and voltage regulators, as illustrated in Figure 4.5A. Additionally, an MCU is positioned on the bottom of the motherboard, facilitating the programming of potentiometers, DACs, ADCs, and handling further processing tasks. The layout of one of the motherboards is presented in Figure 4.5B, designed as a 6-layer board. The top (first), third, and bottom (sixth) layers are dedicated to routing signals, with two middle layers (second and fifth layers) serving as GND planes adjacent to the signal layers. The fourth layer is allocated for routing power traces, and components are strategically placed on either the top or bottom layers of the board.

A

B

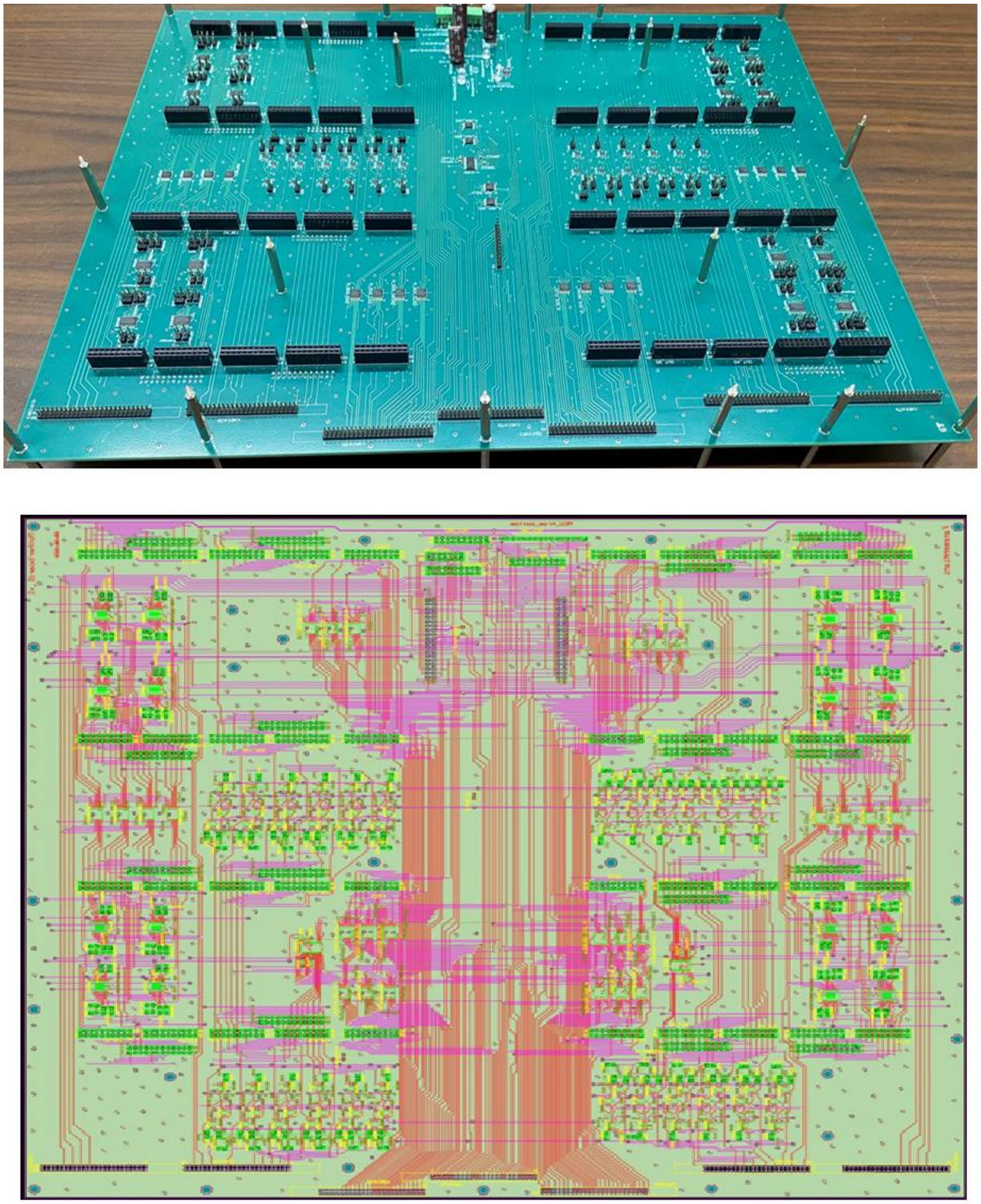


Figure 4.5: Motherboard.

(A) Motherboard PCB designed to accommodate the mounting of 8 rowbank boards. **(B)** PCB layout (438.3 mm × 403.4 mm) illustrating the layers and signal routing of the board.

4.5 Matrix-Matrix Pointwise Multiplication Demonstration

To demonstrate the functionality of the experimentally implemented system in performing matrix-matrix pointwise multiplication as a proof of concept, the focus is on a convolutional operation with a stride of 1. It is noteworthy that the system exhibits adaptability to also accommodate the stride of 2. A convolutional operation is composed of multiple matrix-matrix pointwise multiplications, and as illustrated in Figure 4.6A, the corresponding input pixels are multiplied by the kernel pixels and summed to calculate each output pixel value. Given power consumption constraints, inputs of 4×4 and kernels of 2×2 have been selected for the system. The DACs, situated on the motherboard and under the control of the MCU, are responsible for delivering inputs to the potentiometers located on the core 3D array boards. These potentiometers, in turn, utilize their conductance to implement the kernel weights. Weight programming entails the utilization of SPI, wherein the MCU placed on the motherboard transmits commands to adjust the registers of the potentiometers, effectively setting their conductance. As a result, the resistance of the wiper of the potentiometers adjusts in accordance with the received commands. The outputs are then read from TIAs, converting the current flow through all potentiometers to voltage in just a single step which highlights the superiority of our system compared to the prior art.

To showcase the functionality of the system, two simple tasks have been implemented: in the first task, the input remains fixed while various kernels are applied, while in the second task, a fixed kernel is utilized while targeting an input with varying pixel values.

Initially, an input and a kernel with fixed intensity for all pixels are applied. The outputs are then read for different kernels. Figure 4.6B demonstrates that the higher the intensity of the kernel pixels, the higher the intensity of the output pixels.

Additionally, an input with varying pixel values is convolved with a fixed kernel to further showcase the functionality. As depicted in Figure 4.6C, the higher the sum of the pixel values, the higher the output pixel value, aligning with expectations from convolutional operations.

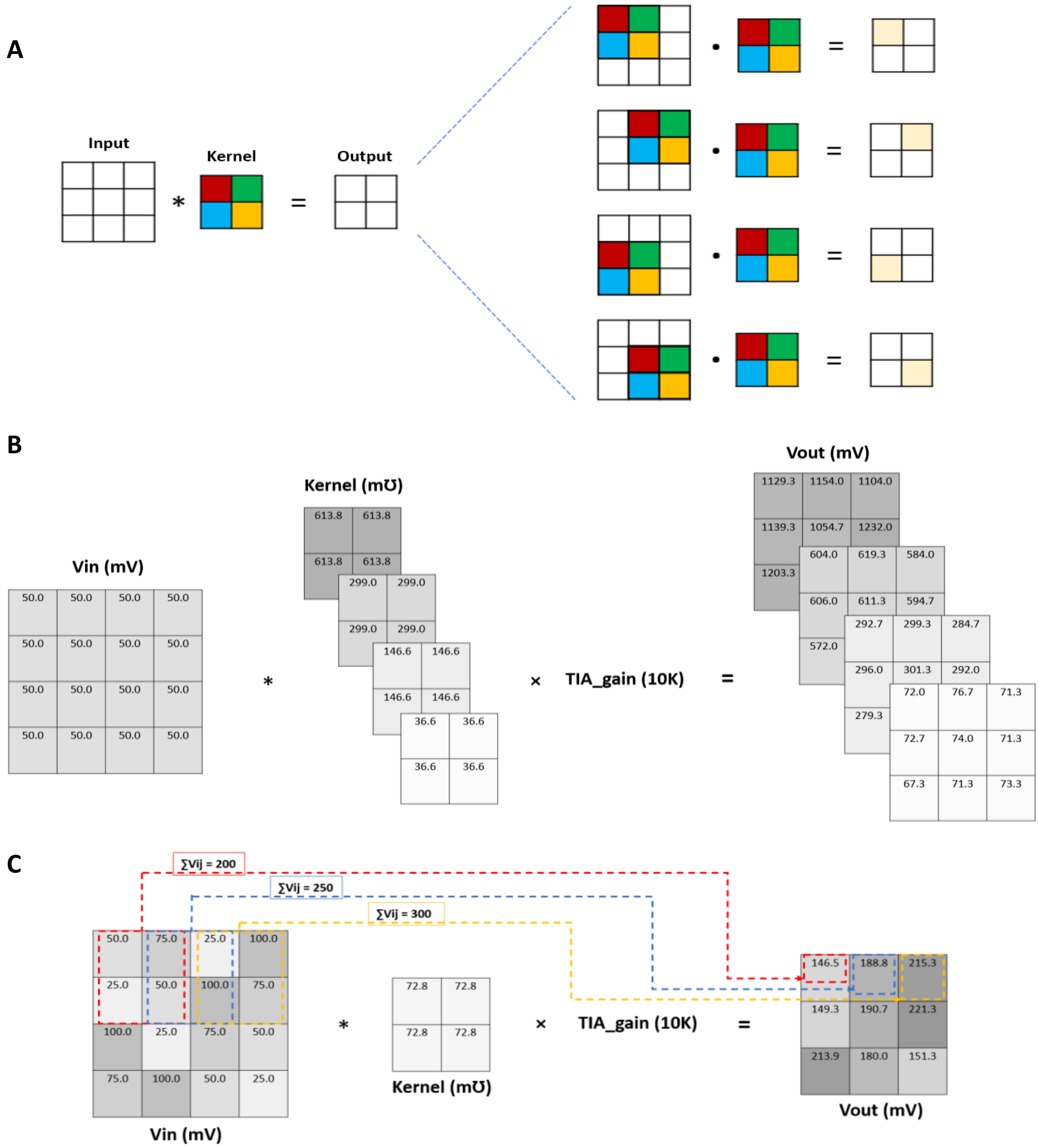


Figure 4.6: Experimental Results of Matrix-Matrix Pointwise Multiplication in Convolutional Operation.

(A) The convolutional operation consists of multiple sub operations of matrix-matrix pointwise multiplication. Utilizing a 3×3 input, a 2×2 kernel, and a stride of 1 yields a 2×2 output. On each step, one output pixel value is calculated which is the result of dot product of the kernel into 2×2 sub matrix of the input. Each step computes one output pixel value by performing a dot product of the kernel with a 2×2 sub-matrix of the input. Subsequently, a horizontal or vertical stride shift is applied to calculate the remaining output pixel values. **(B)** Experimental outcomes of the convolutional operation with various kernels applied to a fixed input. The potentiometers are programmed to the weight of kernels, and the inputs are applied by DACs. The outputs are read by TIA which converts current flowing through potentiometers to voltage by gain of 10K. As anticipated, for a constant input, higher kernel pixel values correspond to increased intensity in the output pixel values. **(C)** Experimental results of the convolutional operation with varied pixel values in the input applied to a fixed kernel. As anticipated, a higher summation of input pixel values within the corresponding sub-matrix leads to increased intensity in the resulting output pixel value.

With adequate power delivery, the system is capable of accommodating inputs as large as 8×8 and kernels of 5×5, thus enabling the demonstration of more realistic applications such as edge detection and feature extraction. Furthermore, the integration of additional layers into the system, such as fully connected layers and recurrent layers, will enhance its capability to showcase more complex applications such as image classification and sequence processing.

4.6 Memristors For Matrix-Matrix Pointwise Multiplication

The experimental implemented system explained earlier in this chapter for matrix-matrix pointwise multiplication is bulky and consumes high energy, primarily attributable to two key factors: the utilization of variable resistors and the incorporation of off-the-shelf components in the peripheral circuitry. Hence, it becomes imperative to investigate taking advantage of the benefits of memristor devices and the integrated design of peripheral circuitry.

Given that memristors offer substantial advantages for in-memory computing paradigms, it is crucial to offer a comprehensive understanding of their integration into the proposed matrix-matrix pointwise multiplication system and how this incorporation influences the performance.

Memristor devices are widely acknowledged for their low energy consumption, elevated speed, and scalability. Memristors are advancing rapidly, with their size continually decreasing. The smallest reported memristor in the literature is as small as 2 nm [104]. Furthermore, memristors exhibit remarkable power efficiency, capable of consuming as low as few μW [105]. The deployment of a 3D memristor array introduces a remarkably compact architecture that excels in executing multiple operations simultaneously with notable efficiency.

The central component of the implemented matrix-matrix pointwise multiplication system is the resistive array designed for matrix implementation. Prior to the realization of the system detailed previously in this chapter, we conceived an innovative concept for a 3D memristor architecture as the core element. However, due to the time-consuming nature of fabricating such an architecture, we opted to substitute it with variable resistors. The fundamental idea of rowbanks and staircases remains consistent, as depicted in the cartoon shown in Figure 3.1A.

Memristor devices are strategically positioned at the crosspoints of input and output electrodes. Each staircase connects multiple memristors with multiple input electrodes and an output electrode. The rowbanks play a pivotal role in assembling the 3D architecture to achieve system density. They also establish connections between staircases, facilitating the realization of the desired functionality. Inputs are applied form the top to the input electrodes and read from the bottom through the output electrodes.

4.7 Resource Estimation on Integrated Design of The System for Matrix-Matrix Pointwise Multiplication

The adoption of integrated design implementation results in a more compact and area-efficient system, as well as minimal signal degradation and latency owing to the reduction in interconnects. Additionally, a system implemented through integrated design contributes to enhanced energy efficiency by optimizing the layout and minimizing parasitic elements. The shorter interconnections and optimized layout further lead to higher operational speeds. While the fabrication of integrated designs of our system may include increased complexity and necessitate advanced manufacturing processes, it is valuable to estimate energy consumption of such a design.

To deploy a CNN with an 8×8 input, employing four 3×3 kernels with both positive and negative weights and a stride of 1 within the integrated version of our specially designed system in the 16 nm technology node, we assess the required resources. The estimation involves utilizing the power consumption values detailed in Table 4.1 for each component, with a frequency set at 2GHz.

Table 4.1: Power consumption of several state-of-the-art components in 16 nm FinFet technology node.

Component	Power Consumption (mW)
DAC_8bit [106]	50
DAC_12bit [107]	330
Amplifier [108]	0.54
ADC_8bit [109]	61.3
ADC_12bit [110]	300
SoC [111]	2100

The total energy consumption is determined by applying the following equations, taking into account two scenarios: one with 8-bit bit precision for the VMM core, and another with 12-bit bit precision for the VMM core.

$$E_{DAC\_8bit} = (Power) \times \left(\frac{1}{frequency}\right) = (50mW) \times \left(\frac{1}{2GHz}\right) = 25\ pJ \tag{4.2}$$

$$E_{DAC\_12bit} = (Power) \times \left(\frac{1}{frequency}\right) = (330mW) \times \left(\frac{1}{2GHz}\right) = 165\ pJ \tag{4.3}$$

$$\begin{aligned} E_{3D_{Array}} &= (\#\ of\ MACs) \times \left(Conductance_{average}\right) \times (Voltage_{read})^2 \times \left(\frac{1}{frequency}\right) \\ &= (2 \times 648) \times \left(\frac{1}{10Kohm}\right) \times (0.2V)^2 \times \left(\frac{1}{2GHz}\right) \\ &= 2.592\ pJ \end{aligned} \tag{4.4}$$

$$E_{amplifier} = (Power) \times \left(\frac{1}{frequency}\right) = (0.54mW) \times \left(\frac{1}{2GHz}\right) = 0.27\ pJ \tag{4.5}$$

$$E_{ADC\_8bit} = (Power) \times \left(\frac{1}{frequency}\right) = (61.3mW) \times \left(\frac{1}{2GHz}\right) = 30.65\ pJ \tag{4.6}$$

$$E_{ADC\_12bit} = (Power) \times \left(\frac{1}{frequency}\right) = (300mW) \times \left(\frac{1}{2GHz}\right) = 150\ pJ \tag{4.7}$$

$$E_{SoC\_avg} = (Power\_avg) \times \left(\frac{1}{frequency}\right) = (2.1W) \times \left(\frac{1}{2GHz}\right) = 1.05\ nJ \tag{4.8}$$

$$E_{total-1core_{8bit}}$$
$$= 2 \times 8 \times 8 \times E_{DAC_{8bit}} + E_{3D_{Array}} + 6 \times 6 \times 3 \times E_{amplifier} + 6 \times 6 \times E_{ADC_{8bit}}$$
$$+ E_{SoC_{avg}} = 5.385\, nJ \tag{4.9}$$

$$E_{total-1core\_12bit}$$
$$= 2 \times 8 \times 8 \times E_{DAC_{12bit}} + E_{3D_{Array}} + 6 \times 6 \times 3 \times E_{amplifier} + 6 \times 6 \times E_{ADC_{12bit}}$$
$$+ E_{SoC_{avg}} = 26.551\, nJ \tag{4.10}$$

Since the number of cores is equal to the number of kernels which is four, opting for an 8-bit bit precision configuration for the VMM core yields a total energy consumption of 18.39 nJ, while the 12-bit bit precision scenario results in a total energy consumption of 107.26 nJ. Notably, the DACs which provide the inputs to the system contribute significantly to the overall energy consumption as shown in Figure 4.7. In future cases where inputs are directly fed into the 3D array, the need for the DACs is eliminated and also the contribution of the SoC diminishes.

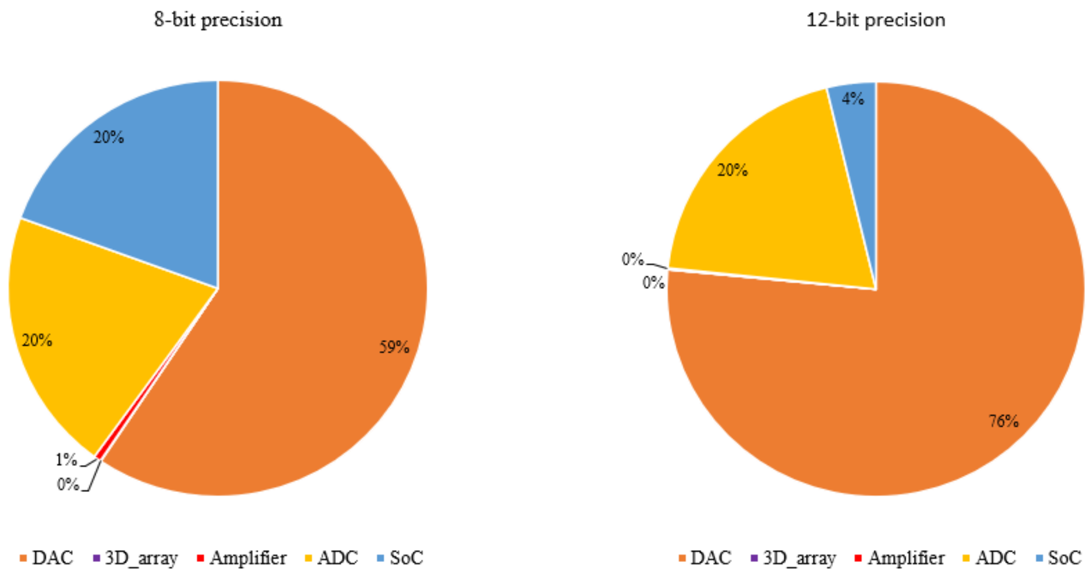


Figure 4.7: Total Energy Consumption of the Integrated System Design.

The DACs which feed the inputs to the system contribute to a substantial share of the overall system energy consumption. In future cases where inputs are directly fed into the 3D array, the need for the DAC is eliminated, and the contribution of the SoC diminishes.

4.8 Summary

This chapter introduced a proof-of-concept that highlights a sophisticated system capable of executing matrix-matrix pointwise multiplications for convolutional operations in a single cycle. The discussion went deep into the details of the system's design and functionality, accompanied by the presentation of experimental results. The implementation employed printed circuit boards and off-the-shelf components, limiting the scalability due to constraints in area and power consumption. Kernels or weights were implemented using variable resistors. To enhance system density, future studies could explore the integration of memristor devices. Additionally, an integrated design of the system holds promise for further improving density and reducing power consumption.

CHAPTER 5

FUTURE WORK AND CONCLUSION

5.1 Summary of Contributions

In this dissertation, we address the challenges associated with the implementation of entirely hardware-based in-memory computing systems. These challenges impede the systems from achieving full hardware-based functionality, resulting in additional resource utilization.

In Chapter 2, we described our developed compact fully analog-based ReLUs which were employed as hidden neurons in a fully hardware-based perceptron designed for the classification of MNIST images. The utilization of our developed ReLUs in neural network demonstrations yields better performance in terms of power consumption, speed, and area efficiency. These performance enhancements hold substantial significance, particularly in the context of deep neural networks that demand numerous layers for effective operation.

In Chapter 3, we introduced a 3D array architecture designed to execute a substantial number of MAC operations efficiently. This array has the capability to carry out matrix-matrix pointwise multiplication of input voltages to the weight matrix within a single cycle. The adoption of this array leads to enhanced performance in terms of energy efficiency, throughput, and speed. To benchmark the proposed design, we employed LTspice to model memristors with resistors and integrated them with MATLAB to demonstrate the functionality for three distinct tasks: feature extraction, image classification, and sequence classification.

In Chapter 4, we delved into the practical implementation of our proposed 3D array architecture and provided a comprehensive overview of the methodology employed to realize it. Through the utilization of our implemented array architecture in conjunction with purpose-designed peripheral circuitry, we demonstrated the execution of matrix-matrix pointwise multiplication operations in convolution. Our findings revealed the ability to conduct all the necessary multiplications for convolution within a single cycle or step, marking a significant advancement in expediting the operations of analog computing-based systems and progressing towards the realization of fully analog computing systems in the future. Moreover, we investigated the feasibility of incorporating memristor devices into the array for upcoming demonstrations, and we estimated the resource utilizations and performance of the integrated circuit designs of our implemented system, employing the 16 nm technology node.

5.2 Applications and Future Directions Based on This Dissertation

The evolution towards 3D in-memory computing systems promises a groundbreaking shift in system-on-chip design for the future. Such systems are poised to offer a range of key functionalities:

Primarily, harnessing their extensive on-chip storage capacity, they effortlessly accommodate the weight requirements of large-scale neural networks.

Furthermore, by the physical implementation of intricate neural circuits, these systems enable efficient in-memory processing, allowing for the execution of complex computational tasks with exceptional efficiency.

Additionally, through the integration of sensing and computing functionalities, they enable the seamless processing of high-volume data without incurring significant communication overheads. This integration optimizes data processing pipelines and elevates overall system performance.

Despite the early potential demonstrated by 3D circuits in providing high density and design flexibility for in-memory computing, their practical realization presents a host of challenges spanning devices, circuits, processes, and beyond. Unlike conventional 2D circuits, 3D circuits involve the integration of multiple device layers fabricated through a series of manufacturing processes, necessitating meticulous design and optimization efforts to meet thermal budget constraints and enhance fabrication yield. Moreover, thermal dissipation emerges as a potential bottleneck for 3D circuits as different layers are stacked atop one another [50].

The proof-of-concept realization of our developed fully analog ReLU and the hardware system for matrix-matrix pointwise multiplication, executed in a single iteration, represents a pivotal milestone that lays the groundwork for future implementations of three-dimensional dense, power-efficient, and high-throughput fully analog in-memory computing systems. Transitioning from variable resistors to memristor devices, coupled with integrated circuit design, is anticipated to yield substantial enhancements.

Moving forward, one of the key future directions based on the work presented in this dissertation is the implementation of matrix-matrix pointwise multiplication using the memristor-based version of the developed three-dimensional array to achieve superior performance. However, this endeavor will require complex fabrication techniques due to the necessity of numerous masks and etching processes for each layer. Additionally, developing efficient peripheral circuitry to program the memristor devices and devising novel approaches to mitigate sneak path current represent significant challenges.

Another promising direction involves integrating sensor arrays with the three-dimensional array to directly receive input data, eliminating the need for digital preprocessing. While this integration promises to enhance overall system delay and latency, it also introduces challenges such as noise and degradation that require careful consideration.

Furthermore, the three-dimensional array, primarily designed for convolutional operations of neural networks, can be extended to integrate additional layers such as LSTM, reservoir computing, and fully connected layers to enable more advanced applications.

The comprehensive system developed atop the proof-of-concept presented in this dissertation holds promise for a variety of applications, including edge detection, image classification, sequence processing, video processing, and beyond, showcasing the versatility and potential impact of this transformative technology.

5.3 Applications and Future Directions on Analog In-Memory Computing Systems

Fully analog in-memory computing systems hold significant appeal in both academic research and industrial applications due to their potential advantages. These systems find utility in scenarios where energy efficiency, parallel processing, minimal data movements, high throughput, and scalability are paramount considerations [112].

Anticipated research trajectories are to enhance the energy efficiency of analog in-memory computing systems. This requires the development of low-power analog devices, exploration of novel materials, and optimization of analog circuit design like development of novel three-dimensional integrated circuits to minimize energy consumption while preserving computational powers.

Analog in-memory computing emerges as a potent tool for the effective deployment of various neural network models. The future landscape involves refining algorithms designed specifically for analog in-memory computing, including novel strategies for mapping neural network computations onto analog hardware and adapting training algorithms to leverage analog system features [113-116].

Moreover, advancements in non-volatile memory technologies, including resistive switching devices and memristors, play a pivotal role in the evolution of analog in-memory computing systems [117-118]. Subsequent efforts may focus on enhancing the reliability, speed, and scalability of these memory technologies for neural network applications.

Additionally, analog in-memory computing proves well-suited for real-time and edge computing applications [119-120]. Future trajectories may include designing systems proficient in efficiently processing neural network computations at the edge, mitigating data transfer requirements, and enhancing application responsiveness.

It is worth mentioning that while current emphasis predominantly centers on deep learning, future investigations may explore applications of analog in-memory computing beyond conventional neural network models. This encompasses potential applications in optimization problems, cognitive computing [121], and diverse AI tasks. Analog computing also excels in signal processing tasks [122-125], including sensor data analysis, image processing, and audio signal processing. The continuous nature of analog signals enables the efficient handling of continuous data streams.

Acknowledging the promising advantages, it is imperative to recognize that fully analog in-memory computing confronts several challenges, including precision issues, noise interference, and the necessity for specialized hardware. However, the dynamic nature of this field is actively addressing these obstacles through ongoing research endeavors, with the goal of fully unlocking the potential of analog computing across a spectrum of applications.

To tackle precision issues, one strategy involves the development of memristors capable of exhibiting a high number of distinguishable conductance levels, coupled with the implementation of high-precision peripheral circuitry. This approach aims to enhance the accuracy and reliability of analog computations.

Addressing noise interference represents another significant challenge. Engineering memristors to achieve higher levels of distinguished conductance can mitigate the impact of noise, thereby improving the signal-to-noise ratio and enhancing the overall performance of analog computing systems. Additionally, employing integrated design instead of discrete implementations and implementing proper layout techniques can contribute to noise reduction. Another critical aspect to consider is the sneak current path problem, where adjacent memristors' current contributes to the overall results, introducing unwanted noise. Various approaches have been proposed to overcome this challenge, including the integration of selectors such as transistors or diodes, alternative programming schemes like the half biasing scheme or 1/3V biasing scheme, and locally connecting cells using architectures such as the staircase architecture like what we have utilized in this dissertation. However, there remains a need for continuous innovation to develop novel approaches and architectures that further minimize these noise-induced effects.

Furthermore, overcoming the requirement for specialized hardware presents an opportunity for innovation. Flexibility in model architectures, such as the utilization of dynamic neural network layers like reservoir computing layers instead of traditional LSTM layers, offers a promising avenue. Additionally, the adoption of reconfigurable hardware platforms like Field-Programmable Gate Arrays (FPGAs) instead of conventional MCUs for post-processing tasks can provide greater adaptability and efficiency in analog computing environments.

By addressing these challenges through innovative approaches and technological advancements, the field of analog in-memory computing is poised to realize its full potential and revolutionize various domains ranging from artificial intelligence to data processing and beyond.

BIBLIOGRAPHY

[1] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: applications in industry, business and science," Communications of the ACM, vol. 37, no. 3, pp. 93-106, Mar. 1994.

[2] W. He, "Load forecasting via deep neural networks," Procedia Computer Science, vol. 122, pp. 308-314, Jan. 2017.

[3] H. Xiao, F. Zhang, Z. Shen, K. Wu, and J. Zhang, "Classification of weather phenomenon from images by using deep convolutional neural network," Earth and Space Science, vol. 8, no. 5, pp. e2020EA001604, May 2021.

[4] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proceedings of the IEEE, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.

[5] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, and A. Peters, "A review of deep learning with special emphasis on architectures, applications and recent trends," Knowledge-Based Systems, vol. 194, pp. 105596, Apr. 2020.

[6] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, "Recent advances and applications of machine learning in solid-state materials science," npj Computational Materials, vol. 5, no. 1, pp. 1-36, Aug 2019.

[7] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," Heliyon, vol. 4, no. 11, pp. e00938, Nov 2018.

[8] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641-646, Jan. 2020.

[9] S. Lim, J. H. Bae, J. H. Eum, S. Lee, C. H. Kim, D. Kwon, and J. H. Lee, "Hardware-based neural networks using a gated Schottky diode as a synapse device," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2018.

[10] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Davila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Willams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," Nature Electronics, vol. 1, no. 1, pp. 52-59, Dec. 2018.

[11] N. Athreyas, W. Song, B. Perot, Q. Xia, A. Mathew, J. Gupta, D. Gupta, and J. J. Yang, "Memristor-CMOS analog coprocessor for acceleration of high-performance computing applications," ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 14, no. 3, pp. 1-30, Nov. 2018.

[12] M. Hu, C. E. Graves, C. Li, Y. Li, N. Ge, E. Montgomery, N. Davila, H. Jiang, R. S. Williams, J. J. Yang, and Q. Xia, "Memristor-based analog computation and neural

network classification with a dot product engine," Advanced Materials, vol. 30, no. 9, pp. 1705914, Mar. 2018.

[13] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–CMOS system for efficient multiply–Accumulate operations," Nature Electronics, vol. 2, no. 7, pp. 290-299, Jul. 2019.

[14] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," Nature Nanotechnology, vol. 12, no. 8, pp. 784-789, Aug. 2017.

[15] Y. Jeong, J. Lee, J. Moon, J. H. Shin, and W. D. Lu, "K-means data clustering with memristor networks," Nano Letters, vol. 18, no. 7, pp. 4447-4453, Jun. 2018.

[16] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "fully hardware implemented memristor convolutional neural network," Nature, vol. 577, pp. 641-646, Jan. 2020.

[17] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," Nature Electronics, vol. 1, no. 1, pp. 52-59, Jan. 2018.

[18] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks," Nature Communications, vol. 9, no. 1, pp. 2385, Jun. 2018.

[19] J. Cui, and Q. Qiu, "Towards memristor based accelerator for sparse matrix vector multiplication," in IEEE International Symposium on Circuits and Systems (ISCAS), pp. 121-124, May 2016.

[20] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," Nature Electronics, vol. 1, pp. 22-29, 2018.

[21] T. F. Schranghamer, A. Oberoi, and S. Das, "Graphene memristive synapses for high precision neuromorphic computing," Nature Communications, vol. 11, no. 1, pp. 5474, Oct. 2020.

[22] C. X. Xue, Y. C. Chiu, T. W. Liu, T. Y. Huang, J. S. Liu, T. W. Chang, H. Y. Kao, J. H. Wang, S. Y. Wei, C. Y. Lee, S. P. Huang, J. M. Hung, S. H. Teng, W. C. Wei, Y. R. Chen, T. H. Hsu, Y. K. Chen, Y. C. Lo, T. H. Wen, C. C. Lo, R. S. Liu, C. C. Hsieh, K. T. Tang, M. S. Ho, C. Y. Su, C. C. Chou, Y. D. Chih, and M. F. Chang, "A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices," Nature Electronics, vol. 4, no. 1, pp. 81-90, Jan. 2020.

[23] W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M. F. Chang, H. J. Yoo, H. Qian, and H. Wu, "Neuro-inspired computing chips," Nature Electronics, vol. 3, no. 7, pp. 371-382, Jul. 2020.

[24] J. Hung, X. Li, J. Wu, and M. Chang, "Challenges and trends in developing nonvolatile memory-enabled computing chips for intelligent edge devices," IEEE Transactions on Electron Devices, vol. 67, no. 4, pp. 1444-1453, Mar. 2020.

[25] W. H. Chen, C. Dou, K. X. Li, W. Y. Lin, P. Y. Li, J. H. Huang, J. H. Wang, W. C. Wei, C. X. Xue, Y. C. Chiu, Y. C. King, C. J. Lin, R. S. Liu, C. C. Hsieh, K. T. Tang, J. J. Yang, M. S. Ho, and M. F. Chang, "CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors," Nature Electronics, vol. 2, no. 9, pp. 420-428, Sep. 2019.

[26] C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "In-memory computing with memristor arrays," in Proceedings of the IEEE International Memory Workshop (IMW), pp. 1-4, May 2018.

[27] W. C. Wei, C. J. Jhang, Y. R. Chen, C. X. Xue, S. H. Sie, J. L. Lee, H. W. Kuo, C. C. Lu, M. F. Chang, and K. T. Tang, "A relaxed quantization training method for hardware limitations of resistive random access memory (ReRAM)-based computing-in-memory," IEEE Journal on Exploration Solid-State Computation Devices Circuits, vol. 6, no. 1, pp. 45-52, May 2020.

[28] B. Yan, B. Li, X. Qiao, C. X. Xue, M. F. Chang, Y. Chen, and H. Li, "Resistive memory-based in-memory computing: From device and large-scale integration system perspectives," Advanced Intelligent Systems, vol. 1, no. 7, pp. 1900068, Nov. 2019.

[29] Y. C. Chiu, Z. Zhang, J. J. Chen, X. Si, R. Liu, Y. N. Tu, J. W. Su, W. H. Huang, J. H. Wang, W. C. Wei, J. M. Hung, S. S. Sheu, S. H. Li, C. I. Wu, R. S. Liu, C. C. Hsieh, K. T. Tang, and M. F. Chang, "A 4-Kb 1-to-8-bit configurable 6T SRAM-based computation-in-memory unit-macro for CNN-based AI edge processors," IEEE Journal on Solid State Circuits, vol. 55, no. 10, pp. 2790-2801, Jul. 2020.

[30] C. X. Xue, W. H. Chen, J. S. Liu, J. F. Li, W. Y. Lin, W. E. Lin, J. H. Wang, W. C. Wei, T. Y. Huang, T. W. Chang, T. C. Chang, H. Y. Kao, Y. C. Chiu, C. Y. Lee, Y. C. King, C. J. Lin, R. S. Liu, C. C. Hsieh, K. T. Tang, and M. F. Chang, "Embedded 1-Mb ReRAM-based Computing-in Memory macro with multibit input and weight for CNN-based AI edge processors," IEEE Journal on Solid State Circuits, vol. 55, no. 1, pp. 203-215, Nov. 2020.

[31] K. Berggren, Q. Xia, K. K. Likharev, D. B. Strukov, H. Jiang, T. Mikolajick, D. Querloiz, M. Salinga, J. R. Erickson, S. Pi, F. Xiong, P. Lin, C. Li, Y. Chen, S. Xiong, B. D. Hoskins, M. W. Daniels, A. Madhavan, J. A. Liddle, J. J. McClelland, Y. Yang, J. Rupp, S. S. Nonnenmann, K. T. Cheng, N. Gong, M. A. L. Montaño, A. A. Talin, A. Salleo, B. J. Shastri, T. F. de Lima, P. Prucnal, A. N. Tait, Y. Shen, H. Meng, C. R. Carmes, Z. Cheng, H. Bhaskaran, D. Jariwala, H. Wang, J. M. Shainline, K. Segall, J. J. Yang, K. Roy, S. Datta, and A. Raychowdhury, "Roadmap on emerging hardware and technology for machine learning," Nanotechnology, vol. 32, no. 1, pp. 012002, Oct. 2020.

[32] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," Nature Communications, vol. 9, no. 1, pp. 2331, Jun. 2018.

[33] O. Krestinskaya, K. N. Salama, and A. P. James, "Learning in memristive neural network architectures using analog backpropagation circuits," IEEE Transactions on Circuits Systems I Regular Pap., vol. 66, no. 2, pp. 719-732, Sep. 2019.

[34] B. Nursultan, and O. Krestinskaya, "Perceptron Linear Activation Function Design with CMOS-Memristive Circuits," in Proceedings of the International Conference on Computing and Network Communications (CoCoNet), pp. 231-234, Aug. 2018.

[35] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," Towards Data Science, vol. 6, no. 12, pp. 310-316, Sep. 2017.

[36] B. Karlik, and A. V. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," International Journal of Artificial Intelligence and Expert Systems, vol. 1, no. 4, pp. 111-122, Feb. 2011.

[37] P. Sibi, S. A. Jones, and P. Siddarth, "Analysis of different activation functions using back propagation neural networks," Journal of Theoretical and Applied Information Technology, vol. 47, no. 3, pp. 1264-1268, Jan. 2013.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Communications on the ACM, vol. 60, no. 6, pp. 84-90, May 2017.

[39] S. Ioffe, and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv Preprint, vol. 1502.03167, Feb. 2015.

[40] X. Liu, and X. Di, "TanhExp: A smooth activation function with high convergence speed for lightweight neural networks," arXiv Preprint, vol. 2003.09855, Sep. 2020.

[41] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on MNIST classification task," arXiv Preprint, vol. 1804.02763, Apr. 2018.

[42] B. Neyshabur, Y. Wu, R. R. Salakhutdinov, and N. Srebro, "Pathnormalized optimization of recurrent neural networks with relu activations," Advances in Neural Information Processing Systems, vol. 29, pp. 3477-3485, 2016.

[43] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in Proceedings of icml, vol. 30, no. 1, pp. 3, Jun. 2013.

[44] Y. Bengio, I. Goodfellow, and A. Courville, "Deep learning," vol. 1, Massachusetts, USA: MIT Press, 2017.

[45] A. Singh, S. Diware, A. Gebregiorgis, R. Bishnoi, F. Catthoor, R. V. Joshi, and S. Hamdioui, " Low-power memristor-based computing for edge-ai applications," In IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2021.

[46] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Davila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Willams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," Nature Electronics, vol. 1, no. 1, pp. 52-59, Dec. 2018.

[47] A. Amirsoleimani, F. Alibart, V. Yon, J. Xu, M. R. Pazhouhandeh, S. Ecoffey, Y. Beilliard, Roman Genov, and Dominique Drouin, "In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal-Oxide-Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives," Advanced Intelligent Systems, vol. 2, no. 11, pp. 2000115, Nov. 2020.

[48] F. Kiani, J. Yin, Z. Wang, J. J. Yang, and Q. Xia, "A fully hardware-based memristive multilayer neural network," Science advances, vol. 7, no. 48, pp. eabj4801, Nov. 2021.

[49] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations," Nature Electronics, vol. 2, no. 7, pp. 290-299, Jul. 2019.

[50] P. Lin, C. Li, Z. Wang, Y. Li, H. Jiang, W. Song, M. Rao, Y. Zhuo, N. K. Upadhyay, M. Barnell, Q. Wu, J. J. Yang, and Q. Xia, "Three-dimensional memristor circuits as complex neural networks," Nature Electronics, vol. 3, no. 4, pp. 225-232, Apr. 2020.

[51] Q. Huo, Y. Yang, Y. Wang, D. Lei, X. Fu, Q. Ren, X. Xu, Q. Luo, G. Xing, C. Chen, X. Si, H. Wu, Y. Yuan, Q. Li, X. Li, X. Wang, M. F. Chang, F. Zhang, and M. Liu, "A computing in-memory macro based on three-dimensional resistive random-access memory," Nature Electronics, vol. 5, no. 7, pp. 469-477, Jul. 2022.

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings on IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[53] F. Kiani, J. Yin, Z. Wang, J.J. Yang, and Q. Xia, “All hardware-based two-layer perceptron implemented in memristor crossbar arrays,” in IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2021.

[54] F. Kiani, A. Jaberirad, Y. Huang, and Q. Xia, “Matrix-matrix-multiplication with Three-dimensional Memristor Arrays,” In Preparation

[55] C. Yakopcic, T. M. Taha, and R. Hasan, "Hybrid crossbar architecture for a memristor based memory," in Proceedings of the IEEE National Aerospace and Electronics Conference, pp. 237-242, Jun. 2014.

[56] M. Zangeneh, and A. Joshi, "Performance and energy models for memristor-based 1T1R RRAM cell," in Proceedings of the Great Lakes Symposium on VLSI, pp. 9-14, May 2012.

[57] C. Nauenheim, "Integration of Resistive Switching Devices in Crossbar Structures," Forschungszentrum Jülich GmbH, Zentralbibliothek, vol. 10, 2010.

[58] B. Razavi, "Fundamentals of Microelectronics," John Wiley & Sons, Apr. 2021.

[59] R. G. D. Steel, and J. H. Torrie, "Principles and Procedures of Statistics," McGraw Hill Book Company, 1960.

[60] N. R. Draper, and H. Smith, "Applied Regression Analysis," John Wiley & Sons, Apr. 1998.

[61] A. P. Barten, R. Heijmans, and H. Neudecker, "The coefficient of determination for regression without a constant term," in The Practice of Econometrics: Studies on Demand, Forecasting, Money, and Income, pp. 181-189, 1987.

[62] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, "Deep Learning," The MIT Press, Jun. 2016.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by backpropagating errors," Nature, vol. 323, no. 6088, pp. 533-536, Oct. 1986.

[64] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for backpropagation," in Proceedings of the 1988 Connectionist Models Summer School, vol. 1, pp. 21-28, Jun. 1988.

[65] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: A theoretical and experimental comparison," Interspeech, vol. 13, pp. 1756-1760, Aug. 2013.

[66] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," arXiv Preprint, vol. 1902.09843, Feb. 2019.

[67] C. Igel, and M. Hüsken, "Improving the Rprop learning algorithm," in Proceedings of the Second International ICSC Symposium on Neural Computation (ICSC Academic Press), vol. 2000, pp. 115-121, 2000.

[68] R. V. K. OReddy, B. S. Rao, and K. P. Raju, "Handwritten hindi digits recognition using convolutional neural network with rmsprop optimization," in Second International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 45-51, Jun. 2018.

[69] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," arXiv Preprint, vol. 1412.6980, Dec. 2014.

[70] T. Tieleman, and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural networks for machine learning, vol. 4, no. 2, pp. 26-31, Oct. 2012.

[71] F. Zou, L. Shen, Z. Jie, W. Zhang, and W. Liu, "A sufficient condition for convergences of adam and rmsprop," Proceedings of IEEE Conference on Computer Vision Pattern Recognition, vol. 1, pp. 11119-11127, 2019.

[72] M. C. Mukkamala, and M. Hein, "Variants of RMSProp and Adagrad with logarithmic regret bounds," arXiv Preprint, vol. 1706.05507, Jun. 2017.

[73] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv Preprint, vol. 1609.04747, Sep. 2016.

[74] Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. P. Strachan, N. Ge, N. McDonald, Q. Wu, M. Hu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, "In situ training of feed-forward and recurrent convolutional memristor networks," Nature Machine Intelligence, vol. 1, no. 9, pp. 434-442, Sep. 2019.

[75] W. Wu, H. Wu, B. Gao, P. Yao, X. Zhang, X. Peng, S. Yu, and H. Qian, "A methodology to improve linearity of analog RRAM for neuromorphic computing," in IEEE Symposium on VLSI Technology, pp. 103-104, Jun. 2018.

[76] Y. Cheng, "The influence and modeling of process variation and device mismatch for analog/rf circuit design," in Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems, no. 02TH8611, pp. D046, Apr. 2002.

[77] W. Tian, P. Steinmann, E. Beach, I. Khan, and P. Madhani, "Mismatch characterization of a high precision resistor array test structure," in Proceedings of the IEEE International Conference on Microelectronic Test Structures, pp. 11-16, Mar. 2008.

[78] H. Thibieroz, P. Shaner, and Z. C. Butler, "Mismatch and flicker noise characterization of tantalum nitride thin film resistors for wireless applications," in Proceedings of the 2001 International Conference on Microelectronic Test Structures, pp. 207-212, Mar. 2001.

[79] H. Hung, and V. Adzic, "Monte carlo simulation of device variations and mismatch in analog intregrated circuits," in Proceedings of the National Conference on Undergraduate Research (NCUR), pp. 1-8, Apr. 2006.

[80] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C. X. Xue, W. H. Chen, J. Tang, Y. Wang, M. F. Chang, H. Qian, and H. Wu, "A fully integrated analog ReRAM based 78.4TOPS/W compute-in-memory chip with fully parallel MAC computing," in Proceedings of the IEEE International Solid-State Circuits Conference, pp. 500-502, Feb. 2020.

[81] Analog Devices, Inc., "High speed op amps (bandwidth>50MHz), LTC6268/LTC6269 datasheet," (Publication 62689f, 2014); www.analog.com/media/en/technicaldocumentation/data-sheets/62689f.pdf.

[82] W. Yan, R. Kolm, and H. Zimmermann, "A low-voltage low-power fully differential rail-to-rail input/output opamp in 65-nm CMOS," in IEEE International Symposium on Circuits and Systems, pp. 2274-2277, May 2008.

[83] D. Li, Z. Zhu, R. Ding, and Y. Yang, "A 1.4-mW 10-bit 150-MS/s SAR ADC with nonbinary split capacitive DAC in 65-nm CMOS," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 11, pp. 1524-1528, Sep. 2018.

[84] A. Bhide, O. E. Najari, B. Mesgarzadeh, and A. Alvandpour, "An 8-GS/s 200-MHz Bandwidth 68-mW ΔΣ DAC in 65-nm CMOS," IEEE Transaction on Circuits Systems II Express Briefs, vol. 60, no. 7, pp. 387-391, Apr. 2013.

[85] J. Semmlow, "Circuits, Signals, and Systems for Bioengineers," Academic Press, 2005.

[86] F. Zhou, and Y. Chai, "Near-sensor and in-sensor computing," Nature Electronics, vol. 3, no. 11, pp. 664-671, Nov. 2020.

[87] L. Mennel, J. Symonowicz, S. Wachter, D. K. Polyushkin, A. J. Molina-Mendoza, and T. Mueller, "Ultrafast machine vision with 2D material neural network image sensors," Nature, vol. 579, pp. 62-66, Mar. 2020.

[88] P. Lin, C. Li, Z. Wang, Y. Li, H. Jiang, W. Song, M. Rao, Y. Zhuo, N. K. Upadhyay, M. Barnell, Q. Wu, J. J. Yang, Q. Xia, " Three-dimensional memristor circuits as complex neural networks," Nature Electronics, vol. 3, no. 4, pp. 225-232, Apr. 2020.

[89] C. M. Jung, J. M. Choi, and K. S. Min, "Two-step write scheme for reducing sneak-path leakage in complementary memristor array," IEEE transactions on nanotechnology, vol. 11, no. 3, pp. 611-618, Feb. 2012.

[90] M. A. Zidan, H. A. H. Fahmy, M. M. Hussain, and K. N. Salama, "Memristor-based memory: The sneak paths problem and solutions," Microelectronics journal, vol. 44, no. 2, pp. 176-183, Feb. 2013.

[91] F. Gül, "Addressing the sneak-path problem in crossbar RRAM devices using memristor-based one Schottky diode-one resistor array," Results in Physics, vol. 12, pp. 1091-1096, Mar. 2019.

[92] C. Li, Z. Wang, M. Rao, D. Belkin, W. Song, H. Jiang, P. Yan, Y. Li, P. Lin, M. Hu, N. Ge, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Long short-term memory networks in memristor crossbar arrays," Nature Machine Intelligence, vol. 1, no. 1, pp. 49-57, Jan. 2019.

[93] R. Midya, Z. Wang, S. Asapu, X. Zhang, M. Rao, W. Song, Y. Zhuo, N. Upadhyay, Q. Xia, and J. J. Yang, "Reservoir computing using diffusive memristors," Advanced Intelligent Systems, vol. 1, no. 7, pp. 1900084, Nov. 2019.

[94] W. Luo, W. Liu, and S. Gao, "Remembering history with convolutional lstm for anomaly detection," in IEEE International conference on multimedia and expo (ICME), pp. 439-444, Jul. 2017.

[95] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in Neural Information Processing: 25th International Conference (ICONIP), pp. 362-373, Dec. 2018.

[96] Z. Xu, Y. Wang, M. Long, and J. Wang, "PredCNN: Predictive Learning with Cascade Convolutions," in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI), pp. 2940-2947, Jul. 2018.

[97] L. Xiao, Y. Cao, Y. Gai, E. Khezri, J. Liu, M. Yang, "Recognizing sports activities from video frames using deformable convolution and adaptive multiscale features," Journal of Cloud Computing, vol. 12, no. 1, pp. 167, Dec. 2023.

[98] Y. Li, L. Chen, R. He, Z. Wang, G. Wu, and L. Wang L, "Multisports: A multi-person video dataset of spatio-temporally localized sports actions," In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13536-13545, 2021.

[99] S. Sarkar, P. J. Phillips, Z. Liu, I. R. Vega, P. Grother, and K. W. Bowyer, "The Human ID Gait Challenge Problem: Data Sets, Performance, and Analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 27, no. 2, pp. 162-177, Jan. 2005.

[100] S. Oladhri, "A Study of Deep Learning Silhouette Extractors for Gait Recognition," Doctoral dissertation, University of South Florida, 2022.

[101] S. I. Yusuf, S. A. Adeshina, and M. M. Boukar, "Upper gait analysis for human identification using convolutional–recurrent neural network," Journal of Theoretical and Applied Information Technology, vol. 100, no. 13, Jul. 2022.

[102] M. Hofmann, J. Geiger, S. Bachmann, B. Schuller, and G. Rigoll, "The TUM Gait from Audio, Image and Depth (GAID) database: Multimodal recognition of subjects and traits," Journal of Visual Communication and Image Representation, vol. 25, no. 1, pp. 195-206, Jan. 2014.

[103] E. D. de Jong, "Incremental sequence learning," arXiv preprint, vol. 1611.03068, Nov. 2016.

[104] S. Pi, C. Li, H. Jiang, W. Xia, H.L. Xin, J.J. Yang, and Q. Xia, "Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension," Nature Nanotechnology, vol. 14, no. 1, pp. 35-39, Jan. 2019.

[105] A. Amirsoleimani, F. Alibart, V. Yon, J. Xu, M. R. Pazhouhandeh, S. Ecoffey, Y. Beilliard, R. Genov, and D. Drouin, "In-Memory Vector-Matrix Multiplication in Monolithic Complementary Metal-Oxide-Semiconductor-Memristor Integrated Circuits: Design Choices, Challenges, and Perspectives," Advanced Intelligent Systems, vol. 2, no. 11, pp. 2000115, Nov. 2020.

[106] P. Caragiulo, O. E. Mattia, A. Arbabian, and B. Murmann, "A compact 14 gs/s 8-bit switched-capacitor dac in 16 nm finfet cmos," In IEEE Symposium on VLSI Circuits, pp. 1-2, Jun. 2020.

[107] C. Erdmann, E. Cullen, D. Brouard, R. Pelliconi, B. Verbruggen, J. Mcgrath, D. Collins, M. De La Torre, P. Gay, P. Lynch, and P. Lim, "16.3 A 330mW 14b 6.8 GS/s dual-mode RF DAC in 16nm FinFET achieving- 70.8 dBc ACPR in a 20MHz channel at 5.2 GHz," In IEEE International Solid-State Circuits Conference (ISSCC), pp. 280-281, Feb. 2017.

[108] J. K. Bha, and P. A. Priya, "Low power & high gain differential amplifier using 16 nm FinFET," Microprocessors and Microsystems, vol. 71, pp. 102873, Nov 2019.

[109] B. Hershberg, D. Dermit, B. van Liempd, E. Martens, N. Markulic, J. Lagos, and J. Craninckx, "3.1 A 3.2 GS/s 10 ENOB 61mW Ringamp ADC in 16nm with Background Monitoring of Distortion," In IEEE International Solid-State Circuits Conference-(ISSCC), pp. 58-60, Feb 2019.

[110] J. Wu, A. Chou, T. Li, R. Wu, T. Wang, G. Cusmai, S. T. Lin, C. H. Yang, G. Unruh, S. R. Dommaraju, and M. M. Zhang, "27.6 A 4GS/s 13b pipelined ADC with capacitor and amplifier sharing in 16nm CMOS," In IEEE International Solid-State Circuits Conference (ISSCC), pp. 466-467, Jan. 2016.

[111] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, and S. G. Tell, "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," IEEE Journal of Solid-State Circuits, vol. 55, no. 4, pp. 920-932, Jan. 2020.

[112] F. Karimzadeh, M. Imani, B. Asgari, N. Cao, Y. Lin, and Y. Fang, "Memory-based computing for energy-efficient AI: grand challenges," In 2023 IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1-8, Oct. 2023.

[113] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, and S. Joshi, "A compute-in-memory chip based on resistive random-access memory" Nature, vol. 608, no. 7923, pp. 504-512, Aug. 2022.

[114] B. Gao, Y. Zhou, Q. Zhang, S. Zhang, P. Yao, Y. Xi, Q. Liu, M. Zhao, W. Zhnag, Z. Liu, and X. Li, "Memristor-based analogue computing for brain-inspired sound localization with in situ training," Nature Communications, vol. 13, no. 1, pp. 2026, Dec. 2022.

[115] I. Chakraborty, D. Roy, and K. Roy, "Technology Aware Training in Memristive Neuromorphic Systems for Nonideal Synaptic Crossbars," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 5, pp. 335-344, Oct. 2018.

[116] Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. P. Strachan, and N. Ge, "In situ training of feed-forward and recurrent convolutional memristor networks," Nature Machine Intelligent, vol. 1, no. 9, pp. 434-442, Sep. 2019.

[117] I. Chakraborty, A. Jaiswal, A. K. Saha, S. K. Gupta, and K. Roy, "Pathways to efficient neuromorphic computing with non-volatile memory technologies," Applied Physics Reviews, vol. 7, no. 2, Jun. 2020.

[118] Z. Wang, H. Wu, G. W. Burr, C. S. Hwang, K. L. Wang, Q. Xia, and J. J. Yang, "Resistive switching materials for information processing," Nature Reviews Materials, vol. 5, no. 3, pp. 173-195, Mar. 2020.

[119] J. M. Hung, C. X. Xue, H. Y. Kao, Y. H. Huang, F. C. Chang, S. P. Huang, T. W. Liu, C. J. Jhang, C. I. Su, W. S. Khwa, and C. C. Lo, "A four-megabit compute-in-memory macro with eight-bit precision based on CMOS and resistive random-access memory for AI edge devices," Nature Electronics, vol. 4, no. 12, pp. 921-930, Dec. 2021.

[120] H. Zhao, Z. Liu, J. Tang, B. Gao, Y. Zhang, H. Qian, and H. Wu, "Memristor-based signal processing for edge computing," Tsinghua Science and Technology, vol. 27, no. 3, pp. 455-71, Nov. 2021.

[121] A. Singh, S. Diware, A. Gebregiorgis, R. Bishnoi, F. Catthoor, R. V. Joshi, and S. Hamdioui, "Low-power memristor-based computing for edge-ai applications," In IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, May 2021.

[122] D. Lee, M. Park, Y. Baek, B. Bae, J. Heo, and K. Lee, "In-sensor image memorization and encoding via optical neurons for bio-stimulus domain reduction toward visual cognitive processing," Nature Communications, vol. 13, no. 1, pp. 5223, Dec. 2022.

[123] Z. Zhao, J. Tang, J. Yuan, Y. Li, Y. Dai, J. Yao, Q. Zhang, S. Ding, T. Li, R. Zhang, and Y. Zheng, "Large-Scale Integrated Flexible Tactile Sensor Array for Sensitive Smart Robotic Touch," ACS Nano, vol. 16, no. 10, pp. 16784-16795, Oct. 2022.

[124] Z. Zhang, X. Zhao, X. Zhang, X. Hou, X. Ma, S. Tang, Y. Zhang, G. Xu, Q. Liu, and S. Long, "In-sensor reservoir computing system for latent fingerprint recognition with deep ultraviolet photo-synapses and memristor array," Nature Communications, vol. 13, no. 1, pp. 6590, Dec. 2022.

[125] C. Wan, G. Chen, Y. Fu, M. Wang, N. Matsuhisa, S. Pan, L. Pan, H. Yang, Q. Wan, L. Zhu, and X. Chen, "An Artificial Sensory Neuron with Tactile Perceptual Learning," Advanced Materials, vol. 30, no. 30, pp. 1801291, Jul. 2018.

[126] F. Kiani, and J. Yin, "A fully hardware-based memristive multilayer perceptron," The International Conference on Memristive Materials, Devices & Systems (MEMRISYS’22), Nov 2022.

[127] X. Zhang, Y. Zhuo, Q. Luo, Z. Wu, R. Midya, Z. Wang, W. Song, R. Wang, N.K. Upadhyay, Y. Fang, F. Kiani, M. Rao, Y. Yang, Q. Xia, Q. Liu, M. Liu, and J. J. Yang,

"An artificial spiking afferent nerve based on Mott memristors for neurorobotics," Nature communications, vol. 11, no. 1, pp. 51, Jan. 2020.

[128] M. Rao, W. Song, F. Kiani, S. Asapu, Y. Zhuo, R. Midya, N. Upadhyay, Q. Wu, M. Barnell, P. Lin, C. Li, Z. Wang, Q. Xia, and J. J. Yang, "Timing selector: Using transient switching dynamics to solve the sneak path issue of crossbar arrays," Small Science, vol. 2, no. 1, pp. 2100072, Oct. 2021.

[129] F. Ye, F. Kiani, Y. Huang, and Q. Xia, "Diffusive memristors with uniform and tunable relaxation time for spike generation in event-based pattern recognition," Advanced Materials, vol. 35, no. 37, pp. 2204778, Sep. 2023.

[130] F. Ye, and F. Kiani, "Hierarchy of Event-Based Time-Surfaces Based on Diffusive Memristors with Uniform and Tunable Relaxation Time-A Preliminary Study," The International Conference on Memristive Materials, Devices & Systems (MEMRISYS'22), Nov. 2022.

[131] Y. Huang, F. Kiani, F. Ye, and Q. Xia, "From memristive devices to neuromorphic systems," Applied Physics Letters, vol. 122, no. 11, pp. 110501, Mar. 2023.

[132] M. Rao, H. Tang, J. B. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang, H. Liu, H. Y. Chen, R. Midya, F. Ye, H. Jiang, Z. Wang, M. Wu, M. Hu, H. Wang, Q. Xia, N. Ge, J. Li, and J. J. Yang, "Thousands of conductance levels in memristors monolithically integrated on CMOS," Nature, vol. 615, no. 7954, pp. 823-829, Mar. 2023.

[133] F. Kiani, and Q. Xia, "Bipolar flash for bifunctional computing operations," Nature Nanotechnology, vol. 18, no. 5, pp. 444-445, May 2023.